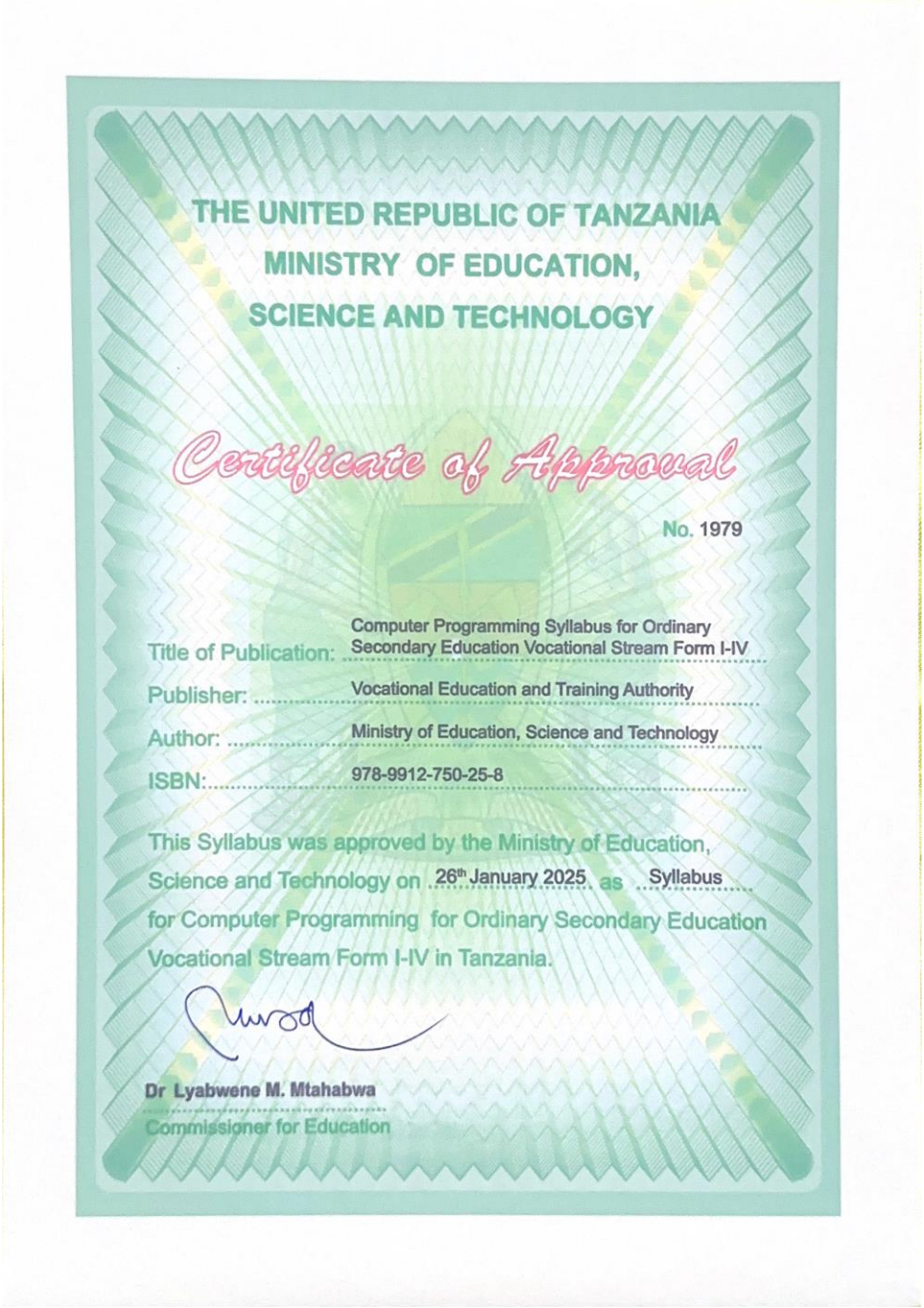
THE UNITED REPUBLIC OF TANZANIA

MINISTRY OF EDUCATION, SCIENCE AND TECHNOLOGY



THE UNITED REPUBLIC OF TANZANIA
MINISTRY OF EDUCATION,
SCIENCE AND TECHNOLOGY

Certificate of Approval

No. 1979

Title of Publication: Computer Programming Syllabus for Ordinary Secondary Education Vocational Stream Form I-IV

Publisher: Vocational Education and Training Authority

Author: Ministry of Education, Science and Technology

ISBN: 978-9912-750-25-8

This Syllabus was approved by the Ministry of Education, Science and Technology on 26th January 2025 as Syllabus for Computer Programming for Ordinary Secondary Education Vocational Stream Form I-IV in Tanzania.

Dr Lyabwene M. Mtahabwa
Commissioner for Education

COMPUTER PROGRAMMING SYLLABUS FOR ORDINARY SECONDARY EDUCATION VOCATIONAL STREAM FORM I-IV

Published 2023

Revised 2025

Vocational Education and Training Authority (VETA)

12 VETA Road,

41104 Tambukareli,

P.O. BOX 802,

Dodoma - Tanzania,

Telephone: +255 26 2963661

Website: www.veta.go.tz

Email: info@veta.go.tz

ISBN: 978-9912-750-25-8

This document should be cited as: Ministry of Education, Science and Technology. (2025). *Computer Programming Syllabus for Ordinary Secondary Education Vocational Stream Form I-IV*. Vocational Education and Training Authority.

Table of Contents

List of Tables

Abbreviations and Acronyms

CBET	-	Competence Based Education and Training Approach.
CIA	-	Confidentiality, Integrity and Availability
GPA	-	Grade Point Average
ICT	-	Information and Communication Technology
IDE	-	Integrated Development Environment
PHP	-	PHP: Hypertext Preprocessor
SQL	-	Structured Query Language
VET	-	Vocational Education and Training
VETA	-	Vocational Education and Training Authority
VS Code	-	Visual Studio Code

Definition of Key Terms

Assessment: The process of collecting evidence and making judgments on whether competency has been achieved or whether specific skills and knowledge have been achieved that will lead to the attainment of competency.

Circumstantial knowledge: Detailed knowledge that allows decision-making regarding different circumstances and cross-cutting issues.

Competence: The ability to use knowledge, understanding, practicality, and thinking skills to perform effectively to the workplace standards required in employment.

Element: A sub-unit (step) which reflects the learning sequence to achieve a unit's broad learning objectives.

Occupational Standards: Specific requirements of competencies people are expected to demonstrate in a particular occupational area, including knowledge and relevant attitudes. They also act as performance tools for the assessment of the prescribed outcomes.

Performance criteria: Indicate the expected results or outcome through evaluative statements.

Standard: A set of statements, which, if proven true under working conditions, means that an individual is meeting an expected level and type of performance.

Underpinning Knowledge: This is essential knowledge needed to demonstrate competencies associated with performing a given task.

Unit: A statement of broad learning objectives, which prescribe the requirements of a standard in the form of practical skills, knowledge and appropriate attitudes.

Acknowledgements

The writing of the Computer Programming Syllabus for Ordinary Secondary Education Vocational Stream Form I-IV was a collaborative effort that involved the dedication and expertise of a wide range of organisations and individuals. Vocational Education and Training Authority (VETA) would like to thank all the organizations and experts who contributed to developing this Syllabus. VETA appreciates the expertise of individuals and their time, effort, and resources that were devoted to this important task. Their contributions have been crucial in developing a relevant and comprehensive Syllabus aimed at equipping students with the skills necessary for success in their fields. Furthermore, valuable inputs from employers in both formal and informal sectors during labour market surveys are acknowledged. Likewise, VETA thanks the Ministry of Education, Science and Technology uniquely for facilitating this Syllabus's preparation, printing and distribution.

For and on behalf of:

Vocational Education and Training Authority

CPA. Antony M. Kasore

Director General

1.0. Introduction

Computer Programming is one of the occupations taught in the Ordinary Secondary Education Vocational Stream. Learning Computer Programming is essential because Tanzania has a growing demand for technology-driven solutions and skilled IT professionals capable of addressing local and global challenges. As advancements in technology reshape industries, the need for modern digital infrastructure, automated processes, and data-driven decision-making continues to rise. This creates immense opportunities for economic growth, technological development, innovation, and employment across various sectors.

By acquiring computer programming skills, students can develop software, systems, and applications tailored to specific needs, such as mobile apps that improve access to services at home, electronic health systems that enhance patient care in hospitals, e-learning platforms that transform education in schools, e-government systems that streamline public services, and financial tools that support businesses and the banking sector. These innovations contribute to local industry development, reduce dependency on costly imported or proprietary software, and increase Tanzania's capacity for technological self-reliance.

Moreover, programming skills empower individuals to participate in the global digital economy by creating exportable software solutions, fostering entrepreneurship, and promoting digital inclusion. This, in turn, drives sustainable development, enhances national competitiveness, strengthens digital literacy, and improves the quality of life for Tanzanians. Ultimately, a well-developed pool of programmers is key to building a robust knowledge economy, fostering innovation, and ensuring that Tanzania keeps pace with the rapidly evolving technological landscape.

An occupation is a specific work area or a group of related job roles that demand particular skills, knowledge, and competencies. It encompasses a structured professional activity within the labour market, marked by distinct tasks, responsibilities, and established standards of practice. In the context of Computer Programming, an occupation involves performing duties related to creating web, mobile, and desktop applications, designing and implementing databases, and analysing data. Computer programming involves writing, testing, and maintaining code that enables computer systems and applications to perform specific tasks or solve problems. It encompasses the use of programming languages to create algorithms, develop software, and control hardware.

Upon completion of the program, students will possess both theoretical and practical knowledge of computer programming, from understanding fundamental concepts and algorithms to designing, coding, testing, and maintaining software solutions. They will be proficient in using various programming languages, developing web and mobile applications, creating databases, and implementing software systems tailored to solve real-world problems. Additionally, students will gain business skills critical for managing technology-driven projects, collaborating in interdisciplinary teams, and creating innovative solutions that meet market demands.

Graduates of this occupation have diverse employment opportunities across various sectors, including the public sector, private sector, and non-governmental organizations. In the public sector, they can work as IT support assistants, junior web developers, database support technicians, junior programmers, or mobile app support assistants in institutions such as government ministries, municipal councils, and public training institutions. In the private sector, they may be employed as web designers, software

developers in tech companies and start-ups, web developers for e-commerce platforms, desktop application developers for small and medium enterprises, or instructors in private training institutions. Additionally, in non-governmental organizations, they can serve as software developers, web developers, or desktop application developers, contributing to various development initiatives.

The Computer Programming Syllabus is designed to guide the teaching and learning of computer programming at the Ordinary Secondary Education Form I-IV Vocational Stream in the United Republic of Tanzania. The syllabus interprets the competences a student needs to develop while learning Computer Programming. It contains valuable information that will enable teachers to effectively plan their teaching process and help learners to develop the intended competences.

2.0. Main Objectives of Education in Tanzania

The main objectives of education in Tanzania are to enable every Tanzanian to:

(a) Develop and improve his or her personality so that he or she values himself or herself and develops self-confidence;
(b) Respect the culture, traditions, norms and customs of Tanzania; cultural differences; dignity; human rights; attitudes and inclusive actions;
(c) Advance knowledge and apply science and technology, creativity, critical thinking, innovation, cooperation, communication and positive attitudes for his or her own development and the sustainable development of the nation and the world at large;
(d) Understand and protect national values, including dignity, patriotism, integrity, unity, transparency, honesty, accountability and the national language;
(e) Develop life and work-related skills to increase efficiency in everyday life;
(f) Develop a habit of loving and valuing work to increase productivity and efficiency in production and service provision;
(g) Identify and consider cross-cutting issues, including the health and well-being of the society, gender equality, as well as the management and sustainable conservation of the environment; and
(h) Develop national and international cooperation, peace and justice per the Constitution of the United Republic of Tanzania and international conventions.

3.0. General Competencies for Ordinary Secondary Education Vocational Stream

The general competences for Ordinary Secondary Education, Form 1–IV, Vocational Education stream are to:

(a) Apply the knowledge, skills and attitudes the student developed in the primary school stage to increase his/her understanding of technical skills;
(b) Apply technical skills in designing, inventing and making various things to cope with life and solve challenges in society;
(c) Appreciate citizenship and national virtues;

(d) Use language skills;
(e) Demonstrate self-confidence in learning in various fields, including science and technology, technical knowledge and technical skills;
(f) Apply technical knowledge and skills in designing, discovering and making various things to solve challenges in society, including cross cutting issues;
(g) Appreciate procedures and safety rules in using technical tools correctly; and
(h) Apply the technical knowledge and skills acquired to develop oneself with vocational and technical education and join the workforce.

4.0. General Competences of the Occupation

Upon completion of this occupation, students are expected to have ability to:

- Solve programming problems using pseudo-code, flowcharts, and logical control structures;
- Write computer programs using different programming paradigms;
- Analyze system requirements and design system architectures and user interfaces;
- Design and implement websites using HTML, CSS, and JavaScript;
- Create and host websites and mobile applications effectively;
- Develop and manage database-driven applications through advanced programming techniques;
- Apply cybersecurity measures in programming by implementing encryption, data validation, and threat mitigation;
- Develop web and mobile applications by integrating databases and scripting languages; and
- Construct user-friendly mobile applications, including user interface design, event handling, and deployment.

5.0. Main and Specific Competences

The main and specific competences to be developed are presented in Table 1

Table 1: Main and Specific Competences for Form I-IV

Main Competence	**Specific competences**
1. Implementing computer applications	1.1. Maintaining computer hardware components 1.2. Managing word processing 1.3. Managing spreadsheet and data presentation
2. Basics of computer programming	2.1. Designing of computer programs 2.2. Implementing control structures in programming 2.3. Implementing functions 2.4. Implementing arrays in a program
3. Designing and hosting websites	3.1. Creating web pages using HTML 3.2. Formatting web pages with Cascading Style Sheets (CSS) 3.3. Implementing web scripting 3.4. Hosting and publishing websites 3.5. Designing websites using content management systems
4. Developing database systems	4.1. Implementing database modelling and relationships 4.2. Implementing physical database structure with structured

Main Competence	Specific competences
	query language
5. Developing database-driven web applications	5.1. Creating PHP programs 5.2. Working with forms 5.3. Linking forms to databases for interactive web applications
6. System analysis and design	6.1. Identifying and analysing software development requirements 6.2. Building structures of the software
7. Web programming frameworks	7.1. Identifying and setting up a web development environment 7.2. Building the front-end using a framework 7.3. Building the back-end using a framework
8. Basics of Object-Oriented Programming	8.1. Creating basic programs using classes and objects 8.2. Creating programs with inheritance and method overriding 8.3. Implementing encapsulation and abstraction 8.4. Implementing polymorphism with interfaces and abstract classes
9. Event-Driven Programming	9.1. Working with IDE and event-driven language 9.2. Developing programs using functions and arrays 9.3. Designing form with menus controls 9.4. Creating database connectivity options and reporting
10. Mobile Application Development	10.1. Working with mobile apps and development platforms 10.2. Handling user input and basic interactions 10.3. Working with data and multimedia 10.4. Optimising and deploying mobile apps
11. Integrating Cybersecurity Measures into Applications	11.1. Identifying and controlling cybersecurity threats 11.2. Practising safe programming practices 11.3. Encrypting and decrypting data 11.4. Handling sensitive information in programming
12. Integrating programming with data science	12.1. Creating basic programs 12.2. Collecting and storing data 12.3. Cleaning and organising data 12.4. Analysing data 12.5. Visualising data

6.0. The Roles of Teachers, Students and Parents in Teaching and Learning

Good relationships between a teacher, student and parent, or guardian is fundamental to ensuring successful learning. This section outlines the roles of each participant in facilitating effective teaching and learning of Computer Processing.

6.1. The Teacher

The teacher is expected to:

(a) Help the student to learn and develop the intended competences in Computer Programming;

(b) Use teaching and learning approaches that will allow students with different needs and abilities to:

i. Develops the competences needed in the 21st Century; and

ii. Actively participate in the teaching and learning process.

(c) Use student centered instructional strategies that make the student a centre of learning, which allow them to think, reflect and search for information from various sources;
(d) Create a friendly teaching and learning environment;
(e) Prepare and improvise teaching and learning resources;
(f) Conduct formative assessment regularly by using tools and methods which assess theory and practice;
(g) Treat all the students according to their learning needs and abilities;
(h) Protect the student from a risky environment while he or she is at school;
(i) Keep track of the student's daily progress;
(j) Identify individual student's needs and provide the proper intervention;
(k) Involve parents/guardians and the society at large in the student's learning process; and
(l) Integrate cross-cutting issues and ICT in the teaching and learning process.

6.2. The Student

The student is expected to:

(a) Develop the intended competences by participating actively in various learning activities inside and outside the classroom; and
(m) Participate in the search for knowledge from various sources, including textbooks, reference books and other publications in online libraries.

6.3. The Parent/Guardian

The Parents/Guardian is expected to:

(a) Monitor the child's academic progress in school;
(n) Where possible, provide a child with the needed academic support;
(o) Provide a child with a safe and friendly home environment which is conducive for learning;
(p) Keep track of a child's progress in behaviour;
(q) Provide the child with any necessary materials required in the learning process; and
(r) Instill in a child a sense of commitment and positive value towards education and work.

7.0. Teaching and Learning Methods

The teaching and learning methods are instrumental in developing student's competences. This syllabus suggests teaching and learning methods for each activity which includes but not limited to demonstration, practical/hands-on activities, observations, role play, simulation, group works, peer teaching/learning, discussions, presentations, field visits, research, and project works. However, a teacher is advised to plan and use other appropriate methods based on the environment or context. All the teaching and learning methods should be integrated with the everyday lives of students. The focus is expected to be on practical application and

developing cognitive, affective, and psychomotor skills through learner-centred methods. Vocational teachers act as facilitators, incorporating both school base teaching and project work supervision.

8.0. Teaching and Learning Resources

The process of teaching and learning requires different resources. In that regard, both a teacher and students should work together to collect or improvise alternative resources available in the school and home environment when needed. Teachers and students are expected to constantly seek for information from various sources to effectively facilitate the teaching and learning process. The list of approved textbooks and reference books shall be provided by the TIE.

9.0. Assessment

Assessment is important in teaching and learning of Computer Programming. It is divided into formative and summative assessments. Formative assessment informs both the teacher and students on the progress of teaching and learning, and in making decisions on improving the teaching and learning process. Teachers are therefore, expected to apply a wide range of formative assessment methods which include but not limited to demonstration, discussions, presentations, oral questions, experiments, observations, practical assignments and projects.
Summative assessment, on the other hand, will focus on determining student's achievement of learning. Teachers are expected to use a variety of summative assessments including Form Two National Assessment, terminal examination, annual examination, mock examination and project. The scores obtained from these assessments will be used as Continuous Assessment (CA). Therefore, the continuous assessments shall contribute 60% and the National Form IV Examination shall be 40% as indicated in Table 2.

Table 2: *Contribution of Continuous Assessment and National Examination in the Final Score*

Assessment Category	**Weight (%)**	**National Examination**
Form Two National Assessment (FTNA)	6.0	**40**
Form Three Terminal Examination	5.0	
Form Three Anual Examination	5.0	
Form Four Mock Examination	7.0	
Project	7.0	
Form Two Practical	10.0	
Form Three Practical	10.0	
Form Four Practical	10.0	
Total	**60**	

10.0. Project Work

Project work is a carefully planned and clearly defined task or problem that a student undertakes, either alone or in a group, to enhance and apply the skills and knowledge gained in the classroom, workshop, kitchen, or laboratory. It is based on the principles of "Learning by Doing" and "Learning by Living." In this context, the implementation of Project Work in

secondary schools' vocational streams is essential. Projects in the vocational stream should be conducted in the core subject (occupation). To ensure its success, the supervision and assessment of student project work must be consistent with the established guidelines provided by National Examinations Council of Tanzania (NECTA).

11.0. Number of Periods

The Computer Programming Syllabus for Ordinary Secondary Education Vocational Stream Form I-IV provides time estimates for teaching and learning each specific competence. The estimates consider the complexity of the specific competences and the learning activities. Eight (08) periods of 40 minutes each have been allocated per week, whereby two (02) periods will be used for theory and 6 for practical sessions which may require double periods (e.g., 80). Double periods will allow sufficient time for hands-on activities.

12.0. Teaching and Learning Contents

The contents of the Syllabus are organised into a matrix with seven (07) columns, which are main competences, specific competences, learning activities, suggested teaching and learning methods, assessment criteria that is divided into (process assessment, products/service assessment and underpinning knowledge), suggested teaching and learning resources and number of periods as presented in Table 3 to 6.

Form One

Table 3: *Detailed contents for Form One*

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
1. Implementing computer applications	1.1. Maintaining computer hardware components	a) Setting up the installation of computer hardware components	**Demonstrations:** Show the process of hardware installation and testing. **Hands-On Activities**: Provide trainees with broken or functional systems to work on. **Problem-Solving Tasks:** Assign tasks to diagnose and repair simulated hardware faults. Use of Multimedia: Show videos of real-life hardware setups and installations.	The student should be able to: • Identify computer hardware components • Identify hardware components accurately. • Follow safety procedures (e.g., grounding to avoid electrostatic discharge). • Select appropriate tools for installation tasks. • Apply systematic procedures for installing and configuring hardware.	The installed hardware components are compatible, function as intended, and the systems have been verified to boot successfully.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Identify and handle different types of hardware components. • Properly install and configure hardware within a system. • Use diagnostic tools to test hardware functionality. **Principles:** The student should explain the principle of: • Compatibility of components (e.g., motherboard and CPU socket types). • Power supply	The following tools, equipment and safety gear are to be available: • Desktop computers (functional and non-functional for practice). • Complete computer toolkits (screwdrivers, anti-static wristbands, thermal paste). • Diagnostic tools (multimeters, POST cards). • Hardware components • Instruction manuals for hardware components. • Multimedia resources such as tutorials and animations. • Mock-up hardware for hands-on practice	32

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						requirements and distribution. **Theories:** The trainee should explain: The trainee should explain: • Data transfer theories within computer hardware (e.g., bus systems, storage hierarchies). • Troubleshooting techniques and fault diagnosis models **Circumstantial knowledge** Detailed knowledge about: • Hardware specifications and compatibility charts. • Common issues and solutions for hardware installation. • Industry safety standards for hardware		

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						maintenance		
		b) Performing the maintenance and repair of the computer system	**Demonstration:** Show students how to identify, diagnose, and repair common hardware and software issues in a computer system. **Hands-On Practice:** Provide students with malfunctioning computer systems for troubleshooting and repair under supervision. **Role-Playing:** Assign roles such as technician and client to simulate real-world repair scenarios and communication skills. **Case Studies:** Analyze real-world examples of hardware and software failures,	The student should be able to: • Identify hardware and software issues accurately. • Select appropriate tools and techniques for maintenance and repair tasks. • Follow systematic diagnostic procedures. • Apply safety precautions during maintenance and repair activities.	Maintenance and repair of the computer system performed according industry standards and practices.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use diagnostic tools and software for troubleshooting hardware and software issues. • Perform maintenance tasks such as cleaning, component replacement, and updates. • Safely dismantle and reassemble computer systems. **Principles:** The student should explain the principle of: • System diagnostics and error detection. • Preventive maintenance for extending the lifespan of	The following tools, equipment and safety gear are to be available: • Diagnostic tools (e.g., multimeters, POST cards, software utilities). • Basic and advanced toolkits (e.g., screwdrivers, anti-static wristbands, thermal paste). • Spare hardware components for repair practice. • Laptops, desktops, and servers for hands-on maintenance. • Maintenance and repair manuals. • Multimedia resources such as tutorial videos and interactive simulations. • Case studies of computer system failures and their solutions.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			their causes, and resolutions. Interactive **Discussions:** Facilitate group discussions on common problems encountered during maintenance and their preventive measures. **Multimedia Resources:** Use videos, diagrams, and simulations to illustrate repair processes and system diagnostics.			computer systems. • Component compatibility and performance optimization. **Theories:** The trainee should explain: The trainee should explain: • Theories of system failure and recovery (e.g., boot failure, memory errors). • Data recovery principles and techniques. **Circumstantial knowledge** Detailed knowledge about: • Common hardware and software problems and their solutions. • Safe handling of internal components such as motherboards,		

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						hard drives, and CPUs. • Proper disposal of non-functional components and e-waste.		
	1.2 Managing word processing	a) Managing files and printing	**Demonstration:** Show how to create, save, organize, and retrieve files within a word processing application. **Hands-On Practice:** Assign students tasks to manage files and configure print settings for different types of documents. Interactive **Tutorials:** Use step-by-step guides and videos to teach file management and printing techniques. **Case Studies:** Analyze scenarios	The student should be able to: • Create, name, and organize files into appropriate folders. • Retrieve, rename, or delete files securely. • Select correct printing options, including page size, orientation, and quality. • Handle printer setup and troubleshoot common printing issues.	Files are managed properly and printed using the appropriate layout.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Create, save, and organize files systematically. • Configure advanced printing options, such as duplex printing and specific page ranges. • Use printer settings to optimize output quality. **Principles:** The student should explain the principle of: • File organization and version control. • Printing	The following tools and equipment are to be available: • Computers with word processing software (e.g., Microsoft Word, Google Docs, LibreOffice). • Functional printers with varying capabilities (e.g., color, duplex). • File storage options such as local drives, USBs, and cloud storage. • User manuals for word processing software and printers. • Tutorial videos and interactive simulations. • Case studies on file management and printing challenges.	76

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			where proper file management and printing techniques enhance productivity.			standards and quality assurance. • Data security in file handling. **Theories:** The trainee should explain: The trainee should explain: • Theories of file compression and storage management. • The functioning of printing technologies and devices. **Circumstantial knowledge** Detailed knowledge about: • File formats and their compatibility with different word processors. • Troubleshooting techniques for common file and printer errors.		
		b) Managing editing and	**Demonstration:** Show the use of	The student should be able to:	Edited documents are free from	**Detailed knowledge of:**	The following tools and equipment are to be	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
		formatting documents	various editing and formatting tools in word processing software. **Hands-On Practice:** Assign tasks to students to edit and format documents, applying features like text alignment, styles, and page layouts. **Interactive Tutorials:** Provide step-by-step guides and multimedia resources to explain advanced editing and formatting techniques. **Group Discussions:** Facilitate discussions about formatting standards for different types of documents (e.g.,	• Edit text using tools such as find and replace, track changes, and comments. • Format text using features like font styles, paragraph alignment, and spacing. • Apply advanced formatting techniques, such as tables, headers/footers, and page numbering. • Review and proofread documents for errors.	grammatical and typographical errors, adheres to formatting standards.	**Methods:** The trainee should explain how to: • Use editing tools such as track changes and comments for collaborative work. • Format text, paragraphs, and pages to meet specific document requirements. • Apply templates and styles for consistency. **Principles:** The student should explain the principle of: • Document design and layout for readability and professionalism. • Typography and its impact on document clarity. **Theories:** The trainee should explain: The trainee should	available: • Computers with word processing software (e.g., Microsoft Word, Google Docs, LibreOffice). • Sample documents for editing and formatting exercises. • Access to templates and style guides for different document types. • User manuals and tutorials for word processing software. • Multimedia resources, such as instructional videos and interactive guides. • Case studies and examples of professional document formatting.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			official letters, reports). Problem-Solving **Activities:** Challenge students to reformat poorly designed documents into professional, well-structured outputs. **Case Studies:** Analyze examples of well-formatted and poorly formatted documents to highlight best practices.			explain: • Theories of information presentation and visual hierarchy. • The impact of document aesthetics on communication effectiveness. **Circumstantial knowledge** Detailed knowledge about: • Common formatting errors and how to avoid them.		
		c) Creating tables	**Demonstration:** Show how to insert, modify, and format tables using word processing software. **Hands-On Practice:** Assign students tasks to create and format tables for	The student should be able to: • Insert and configure tables accurately in a document. • Modify table structure, including adding/removing rows and columns.	Tables are created with proper alignment and formatting to enhance clarity and readability.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Insert and customize tables in a word processing document. • Use formatting tools to enhance the appearance	The following tools and equipment are to be available: • Computers with word processing software (e.g., Microsoft Word, Google Docs, LibreOffice). • Sample data for creating tables (e.g., schedules, financial records, reports).	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			different purposes, such as data presentation and layout design. **Interactive Tutorials:** Provide multimedia guides to explain advanced table features like merging cells, applying formulas, and customizing borders. **Problem-Solving Activities:** Challenge students to recreate or modify existing tables to meet specific formatting and design requirements. **Group Projects:** Facilitate collaborative projects where students create tables for real-	• Apply formatting features such as borders, shading, and alignment.		of tables. **Principles:** The student should explain the principle of: • Data organization and logical structure within tables. • Visual hierarchy and its role in table design. • Alignment and spacing to enhance readability. **Theories:** The trainee should explain: The trainee should explain: • Data presentation and its impact on communication. • Effective use of visual elements to guide the reader's attention. **Circumstantial knowledge** Detailed knowledge	• User manuals and video tutorials for table creation and formatting. • Examples of well-designed tables for reference. • Interactive guides on advanced table features (e.g., formulas, sorting).	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			world applications, such as schedules or reports.			about: • Common table formatting errors and how to avoid them. • Best practices for presenting complex data in tables.		
	1.3 Managing spreadsheet and data presentation	a) Managing components, worksheets and formatting	**Demonstration:** Show how to manage spreadsheet components such as rows, columns, and cells, and demonstrate worksheet creation and formatting techniques. **Hands-On Practice**: Assign students tasks to manage multiple worksheets, adjust formatting, and apply styles for readability and presentation. **Interactive Tutorials**: Use	The student should be able to: • Insert and adjust rows, columns, and cells in a spreadsheet • Create, rename, and manage multiple worksheets. • Apply formatting features such as font styles, borders, shading, and conditional formatting. • Organize data into logical categories and use sorting and	The spreadsheet is well-organized, with structured worksheets and clear, visually appealing formatting.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Format cells, rows, and columns to improve the readability of data. • Use worksheet management tools, such as grouping and linking worksheets. • Apply conditional formatting to emphasize important data points. **Principles:** The	The following tools and equipment are to be available: • Computers with spreadsheet software (e.g., Microsoft Excel, Google Sheets, LibreOffice Calc). • Sample data sets for formatting and presentation exercises. • Templates for professional spreadsheet designs. • User manuals and interactive guides for spreadsheet tools. • Case studies of effective spreadsheet designs. • Video tutorials for advanced formatting	99

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			videos and step-by-step guides to teach advanced features, such as conditional formatting and cell protection. **Problem-Solving Activities:** Provide scenarios where students must organize and format data to meet specific requirements. **Group Projects:** Encourage collaborative work on larger datasets that require efficient management and presentation. **Case Studies:** Analyze examples of well-formatted spreadsheets and discuss their effectiveness in data presentation.	filtering tools effectively.		student should explain the principle of: • Effective data organization and visual hierarchy. • Formatting standards for professional data presentation. **Theories:** The trainee should explain: The trainee should explain: • Visual data communication and its impact on understanding. • The importance of consistency in spreadsheet formatting for collaboration. **Circumstantial knowledge** Detailed knowledge about: • Common formatting errors and how to prevent them. • The role of	and worksheet management techniques.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						templates and pre-defined styles in professional spreadsheets.		
		b) Implementing conditions and data	**Demonstration:** Show the use of conditional functions (e.g., IF, COUNTIF, SUMIF) and applying conditional formatting for data analysis. **Hands-On Practice:** Assign students exercises to implement conditions using formulas and rules in spreadsheets. **Interactive Tutorials:** Provide step-by-step instructions and video resources for creating and applying conditional logic in data	The student should be able to: • Write and apply conditional formulas (e.g., IF, COUNTIF, AVERAGEIF) • Use conditional formatting to highlight specific data points. • Validate data using custom rules and input restrictions. • Organize and analyze data sets with logical conditions.	Conditional formulas are correctly applied to data.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use conditional formulas to automate calculations and data analysis. • Apply rules for conditional formatting to emphasize trends or outliers. **Principles:** The student should explain the principle of: • Logical operations and their application in spreadsheet conditions. • Data validation and error prevention techniques.	The following tools and equipment are to be available: • Computers with spreadsheet software (e.g., Microsoft Excel, Google Sheets, LibreOffice Calc). • Sample data sets for applying conditional logic and validation rules. • Pre-designed templates for conditional formatting exercises. • User manuals and video tutorials for conditional functions and formatting. • Case studies of effective conditional logic in business and data management. • Interactive guides and real-world scenarios for	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			processing. **Problem-Solving Activities:** Present real-world problems that require conditional logic, such as calculating bonuses based on performance thresholds. **Case Studies:** Analyze examples where conditions and data organization improved decision-making or data visualization. **Group Projects:** Encourage teams to work on datasets, applying conditional logic and formatting to prepare reports.			• Effective use of formatting to convey data insights. **Theories:** The trainee should explain: The trainee should explain: • Conditional logic and its applications in decision-making. • The role of data visualisation in enhancing data interpretation **Circumstantial knowledge** Detailed knowledge about: • Conditional formulas and their real-world applications. • Common errors in conditional logic and how to troubleshoot them	practice.	
		c) Implementing functions	**Demonstration:** Illustrate how to use various	The student should be able to: • Use basic and	Appropriate functions and formulas applied to automate	**Detailed knowledge of:** **Methods:** The trainee	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
		and formulas	spreadsheet functions (e.g., mathematical, statistical, text, logical) and create formulas. **Hands-On Practice:** Assign tasks to students to implement functions and formulas for solving real-world problems, such as financial calculations or data analysis. **Interactive Tutorials:** Provide step-by-step instructions and video resources for creating and applying advanced formulas. **Problem-Solving Activities:** Present datasets requiring students to use multiple	advanced spreadsheet functions correctly (e.g., SUM, AVERAGE, VLOOKUP, IF). • Create complex formulas combining multiple functions. • Apply cell referencing methods (absolute, relative, mixed).	calculations for producing accurate and reliable results.	should explain how to: • Use various categories of functions (e.g., text, logical, lookup) for different tasks. • Combine multiple functions within a single formula to solve complex problems. **Principles:** The student should explain the principle of: • Formula design for scalability and reusability. • Logical operations and their application in decision-making functions. **Theories:** The trainee should explain: The trainee should explain: • Data analysis	• Computers with spreadsheet software (e.g., Microsoft Excel, Google Sheets, LibreOffice Calc). • Sample datasets for practice with formulas and functions. • Templates for common business applications (e.g., financial models, attendance trackers). • User manuals and video tutorials for advanced spreadsheet functions. • Case studies of successful spreadsheet applications. • Practice exercises for troubleshooting and debugging formulas.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			functions to analyze and generate results. **Group Projects:** Assign collaborative projects to create comprehensive spreadsheets with interlinked formulas. **Case Studies:** Examine real-world examples where functions and formulas were used to streamline operations or make decisions.			and how functions enhance insights. **Circumstantial knowledge** Detailed knowledge about: • Common errors in formulas and how to resolve them. • Advanced functions like INDEX, MATCH, and ARRAYFORMULA.		
		d) Implementing data visualisation with charts and tables	**Demonstration:** Show how to create and format charts and tables in spreadsheet software, including customization of labels, axes, and legends. **Hands-On Practice:** Assign	The student should be able to: • Select suitable chart types for different data scenarios. • Create and format tables for clear and logical data presentation	Charts and tables that accurately represent the data visually to convey clear insights.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Select and create appropriate chart types based on data characteristics. • Format and customize charts	The following tools and equipment are to be available: • Computers with spreadsheet software (e.g., Microsoft Excel, Google Sheets, LibreOffice Calc). • Sample datasets for creating charts and tables.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			tasks where students create various types of charts (e.g., bar, line, pie, scatter) and tables for specific datasets. **Interactive Tutorials:** Provide multimedia guides to explore advanced features like dynamic charts and pivot tables. **Group Projects:** Encourage collaborative work on projects requiring visual data presentations, such as reports or dashboards. **Problem-Solving Activities:** Present datasets where students must choose appropriate visualizations to	• Customize charts and tables using formatting tools (e.g., colors, styles, labels).		and tables for effective communication. • Utilize tools like pivot tables for summarizing and analyzing large datasets. **Principles:** The student should explain the principle of: • Data visualization design, including clarity, simplicity, and accuracy. • Choosing the right chart types for specific data comparisons (e.g., trends, proportions, distributions). • Aligning visualizations with the audience's needs and comprehension level.	• Templates for common visualizations, such as sales dashboards or performance reports. • User manuals and video tutorials on data visualization. • Case studies and examples of professional visualizations. • Practice exercises to improve chart selection and formatting skills.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			represent trends or comparisons. **Case Studies:** Analyze examples of effective and ineffective visualizations, emphasizing design principles and best practices.			**Theories:** The trainee should explain: The trainee should explain: • Visual perception and how they influence data interpretation. • The role of visual hierarchies in guiding the viewer's attention. **Circumstantial knowledge** Detailed knowledge about: • Common mistakes in data visualization and how to avoid them. • Best practices for using colors, fonts, and spacing in charts and tables.		
2 Basics of computer programing	2.1 Designing of computer programs	a) Demonstrating application tools and	**Demonstration:** Show how to set up programming environments	The student should be able to: • Install and configure	Programming tools and environments are correctly set up and operational.	**Detailed knowledge of:** **Methods:** The trainee should explain how	The following tools and equipment are to be available: • Computers with pre-	86

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
		environment used in programming	(e.g., IDEs, code editors, compilers) **Hands-On Practice:** Assign tasks where students install, configure, and use programming tools **Interactive Tutorials:** Provide guided exercises to explore features of common programming tools and environments.	programming environments correctly. • Navigate and utilize IDEs or code editors efficiently. • Use debugging tools and techniques effectively.		to: • Set up and configure development environments and tools. • Utilize debugging tools to identify and resolve coding errors. **Principles:** The student should explain the principle of: • Efficient programming workflows using tools and environments. • Version control and collaboration in software development. • Code debugging and testing processes. **Theories:** The trainee should explain: The trainee should explain: • Software	installed IDEs and programming tools (e.g., Visual Studio Code, PyCharm, IntelliJ). • Access to programming language compilers and interpreters • User manuals and video tutorials for programming tools. • Sample code repositories for demonstration and practice. • Interactive guides for setting up and troubleshooting programming environments.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						development lifecycle and the role of tools at different stages. • The importance of integrated development environments (IDEs) in streamlining coding tasks. **Circumstantial knowledge** Detailed knowledge about: • Features and advantages of popular programming tools and environments. • Common issues in tool setup and how to resolve them.		
		b) Constructing pseudo-code and flow charts	**Demonstration:** Illustrate the creation of pseudo-code and flowcharts for simple programs using examples.	The student should be able to: • Understand problem statements and decompose them into	Pseudo-code and flow that provides a clear and logical representation of program are constructed.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Break down a problem into logical steps for	The following tools and equipment are to be available: • Computers with flowchart design tools (e.g., Lucid chart, Draw.io, Visio) or graph	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			Hands-On Practice: Assign students exercises to develop pseudo-code and corresponding flowcharts for various problem-solving scenarios. **Interactive Tutorials:** Use software tools (e.g., Lucidchart, Draw.io, or hand-drawn sketches) to create and visualize flowcharts. **Problem-Solving Activities:** Provide tasks where students translate problem statements into pseudo-code and design flowcharts.	logical steps. • Write clear and logically correct pseudo-code. • Design accurate and readable flowcharts representing program logic. • Use appropriate symbols and conventions for flowcharts.		pseudo-code and flowchart construction. • Use standard flowchart symbols and structures. • Ensure consistency between pseudo-code and flowcharts. **Principles:** The student should explain the principle of: • Problem-solving techniques in programming. • Logical sequencing and decision-making in program design **Theories:** The trainee should explain: The trainee should explain: • The role of pseudo-code and flowcharts in planning and	paper for manual design. • Reference materials on flowchart symbols and pseudo-code conventions. • Sample problems for practice in pseudo-code and flowchart creation. • Tutorials and guides on logical problem-solving and design. • Examples of well-constructed pseudo-code and flowcharts. • Practice exercises for debugging pseudo-code and improving flowcharts.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						debugging programs. • How clear visual representation aids in understanding and communication of program logic. **Circumstantial knowledge** Detailed knowledge about: • Common errors in pseudo-code and flowchart design and how to avoid them. • Best practices for aligning pseudo-code with standard programming syntax. • Tools and resources available for creating and sharing flowcharts.		
		c) Implementing basic computer	**Demonstration:** Show how to write, compile,	The student should be able to: • Write clear,	Basic computer programs that run without errors and	**Detailed knowledge of:** **Methods:** The trainee	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
		programming	execute, and debug simple programs using programming tools. **Hands-On Practice:** Assign students to write and execute basic programs that include input/output, loops, conditionals, and basic calculations. **Interactive Tutorials:** Provide guided exercises to explore syntax, structure, and execution in a programming language **Problem-Solving Activities:** Present scenarios requiring students to design and implement solutions in code	logically structured code for basic problems. • Use proper syntax and programming conventions • Debug and resolve errors during program execution.	produce expected results.	should explain how to: • Write code using correct syntax and logic. • Test and debug programs to identify and fix errors. **Principles:** The student should explain the principle of: • Logical flow in programming and problem-solving. • Data types, variables, and control structures. **Theories:** The trainee should explain: The trainee should explain: • How algorithms translate into program logic. • The role of control structures in decision-making and	• Computers with installed programming environments (e.g., Python, Java, C++). • Debugging tools and text editors (e.g., Visual Studio Code, PyCharm). • Sample programs for editing, debugging, and testing. • Programming language guides and reference materials. • Case studies of beginner programs and real-world examples. • Exercises for practicing coding and debugging.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						iteration. **Circumstantial knowledge** Detailed knowledge about: • Common programming errors and debugging techniques. • Best practices in coding, including commenting and naming conventions. • Using built-in libraries and tools to simplify coding tasks		
	2.2 Implementing control structures in programming	a) Implementing condition structures	**Demonstration:** Show how to use conditional statements (e.g., if, if-else, switch-case) in a programming language. **Hands-On Practice:** Assign tasks to students to write programs incorporating	The student should be able to: • Write syntactically correct conditional statements. • Apply nesting and logical operators in condition structures. • Debug and	Programs that execute correctly with all possible condition outcomes handled are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use conditional statements (if, if-else, switch-case) effectively. • Combine logical operators (e.g., AND, OR, NOT) for complex conditions.	The following tools and equipment are to be available: • Computers with programming environments installed • Debugging tools and software for analyzing program execution. • Sample programs for editing, testing, and debugging	108

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			condition structures to solve specific problems. **Interactive Tutorials:** Provide guided exercises to understand the syntax, logic, and nesting of condition structures. **Problem-Solving Activities:** Present scenarios requiring decision-making logic and implement solutions using condition structures.	test programs involving condition structures.		**Principles:** The student should explain the principle of: • Logical decision-making in programming. • Efficiency and clarity in designing condition structures. • Testing all possible outcomes to ensure robust code. **Theories:** The trainee should explain: The trainee should explain: • The role of branching in program execution. • How condition structures affect program flow and logic **Circumstantial knowledge** Detailed knowledge	conditional statements. • Programming language reference guides and syntax manuals. • Case studies showcasing real-world use of condition structures. • Exercises for constructing condition structures.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						about: • Common errors in conditional logic and debugging strategies.		
		b) Implementing loop structures	**Demonstration:** Show how to write and execute basic loop structures (for, while, do-while) in a programming language. **Hands-On Practice:** Assign tasks requiring students to solve problems using different loops. **Interactive Tutorials:** Provide guided examples to explore nested loops, infinite loops, and loop termination conditions. **Problem-Solving Activities:** Challenge students with	The student should be able to: • Write syntactically correct loop structures. • Identify and apply the appropriate type of loop for specific problems. • Use break and continue statements effectively to control loop execution.	Programs that execute loops correctly with the desired looping behaviour are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use different types of loops (for, while, do-while) effectively. • Implement nested loops for solving complex problems. • Debug loop-related issues such as infinite loops or incorrect iteration. **Principles:** The student should explain the principle of: • Iteration in programming for repetitive tasks.	The following tools and equipment are to be available: • Computers with programming environments installed • Debugging tools and software for analyzing program execution. • Sample problems and datasets for practicing loop structures. • Programming language reference guides and syntax manuals. • Case studies demonstrating the use of loops in solving real-world problems. • Exercises to practice nested loops, loop optimization, and debugging.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			scenarios that require iteration, such as summing numbers or creating patterns. **Case Studies:** Analyze sample programs to identify and correct issues in loop logic and structure.			• Efficiency in designing loops to minimize unnecessary iterations. • Logical structure of loop conditions and initialization. **Theories:** The trainee should explain: The trainee should explain: • The role of loops in program automation and efficiency. • Differences between pre-test (while, for) and post-test (do-while) loops. **Circumstantial knowledge** Detailed knowledge about: • Common errors in loop implementation and debugging strategies. • Practical use		

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						cases of loops in data processing, gaming, and algorithm design.		
		c) Implementing nested structures	**Demonstration:** Show how to use nested control structures (nested loops and nested conditionals) to solve complex problems. **Hands-On Practice:** Assign exercises where students implement nested structures for tasks such as creating patterns, multi-level decision-making, or multidimensional data processing. **Interactive Tutorials:** Provide guided examples and simulations of nested loops and conditionals in real-world	The student should be able to: • Write syntactically correct nested control structures. • Debug and test nested loops and conditionals for logical accuracy. • Identify when and where nested structures are appropriate.	Programs that handle complex scenarios accurately using nested structures are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Construct and debug nested loops and conditionals. • Use appropriate nesting levels to handle multi-dimensional or hierarchical data. **Principles:** The student should explain the principle of: • Logical flow and sequencing in nested structures. • Efficiency in designing nested loops and conditionals to minimize redundancy. • Error handling and debugging	The following tools and equipment are to be available: • Computers with programming environments installed • Debugging tools for analyzing program logic and execution flow. • Sample problems and datasets for practicing nested structures. • Tutorials and guides for nested loops and conditionals. • Case studies demonstrating real-world use of nested structures. • Exercises focusing on optimizing nested structures for efficiency.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			scenarios. **Problem-Solving Activities:** Present programming challenges requiring the use of nested structures, such as generating multiplication tables or matrix operations.			nested structures. **Theories:** The trainee should explain: The trainee should explain: • How nested structures contribute to solving complex programming problems. • The role of logical hierarchies in program design and decision-making. • The importance of testing edge cases in nested conditions. **Circumstantial knowledge** Detailed knowledge about: • Common pitfalls in implementing nested structures and how to resolve them. • Best practices for		

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						structuring nested loops and conditionals for readability.		
	2.3 Implementing functions	a) Implementing user-defined functions and function calls in a program	**Demonstration:** Show how to create, define, and call user-defined functions in a programming language. **Hands-On Practice:** Assign exercises requiring students to implement user-defined functions to solve specific problems. **Interactive Tutorials:** Provide guided examples to teach function parameters, return values, and recursive function calls. **Problem-Solving Activities:** Challenge students with	The student should be able to: • Define functions with proper syntax and structure. • Pass parameters to functions and use return values effectively. • Debug and test functions for correctness and efficiency. • Write modular and reusable code.	User-defined functions that operate correctly and produce the expected results are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Create and call user-defined functions in a program. • Use parameters and return values to transfer data between functions. **Principles:** The student should explain the principle of: • The role of functions in simplifying program design. • The impact of modular programming on scalability and debugging. **Theories:** The trainee should explain:	The following tools and equipment are to be available: • Computers with programming environments installed. • Debugging tools to analyze program flow and function execution. • Sample problems for writing and testing user-defined functions. • Tutorials and guides for function implementation and debugging. • Case studies demonstrating modular programming techniques. • Exercises focusing on advanced function features, such as recursion and default parameters.	67

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			scenarios where modular programming is essential, such as calculating factorials or simulating processes. **Case Studies:** Analyze sample programs to highlight the importance of reusable and well-defined functions.			The trainee should explain: • How nested structures contribute to solving complex programming problems. • The role of logical hierarchies in program design and decision-making. • The importance of testing edge cases in nested conditions. **Circumstantial knowledge** Detailed knowledge about: • Common errors in function definition and usage and how to resolve them. • Real-world applications of user-defined functions in software development.		
		b) Implementing recursive	**Demonstration:** Show how recursive	The student should be able to: • Define recursive	Recursive functions execute correctly and return expected	**Detailed knowledge of:** **Methods:** The trainee	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
		function in a program	functions work using visual aids (e.g., diagrams or call stack visualization). **Hands-On Practice:** Assign tasks to write and execute recursive functions for problems like factorial calculation and Fibonacci sequence **Interactive Tutorials:** Provide guided examples demonstrating the principles of base cases and recursive calls. **Problem-Solving Activities:** Challenge students to identify and implement recursive solutions for problems suited to	functions with proper base and recursive cases. • Debug and trace recursive calls using visual tools or debugging software. • Recognize when recursion is a suitable approach for problem-solving.	results that are consistent with program specifications.	should explain how to: • Identify problems suitable for recursion. • Write recursive functions with proper base cases and recursive logic. **Principles:** The student should explain the principle of: • Recursive problem-solving, including base case and recursive case design. • Trade-offs between recursion and iteration in terms of performance and clarity. **Theories:** The trainee should explain: The trainee should explain: • How recursion simplifies solving problems with repetitive or nested substructures.	• Computers with programming environments. • Debugging tools for tracing recursive calls (e.g., visualizers or IDE-integrated tools). • Sample problems for implementing and testing recursive functions. • Tutorials on recursion fundamentals and techniques. • Case studies demonstrating recursive solutions for practical problems. • Exercises comparing recursion and iteration for various problem types.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			recursion.			• The concept of divide-and-conquer and its implementation via recursion. **Circumstantial knowledge** Detailed knowledge about: • Common pitfalls in recursive programming and how to prevent them. • Real-world applications of recursion.		
	2.4 Implementing arrays in a program	a) Demonstrating declaration and initialization of single dimensional array	**Demonstration:** Show how to declare and initialize single-dimensional arrays in a programming language. **Hands-On Practice:** Assign tasks requiring students to create arrays, populate them with data, and perform	The student should be able to: • Declare and initialize arrays using correct syntax. • Access, modify, and iterate over elements in the array. • Debug and resolve errors related to array	Arrays are correctly declared and initialized with valid data to meet specified requirements.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Declare and initialize arrays with fixed sizes and dynamic data. • Access and manipulate array elements using indices. • Iterate through arrays using	The following tools and equipment are to be available: • Computers with programming environments installed. • Debugging tools to visualize and analyze array usage in programs. • Sample problems for practicing array operations. • Tutorials on array syntax and	63

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			operations like accessing or modifying elements. **Interactive Tutorials:** Use guided exercises to teach the syntax and operations of arrays, including indexing and iteration. **Problem-Solving Activities:** Present challenges where students must use single-dimensional arrays to store and process data (e.g., calculating averages or finding maximum values). **Group Projects:** Facilitate collaborative tasks where students design programs utilizing single-	boundaries or initialization • Use arrays to solve problems efficiently.		loops for various operations. **Principles:** The student should explain the principle of: • Memory allocation for arrays and its impact on program efficiency. • Logical indexing and boundary conditions in array operations. • Data organization using arrays for sequential storage. **Theories:** The trainee should explain: The trainee should explain: • How arrays simplify data management and retrieval in programming. • The relationship between arrays	operations in different languages. • Case studies demonstrating the use of arrays in real-world applications. • Exercises focusing on array declaration, initialization, and iteration.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			dimensional arrays for specific applications. **Case Studies:** Analyze real-world scenarios where arrays are used effectively, discussing their advantages and limitations.			and iterative processing. **Circumstantial knowledge** Detailed knowledge about: • Common errors in array declaration and how to resolve them (e.g., out-of-bounds errors). • Real-world use cases of arrays in data storage and processing.		
		b) Demonstrating accessing and printing of elements from an array	**Demonstration:** Show how to access and print elements of single-dimensional arrays using indices in a programming language. **Hands-On Practice:** Assign exercises where students retrieve and display specific elements	The student should be able to: • Access array elements correctly using indices. • Traverse arrays using loops (e.g., for and while loops).	Array elements are accessed and displayed (printed) correctly as per the task requirements.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use indices to retrieve specific elements from an array. • Iterate through arrays to display all elements or subsets. • Debug errors related to incorrect index access.	The following tools and equipment are to be available: • Computers with programming environments. • Debugging tools for analyzing array operations. • Sample problems for accessing and printing array data. • Tutorials and guides on array traversal and output formatting. • Case studies	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
			from arrays, iterate through arrays, and print all elements. **Interactive Tutorials:** Provide guided examples demonstrating array traversal using loops and conditional checks. **Problem-Solving Activities:** Present tasks requiring data extraction and printing from arrays, such as finding specific elements or calculating their properties (e.g., sum or average).			**Principles:** The student should explain the principle of: • Indexing in arrays and its role in element access. • Loop structures for traversing and printing array elements. • Error prevention and handling in array operations. **Theories:** The trainee should explain: The trainee should explain: • How array traversal facilitates data manipulation and presentation. • The importance of bounds checking in accessing array elements. **Circumstantial**	showcasing effective array data presentation. • Exercises focusing on error handling and efficient array traversal.	

Module Title (Main Competence)	Unit Title (Specific Competencies)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product / Services Assessment	Underpinning Knowledge		
						knowledge Detailed knowledge about: • Common pitfalls in accessing array elements (e.g., off-by-one errors) and their resolutions. • Real-world applications of array data retrieval, such as in data analysis and reporting.		

Form Two

Table 4*: Detailed Contents for Form Two*

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
1. Designing and hosting websites	1.1 Creating web pages using HTML	(a) Creating basic webpage using HTML	**Brainstorm**: Introduce HTML basics and guide students to define syntax, and structure of HTML. **Practical Work**: Guide students to create a basic webpage using common tags (e.g., <html>, <head>, <body>, <h1>). **Group Activity**: Students create a simple webpage using the basic HTML tags and present.	• Identify required tags for webpage structure. • Set up an HTML document (HTML, head, body). • Add headings and paragraphs. • Save and preview the webpage in a browser. • Troubleshoot errors and correct them.	Web pages that load and display correctly in a browser are created.	**Detailed knowledge of:** **Method used:** The student should explain how to • create a basic webpage using HTML. **Principles:** The student should explain the HTML structure and its components. **Theories:** The student should explain: • HTML tag types and usage. **Circumstantial knowledge:** Detailed knowledge about: • Common HTML tags	The following tools and equipment are to be available: • Computer with a code editor (e.g., VS Code, Notepad++). • Web browser (e.g., Chrome, Firefox). • Projector for demos.	54
		(b) Implementing HTML lists	**Lecture**: Explain types of lists in HTML (ordered, unordered, description).	• Identify list types (ordered and unordered). • Add list items using .	The web page containing the HTML lists is correctly implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • implement lists in	The following tools and equipment are to be available: • Computer with a code editor (e.g., VS Code,	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Demonstration: Show examples of each list type. **Hands-on Exercise**: Students practice creating lists.	• Nest lists where applicable. • Save and preview the lists in a browser. • Troubleshoot errors and correct them.		HTML. **Principles:** The student should explain the principle of when to use each list type **Theories:** The student should explain: • Types of lists • Nested lists **Circumstantial knowledge:** Detailed knowledge about: • Lists in web pages	Notepad++). • Web browser (e.g., Chrome, Firefox). • Projector for demos.	
		(c) Creating HTML links	**Lecture:** Introduce hyperlinking using the <a> tag. **Practical Work:** Students create pages with internal, external, and email links. **Discussion**: Explore linking best practices and accessibility.	• Identify types of links (internal, external, email). • Add hyperlinks using the <a> tag. • Set attributes (href, target, title). • Test links in the browser. • Troubleshoot and correct broken links.	Web pages containing correctly functioning internal and external links are created.	**Detailed knowledge of:** **Method used:** The student should explain how to: • Create HTML links. **Principles:** The student should explain the concept of hyperlinking. **Theories:** The student should explain: • Types of links (e.g. internal, external, email). • The link tags **Circumstantial knowledge:**	The following tools and equipment are to be available: • Computer with a code editor (e.g., VS Code, Notepad++). • Web browser (e.g., Chrome, Firefox). • Projector for demos.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						Detailed knowledge about: • Attributes affecting link behavior.		
		(d) Creating data tables and forms using HTML	**Lecture**: Introduce table and form tags (<table>, <tr>, <td>, <form>, <input>, etc.). **Demonstration**: Show examples of tables and forms. **Practice**: Students create a form and a table.	• Set up table structure using <table>, <tr>, <td>. • Add form elements (<input>, <select>, <textarea>). • Set form attributes (action, method). • Save and preview the table and form in a browser. • Troubleshoot display issues.	Tables and forms that use HTML to display data correctly are created.	**Detailed knowledge of:** **Method used:** The student should explain how to: • Create HTML Tables • Create HTML forms **Principles:** The student should explain the principle of why and when to use HTML tables and forms **Theories:** The student should explain: • The concepts of HTML tables • The concepts of HTML forms. • Structure of tables and form elements. **Circumstantial knowledge:** Detailed knowledge about:	The following tools and equipment are to be available: • Computer with a code editor (e.g., VS Code, Notepad++). • Web browser (e.g., Chrome, Firefox). • Projector for demos.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						• form attributes (e.g., action, method).		
	1.2 Formatting web pages with Cascading Style Sheet (CSS)	(a) Managing colours in CSS	**Brainstorm**: Guide students to define colors properties in CSS (e.g. color, background-color). **Demonstration:** Show examples of setting text and background colours. **Practice**: Students apply colours in web pages.	• Identify colour properties in CSS. • Apply text colour using color. • Apply background colour using background-color. • Use different colour formats (hex, RGB, HSL). • Test colour application in the browser.	Colours in CSS are appropriately managed by formatting web pages according to design requirements.	**Detailed knowledge of:** **Method used:** The student should explain how to: • Apply CSS properties to manage colors, typography, and layout **Principles:** The student should explain the principle of: • Colour models (RGB, hex, HSL). **Theories:** The student should explain: • CSS color propertiesand inheritance • Color formats. **Circumstantial knowledge:** Detailed knowledge about: • color contrast for accessibility.	The following tools and equipment are to be available: • Code editor and browser. • Colour model charts and online CSS colour tools.	36
		(b) Implementing typography	**Brainstorm**: Guide students about typography	• Set font family using font-family.	Topography in CSS is correctly	**Detailed knowledge of:** **Method used:** The	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
		in CSS	properties (font-family, font-size, line-height, font-weight). **Practice**: Students apply typography properties in sample pages.	• Adjust font size using font-size. • Control line spacing using line-height. • Use font-weight for emphasis. • Preview and adjust the typography.	implemented according to design requirements.	student should explain how to: • implement typography in CSS. **Principles:** The student should explain the principle of: • font-family stacks and fallback fonts. **Theories:** The student should explain: • CSS typography properties. **Circumstantial knowledge:** Detailed knowledge about: • responsive typography.	• Code editor and browser. • Colour model charts and online CSS colour tools.	
		(c) Implementing the box model in CSS	**Brainstorm:** Explain the CSS box model (margin, padding, border, width, height). **Demonstration:** Show how to style boxes using these properties. **Practice:** Students apply box model	• Set width and height of elements. • Apply padding, margin, and border. • Test and adjust spacing between elements. • Use box-	Web pages with a box model employing CSS are implemented correctly.	**Detailed knowledge of:** The student should explain how to: • implement the CSS box model. **Principles:** The student should explain the principle of: • spacing and box-sizing. **Theories:** The student should explain: • CSS box model	The following tools and equipment are to be available: • Code editor and browser.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			properties.	sizing property. • Preview and verify box model implementation.		structure **Circumstantial knowledge:** Detailed knowledge about: • border styles and effects.		
		(d) Implementing page layout techniques	**Instruction:** Introduce page layout techniques (CSS Grid, Flexbox, float, and positioning). **Demonstration:** Show examples of page layouts using these techniques. **Group Activity:** Students create layouts.	• Apply layout techniques using CSS Grid and Flexbox. • Use float and clear properties for layout. • Apply CSS positioning (relative, absolute, fixed). • Test page layout responsiveness.	Web pages displaying structured and responsive layouts that adapt to different screen sizes are implemented.	**Detailed knowledge of:** The student should explain how to: • implement page layout techniques in CSS. **Principles:** The student should explain the principle of: • responsive web design. **Theories:** The student should explain: • layout models (Grid, Flexbox) **Circumstantial knowledge:** Detailed knowledge about: • media queries	The following tools and equipment are to be available: • Code editor and browser. • Sample layout designs.	
		(e) Implementing graphics in webpages using CSS	**Brainstorm:** Explain CSS properties for graphics (background-image, background-size, border-radius, box-	• Apply background images using background-image. • Adjust	Web pages displaying properly styled and positioned graphics that	**Detailed knowledge of:** The student should explain how to: • implement graphics using CSS.	The following tools and equipment are to be available: • Code editor and browser. • Sample images	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			shadow). **Demonstration:** Show how to apply and style graphics in web pages. **Practice:** Students add graphics.	background size using background-size. • Add rounded corners using border-radius. • Add shadows using box-shadow. • Test graphics implementation	enhance visual appeal without affecting performance are implemented.	**Principles:** The student should explain the principle of: • layering of backgrounds and shadows. **Theories:** The student should explain: • Concepts of graphics • CSS properties for background and borders. **Circumstantial knowledge:** Detailed knowledge about: • performance optimization for graphics.	and webpages with graphics.	
	1.3 Implementing web scripting	(a) Implementing variables	**Brainstorm:** Introduce and guide students on variable declaration such as var, let, and const. **Demonstration**: Show examples of initializing and using variables. **Practice:** Students write and test simple	• Declare variables using var, let, and const. • Initialize variables with different data types. • Test variable usage in scripts.	Declaration and initialization of variables of different data types are implemented	**Detailed knowledge of:** The student should explain how to: • declare variables • use variables in JavaScript. **Principles:** The student should explain the principle of: • data types and scope. **Theories:** The student should explain:	The following tools and equipment are to be available: • Code editor and browser. • Sample scripts for variable demonstration	81

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			scripts.			• Concepts of variables and data types • Variable declaration and assignment rules **Circumstantial knowledge:** Detailed knowledge about: • Data types		
		(f) Implementing selection, loop, and array	**Brainstorm:** Guide students to identify cases that need conditional statements (if, else, switch) and loops (for, while, do-while). **Practice:** Students implement conditions and loops in sample tasks.	• Apply conditional statements (if-else, switch). • Use loops (for, while, do-while). • Declare and manipulate arrays.	Selection, loop and array are correctly implemented in scripts to perform the required tasks.	**Detailed knowledge of:** **Method used:** The student should explain how to: • implement conditions, loops, and arrays. **Principles:** The student should explain the principle of: • control flow and iteration. **Theories:** The student should explain: • The concepts of flow control and arrays • use of arrays in storing and accessing data. **Circumstantial**	The following tools and equipment are to be available: • Code editor and browser. • Sample problems requiring loops and arrays.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						knowledge: Detailed knowledge about: • array methods		
		(g) Implementing basics of functions	**Instruction:** Introduce functions and their syntax (function keyword, parameters, return values). **Demonstration:** Show examples of function definition and invocation. **Practice:** Students create functions.	• Define functions with parameters and return values. • Invoke functions with different arguments. • Test and debug functions	The basics of functions are correctly implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • define functions • implement function. **Principles:** The student should explain the principle of: • function parameters and return values. **Theories:** The student should explain: • The meaning of functions, arguments, and parameters • Function declaration vs. function expressions. **Circumstantial knowledge:** Detailed knowledge about: • modular	The following tools and equipment are to be available: • Code editor and browser. • Sample tasks requiring function implementation	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						programming		
		(h) Implementing event handling	**Brainstorm:** Guide students to identify different event handling scenarios (e.g., onclick, onmouseover). **Demonstration:** Show how to attach event handlers to elements. **Practice:** Students create interactive web pages using events.	• Attach event handlers to HTML elements. • Use inline and external event handling methods. • Test event-driven scripts for interactivity.	Web pages responding correctly to user actions are implemented according to requirements.	**Detailed knowledge of:** **Method used:** The student should explain how to: • handle events in JavaScript. **Principles:** The student should explain the principle of: • event propagation model (bubbling and capturing). **Theories:** The student should explain: • Inline vs. external event handling. **Circumstantial knowledge:** Detailed knowledge about: • common events in web programming	The following tools and equipment are to be available: • Code editor and browser. • Sample interactive web pages.	
	1.4 Hosting and publishing websites	(a) Setting up web servers	**Brainstorm:** Introduce and guide students on the types of web servers (e.g., Apache, Nginx).	• Install web server software (e.g., Apache, Nginx). • Configure	A fully functional web server that can serve static and dynamic content is set	**Detailed knowledge of:** **Method used:** The student should explain how to: • set up a web	The following tools and equipment are to be available: • Computer with internet access. • Web server	9

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Demonstration: Guide students through installing and configuring a local web server. **Practice:** Students set up a basic server.	web server settings (ports, virtual hosts).	up.	server **Principles:** The student should explain the principle of: • client-server architecture. **Theories:** The student should explain: • Different types of web servers and their use cases **Circumstantial knowledge:** Detailed knowledge about: • Security considerations in server setup.	software (e.g., Apache, Nginx). • Sample website files for testing.	
		(b) Deploying website	**Instruction:** Explain deployment steps (uploading files, configuring domain names). **Demonstration:** Guide students through deploying a website on a live server. **Practice:** Students deploy their own websites.	• Upload website files to a web server. • Configure domain and DNS settings. • Test the deployed website for accessibility.	An accessible website hosted on a server, with correct DNS and URL configurations is deployed.	**Detailed knowledge of:** **Method used:** The student should explain how to: • website deployment process. **Principles:** The student should explain the principle of: • DNS, domains, and hosting. **Theories:** The student should explain:	The following tools and equipment are to be available: • Computer with internet access. • Access to hosting platforms (e.g., free hosting services or local servers). • Domain configuration tools.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						• the concepts of name resolutions • DNS translation process of domain names to IP addresses. **Circumstantial knowledge:** Detailed knowledge about: • FTP, SFTP, and hosting platforms.		
	1.5 Designing websites using content management systems	(a) Setting up blog sites	**Instruction:** Introduce CMS platforms (e.g., WordPress, Joomla). **Demonstration:** Show how to install and configure a CMS. **Practice:** Students set up personal blog sites using a CMS.	• Install a CMS (e.g., WordPress). • Configure general settings (title, theme, users). • Create initial web pages.	A functional blog site with basic pages is properly designed.	**Detailed knowledge of:** **Method used:** The student should explain how to: • set up a CMS. **Principles:** The student should explain the principle of: • architecture of a CMS. **Theories:** The student should explain: • the concepts of CMS • Difference between static websites and CMS-based websites. **Circumstantial**	The following tools and equipment are to be available: • Local computer or hosting platforms with CMS installation support. • Internet connection. • Sample content for blogs.	54

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						knowledge: Detailed knowledge about: • Managing CMS settings.		
		(b) Using plugins and widgets in blog sites	**Brainstorm:** Guide students discussion on the purpose of plugins and widgets. **Demonstration:** Show how to install and configure plugins (e.g., SEO, contact forms). **Practice:** Students install and use plugins.	• Install essential plugins (SEO, security). • Add and configure widgets (menus, social media links).	Appropriate plugins and widgets are used in blog sites.	**Detailed knowledge of:** **Method used:** The student should explain how to: • manage plugins and widgets. **Principles:** The student should explain the principle of: • plugin architecture and functionality **Theories:** The student should explain: • the concepts of plugins and widgets • Choosing the right plugins and widgets according to purposes • Security implications of using third-party plugins. **Circumstantial knowledge:** Detailed knowledge	The following tools and equipment are to be available: • Local computer or hosting platforms with CMS installation support. • CMS platform (e.g., WordPress). • Access to plugin repositories. • Internet connection for downloading plugins and widgets.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						about: • Compatibility issues between plugins.		
2. Developing Database Systems	2.1 Implementing database modelling and relationships	(a) Designing relations	**Brainstorm:** Guide students on the concept of database relations and normalization. **Practical:** Guide students to design relations based on a real-world scenario. **Discussion:** Group activity to evaluate and refine designs.	• Identify entities • Identify attributes and keys. • Apply normalization rules to organize data into relations. • Create relational tables.	Database relations are correctly designed.	**Detailed knowledge of:** **Method used:** The student should explain how to: • process of designing relations. **Principles:** The student should explain the principle of: • Normalization and data redundancy **Theories:** The student should explain: • The concepts of entities, attributes, relationships • Types of relationships (1:1, 1:M, M:N). **Circumstantial knowledge:** Detailed knowledge about: • Data integrity and dependencies.	The following tools and equipment are to be available: • Database design tools (e.g., MySQL Workbench, Lucidchart). • Sample case studies or scenarios.	54
		(b) Designing entity	**Brainstorm:** Introduce and guide	• Identify entities and	An entity relationship	**Detailed knowledge of:**	The following tools and equipment are to	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
		relationships	students on the concept of entities and relationships. **Practical:** Guide students in identifying entities and defining relationships between them. **Case Study:** Analyze and improve existing entity designs.	their attributes. • Define relationships between entities (e.g., foreign keys). • Validate the relationships.	that effectively represents the problem domain and shows logical connections between entities is designed.	**Method used:** The student should explain how to: • design entity relationships. **Principles:** The student should explain the principle of: • primary keys, foreign keys, and cardinality. **Theories:** The student should explain: • The concepts and importance of relational/referential integrity **Circumstantial knowledge:** Detailed knowledge about: • Impact of poorly defined relationships.	be available: • Computer installed with tools like Microsoft Access or MySQL Workbench. • Example database schemas. • Problem statements for practice scenarios.	
		(c) Designing entity relationship diagrams	**Brainstorm**: Explain ER diagrams and their components (entities, attributes, relationships). **Practical Work**: Guide students in creating ER	• Create ER diagrams based on provided scenarios. • Use standard notations (e.g. Chen, Crow's Foot).	A complete and accurate ER diagram illustrating entities, relationships, and attributes in a logical structure is	**Detailed knowledge of:** **Method used:** The student should explain how to: • create and interpret ER diagrams. **Principles:** The	The following tools and equipment are to be available: • Computer installed with diagramming tools (e.g., Lucidchart, Draw.io, Visio).	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			diagrams using a tool. **Activity**: Peer review of ER diagrams for improvement.	• Include attributes, keys, and relationships.	designed.	student should explain the principle of: • diagrammatic representation of databases. **Theories:** The student should explain: • Types of notations (e.g., Chen, Crow's Foot). **Circumstantial knowledge:** Detailed knowledge about: • Common pitfalls in diagram design.	• Case studies or problem scenarios. • Templates for standard notations.	
	2.2 Implementing physical database structure with Structured Query Language	(a) Implementing Data Definition Language (DDL)	**Instruction**: Explain DDL commands (CREATE, ALTER, DROP). **Demonstration:** Show how to create tables **Practice:** Students write SQL scripts to define database structures.	• Write DDL scripts for creating database tables • Write DDL commands to modify database tables. • Write DDL commands to delete database tables.	Database schema with well-defined tables, columns, and constraints implemented.	**Knowledge evidence:** **Detailed knowledge of:** **Method used:** The student should explain how to: • create database tables using DDL queries • alter database tables using DDL queries • drop database objects using DDL queries. **Principles:** The student should explain	The following tools and equipment are to be available: • Computer installed with SQL tools (e.g., MySQL, PostgreSQL, SQL Server). • Sample datasets and schema design tasks. • Reference guides on SQL syntax.	72

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						the principles of: • data types, primary keys, and indexes. **Theories:** The student should explain: • The concepts of DDL • Schema design best practices. **Circumstantial knowledge:** Detailed knowledge about: • Impact of poorly designed schemas.		
		(b) Implementing Data Manipulation Language (DML)	**Brainstorm**: Guide students on to define and use DML commands (such as INSERT, UPDATE, DELETE). **Practical:** Guide students to manipulate data in tables. **Activity:** Students perform CRUD operations on sample databases.	• Write DML commands to insert, update, and delete data into the database table. • Write DML commands to update in the database table. • Write DML commands to delete data from the	The accurate and consistent data manipulation is implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • insert data into the database table using DML queries • update data in a table using DML queries • delete data from a database table using DML **Principles:** The	The following tools and equipment are to be available: • Computer installed with database access tools (e.g., phpMyAdmin, pgAdmin). • Predefined tables for manipulation tasks. • Sample SQL scripts for practice.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
				database table. • Handle data consistency during operations. • Validate results.		student should explain the principles of: • ACID properties in data manipulation. **Theories:** The student should explain: • The basic concepts of DML queries. **Circumstantial knowledge:** Detailed knowledge about: • Error handling during data manipulation		
		(c) Implementing relational constraints	**Brainstorm**: Guide students to explore relational database constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK). **Demonstration:** Show the creation of primary keys, foreign keys using SQL commands. **Practice:** Students define	• Define primary and foreign keys. • Add UNIQUE and CHECK constraints. • Test constraints with sample data.	A database structure with constraints that ensures data integrity and enforces business rules is implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • write SQL commands to show the primary key, foreign key, unique key • role of relational constraints. **Principles:** The student should explain the principles of:	The following tools and equipment are to be available: • Computer • SQL databases with predefined scenarios. • Problem tasks requiring integrity constraints.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			relational constraints in tables.			• ensuring data integrity with constraints. **Theories:** The student should explain: • Types and purpose of constraints. **Circumstantial knowledge:** Detailed knowledge about: • Common errors when defining constraints.		
		(d) Implementing Data Query Language (DQL)	**Discussion:** Guide students to come up with different scenarios for using the SELECT queries. **Demonstration:** Illustrate using basic DQL commands and advanced querying techniques (e.g., JOIN, GROUP BY, subqueries). **Practice:** Students write queries to extract all datafrom the	• Write SELECT queries to extract required data from a database table. • Use filtering, grouping, and sorting.	Queries performing correct retrieval of data based on user needs are implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • retrieve data from database tables • filter or sort data retrieved from the database **Principles:** The student should explain the principles of: • Efficient data retrieval. **Theories:** The student should explain: • Different DQL	The following tools and equipment are to be available: • Computer with a working database that has data. • Query practice scenarios (e.g., analytics tasks).	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			database or some of the data based on different criteria.			syntaxes (such as SELECT) • JOINs, subqueries, and aggregate functions **Circumstantial knowledge:** Detailed knowledge about: • Query optimization and indexing.		
		(e) Implementing Transaction Control Language (TCL)	**Discussion:** Guide students on TCL commands (such as COMMIT and ROLLBACK). **Demonstration:** Illustrate transaction handling in SQL. **Practice:** Students manage transactions in a multi-step task.	• Use the COMMIT command to manage transactions. • Implement ROLLBACK on errors.	A database that maintains consistency after transaction execution and handles errors correctly is implemented.	**Detailed knowledge of:** **Method used:** The student should explain how to: • commit data into the database tables • roll back data if an error happens before successful saving of the data in the database **Principles:** The student should explain the principles of: • ACID properties in transactions. **Theories:** The student should explain: • Importance of	The following tools and equipment are to be available: • Computer with a working database that has data. • Scenarios requiring transaction management.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						transaction management. **Circumstantial knowledge:** Detailed knowledge about: • Resolving deadlocks and handling transaction failures.		
3. Developing database-driven web applications	• Creating PHP programs	(a) Applying basic PHP syntax	**Brainstorm:** Introduce and guide students on the PHP syntax (e.g., tags, statements, variables). **Demonstration:** Show how to embed PHP into HTML. **Practice:** Students write PHP scripts to output text and variables.	• Write basic PHP scripts. • Use PHP tags appropriately • Declare and initialize variables correctly.	Simple programs demonstrating the correct use of PHP syntax	**Detailed knowledge of:** **Method used:** The student should explain how: • PHP processes scripts on the server. **Principles:** The student should explain the principles of: • separating PHP from HTML for maintainable code. **Theories:** The student should explain: • the PHP concepts and program syntax • server-side scripting lifecycle, including request-	The following tools and equipment are to be available: • Computer with PHP development environment (e.g., XAMPP, WAMP). • Text editors/IDEs (e.g., VS Code, PHPStorm).	108

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						response cycles. **Circumstantial knowledge:** Detailed knowledge about: • Differences between server-side and client-side programming and their integration in web applications.		
		(b) Implementing conditional statements, loops and arrays	**Brainstorm**: Explain and guide students on PHP conditional statements (if-else, switch), loops (for, while), and arrays. **Demonstration:** Illustrate examples of control structures and arrays. **Practice:** Students write scripts with decision-making logic and iteration.	• Write scripts using conditional logic. • Create loops for repetitive tasks. • Use arrays to store and access multi-dimensional data.	Programs performing decision-making (e.g., grading system), loops (e.g., summing numbers), and array manipulation.	**Detailed knowledge of:** **Method used:** The student should explain how • PHP handles logic, iteration, and data storage in memory. **Principles:** The student should explain the principles of: • control structures for decision-making and data iteration. **Theories:** The student should explain: • algorithmic flow control (decision trees, loop	The following tools and equipment are to be available: • Computer with PHP development environment (e.g., XAMPP, WAMP). • Text editors/IDEs (e.g., VS Code, PHPStorm). • Sample scenarios for implementing control structures.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						invariants) in PHP • memory management for arrays in PHP. **Circumstantial knowledge:** Detailed knowledge about: Safety precautions in electronic work. • Debugging and handling errors in scripts with complex logic.		
		(c) Implementing functions	**Instruction:** Explain PHP functions, parameters, and return values. **Demonstration:** Show how to define and call functions. **Practice:** Students write reusable functions for simple tasks.	• Write PHP functions with inputs and outputs. • Use global and local variables appropriately. • Test function behavior.	Programs with reusable functions (e.g., calculating factorials, formatting strings, etc.).	**Detailed knowledge of:** **Method used:** The student should explain how to • define functions in PHP • Call functions in other parts of the PHP program **Principles:** The student should explain the principles of: • procedural programming and modular design. **Theories:** The student should explain: • The basic	The following tools and equipment are to be available: • Computer with PHP development environment (e.g., XAMPP, WAMP). • Text editors/IDEs (e.g., VS Code, PHPStorm). • Practice problems requiring function implementation.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
						concepts of PHP functions and their definitions • Scope (global vs. local), recursion, and the role of functions in reducing code redundancy. **Circumstantial knowledge:** Detailed knowledge about: • Best practices for writing maintainable and reusable functions, such as meaningful naming and parameter validation.		
		(d) Integrating PHP with JavaScript and event handling	**Brainstorm**: Guide students and explain the role of PHP and JavaScript in full-stack development. **Demonstration:** Show examples of PHP generating dynamic JavaScript.	• Use PHP to generate JavaScript code. • Handle user interactions with JavaScript events. • Test integration	Functional web pages that use PHP for backend processing and JavaScript for client-side interactivity (e.g., form validation).	**Detailed knowledge of:** **Method used:** The student should explain how • PHP and JavaScript interact in a client-server **Principles:** The student should explain the principles of:	The following tools and equipment are to be available: • Computer with PHP development environment (e.g., XAMPP, WAMP). • Text editors/IDEs (e.g., VS Code, PHPStorm).	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Practice: Students create PHP scripts with event-driven JavaScript integration.	between PHP and JavaScript.		• asynchronous operations and the role of event-driven programming. **Theories:** The student should explain: • Discuss the DOM, event propagation (bubbling vs. capturing), and asynchronous programming models. **Circumstantial knowledge:** Detailed knowledge about: • Common integration challenges, such as data serialization, and handling security concerns.	• Sample projects for PHP-JavaScript integration.	
	• Working with forms	(a) Creating forms and handle form data	**Instruct:** Explain the structure and purpose of HTML forms (e.g., <form>, input fields, and attributes like method and action).	• Create forms with appropriate elements (text inputs, dropdowns, checkboxes, etc.).	Functional forms that can accept, submit, and display user input (e.g., a contact form submitting	**Detailed knowledge of:** **Method used:** The student should explain how to • HTTP requests work and how data is passed	The following tools and equipment are to be available: • Computer with Web server environment (e.g., XAMPP, WAMP).	36

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Demonstration: Show how to create forms and retrieve data using PHP's $_POST and $_GET. **Practice:** Guide students to build forms for real-world scenarios (e.g., contact forms, survey forms).	• Retrieve and display form data using PHP. • Use proper form attributes for data submission.	data to the server).	using GET and POST. **Principles:** The student should explain the principles of: • structuring forms to ensure usability and accessibility. **Theories:** The student should explain: • form behaviours and methods • client-server model in the context of forms, focusing on how data flows between the browser and server. **Circumstantial knowledge:** Detailed knowledge about: • Data types	• Examples of form data	
		(b) Performing validation and sanitation of user input	**Brainstorm:** Guide students to discuss the concepts and importance of validation and sanitation (client-side vs. server-side).	• Validate form data on the client-side • Validate form data on the server-side. • Use PHP functions like	Functional forms that validate input (e.g., ensuring required fields are filled) and sanitize user data before	**Detailed knowledge of:** **Method used:** The student should explain how to • validate form data using JavaScript and PHP.	The following tools and equipment are to be available: • Computer with Web server environment (e.g., XAMPP, WAMP).	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Demonstration: Show how to implement validation using JavaScript (client-side) and PHP (server-side). **Practice:** Students write scripts to validate form input (e.g., checking required fields, sanitizing email input).	filter_var() for input sanitation. • Test scripts for handling invalid input gracefully.	processing.	• sanitize form data before submitting **Principles:** The student should explain the principles of: • securing user input to prevent common attacks like SQL Injection and XSS. **Theories:** The student should explain: • validation rules (e.g., regex for patterns, email validation) • PHP sanitation functions. **Circumstantial knowledge:** Detailed knowledge about: • common security issues in form handling and mitigation strategies	• Validation cases	
	• Linking forms to databases for interactive web applications	(a) Implementing insertion of form data into a database	**Brainstorm**: Guide students to recap on database CRUD operations and their importance in interactive	• Create forms with fields mapped to database columns. • Write PHP	Functional forms that successfully insert data into a database (e.g., user	**Detailed knowledge of:** **Method used:** The student should explain how to • use INSERT SQL	The following tools and equipment are to be available: • Computer • Database server (e.g., MySQL,	81

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			applications. **Demonstration:** Show how to connect to a database using PHP and insert data using prepared statements. **Practice:** Guide students to create forms that allow data insertion (e.g., a user registration form).	scripts to open and close database connection • Write PHP scripts to insert form data into a database. • Use prepared statements for secure data insertion.	records, product entries).	commands with PHP scripts to add data to a database • Open and close database connection **Principles:** The student should explain the principles of: • secure database connectivity **Theories:** The student should explain: • Concepts of database connectivity using PHP **Circumstantial knowledge:** Detailed knowledge about: • error handling during database insertion (e.g., handling duplicate entries, logging failed inserts).	MariaDB). • Sample datasets for practice.	
		(b) Implementing retrieval and display of data from a database	**Brainstorm**: Guide students to discuss the importance of data retrieval for dynamic applications.	• Write PHP scripts that retrieve data from a database using	Web applications that can dynamically retrieve and display	**Detailed knowledge of:** **Method used:** The student should explain how to • query and display	The following tools and equipment are to be available: • Computer • Database server (e.g., MySQL,	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			Demonstration: Show how to query a database and display results in HTML tables or other formats. **Practice:** Students create scripts to retrieve and display data (e.g., a product catalog).	• SELECT queries. • Write PHP and HTML scripts to display data in user-friendly formats (e.g., tables).	database content (e.g., user profiles, product listings).	data from the database **Principles:** The student should explain the principles of: • query optimization for efficient data retrieval. **Theories:** The student should explain: • the concepts of indexing, pagination, and filtering to manage large datasets effectively. **Circumstantial knowledge:** Detailed knowledge about: • security concerns like protecting against SQL injection in dynamic queries.	MariaDB). • Sample datasets for practice.	
		(c) Implementing updating and deletion of data in a database	**Brainstorm**: Guide students to discuss the role of update and delete operations in database	• Create HTML forms to update database records using UPDATE	Interactive forms that allow users to modify or delete database records (e.g.,	**Detailed knowledge of:** **Method used:** The student should explain how to • write UPDATE	The following tools and equipment are to be available: • Computer • Database server (e.g., MySQL,	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Services/Product Assessment	Knowledge Assessment		
			management. **Demonstration:** Illustrate how to implement these operations using forms (e.g., an edit profile page). **Practice:** Guide students to create forms that enable data updates and deletions.	queries. • Implement functionality for deleting records using DELETE queries. • Test for secure handling of sensitive operations.	editing user profiles or removing products).	and DELETE SQL commands embedded in PHP scripts. **Principles:** The student should explain the principles of: • user authentication and authorization in sensitive operations. **Theories:** The student should explain: • Cascading effects of deletion in relational databases (e.g., foreign key constraints). **Circumstantial knowledge:** Detailed knowledge about: • best practices for confirming critical actions (e.g., confirmation dialogs for deletion).	MariaDB). • Sample datasets for practice.	

Form Three

Table 5: *Detailed contents for Form Three*

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
1. System analysis and design	1.1. Identifying and analysing software development requirements	a) Collecting requirements	**Brainstorm**: Guide students to identify different methods for requirements collection (such as interviews, questionnaires, observation, and document analysis). **Role-Playing:** Students simulate interactions between stakeholders (e.g., clients and developers) to practice gathering requirements. **Activity:** Students conduct mock requirement-gathering exercises for a given project. **Field Study:** Students visit local businesses to gather real-world requirements under guidance.	• Identify stakeholders and define their roles in requirements collection. • Conduct requirement-gathering sessions. • Document requirements in an organized manner.	Well-documented requirements list capturing functional and non-functional needs, constraints, and assumptions.	**Detailed knowledge of:** **Method used:** The student should explain how to • collect requirements through structured techniques such as interviews, surveys, and focus groups **Principles:** The student should explain the principles of: • active listening • stakeholder management • thorough documentation **Theories:** The student should explain: • Requirement Engineering theories, such as the Requirements Hierarchy (functional, non-functional, and constraints). **Circumstantial knowledge:** Detailed knowledge about: • Data collection challenges like ambiguous requirements, conflicting stakeholder needs, and evolving requirements	The following tools and equipment are to be available: • Sample client scenarios. • Templates for requirement documentation (e.g., System Requirement Specification (SRS) Document). • Case studies.	90

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						during collection.		
		b) Analysing requirements	**Demonstration:** Guide students to categorize requirements into functional, non-functional, and constraints using an example project. **Group Discussions:** Groups collaborate to categorize and prioritize requirements, presenting their findings to the class. **Mind Mapping:** Students create visual diagrams to explore relationships between requirements.	• Accurately categorize requirements into appropriate groups. • Validate and prioritize requirements according to project constraints and goals. • Clearly document analysis outcomes.	A detailed requirements analysis report, including categorized, prioritized, and validated requirements.	**Detailed knowledge of:** **Method used:** The student should explain how to • categorize requirements (such as functional, and non-functional). **Principles:** The student should explain the principles of: • aligning with project objectives and traceability of requirements. **Theories:** The student should explain: • requirements engineering process • requirements validation (e.g., feasibility, consistency, completeness) **Circumstantial knowledge:** Detailed knowledge about: • scenarios requiring trade-offs between requirements and practical solutions for resolving conflicts.	The following tools and equipment are to be available: • Case studies on real-world requirement analysis. • Tools like mind-mapping software (Lucidchart, XMind).	
		c) Modelling use cases	**Demonstration:** Demonstrates how to create a basic use case	• Identify actors and their	Complete and accurate use case diagrams, with	**Detailed knowledge of:** **Method used:** The student should explain how to	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			diagram using UML tools. **Activity and Practice:** Students individually identify actors and use cases for a given scenario. **Project-Based Learning:** Students work on a small project (e.g., attendance tracking system) to develop use case diagrams. **Peer Teaching:** Pairs of students collaborate on creating and reviewing use case diagrams. **Think-Pair-Share:** Students brainstorm use case scenarios, discuss in pairs, and share with the class.	interactions with the system. • Develop accurate use case descriptions and diagrams. • Validate diagrams with peers or stakeholders.	detailed descriptions that cover various system interactions and scenarios.	• model use case using UML standards. **Principles:** The student should explain the principles of: • simplicity, stakeholder validation, and completeness. **Theories:** The student should explain: • Concepts of Unified Modeling Language (UML) framework. **Circumstantial knowledge:** Detailed knowledge about: • edge cases and system interactions	• Computer with UML diagramming tools (e.g., Lucidchart, Visio, Draw.io). • Examples of completed use case diagrams and scenarios	
	1.2. Building structures of the software	a) Designing system architecture	**Brainstorm**: Guide students to define systems architectures and diagramming tools. **Demonstration:** Illustrates the process of designing architecture diagrams for a real-world system using examples like an e-commerce application.	• Create basic architecture diagrams showing key components and their interactions. • Design architectures based on scalability,	Well-documented system architecture diagrams that outline components, interactions, and deployment environments.	**Detailed knowledge of:** **Method used:** The student should explain how to • justify the architectural designs. **Principles**: The student should explain the principles of: • modularity, scalability, and maintainability. **Theories**: The student should explain:	The following tools and equipment are to be available: • Computer with Diagramming tools (e.g., Lucidchart, Draw.io, Microsoft Visio). • Case studies or	90

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Activity and Practice: Students create simple system architecture diagrams for given scenarios (e.g., a library management system). **Group Discussion:** Groups brainstorm and draft potential architectures for a case study.	reliability, and efficiency.		• concepts like multi-tier architecture (e.g., client-server, 3-tier systems). • architectural patterns such as MVC, microservices, and layered architecture. **Circumstantial knowledge:** Detailed knowledge about: • trade-offs in design, such as cost versus scalability or performance.	templates of system architecture.	
		b) Designing data models	**Brainstorm**: Guide students on the design data models such as the ERD, and class diagrams **Demonstration:** **S**howcase the process of creating an ERD using a database scenario (e.g., online bookstore). **Activity and Practice:** Students develop ERDs and translate them into relational schema. **Project-Based Learning**: Students design a database model for a mini-project (e.g., school attendance system).	• Identify entities, attributes, and relationships • Translate ERDs into normalized tables and relational schema. • Use appropriate notations and conventions.	Completed ERDs and relational schemas that correctly represent the data structure and relationships.	**Detailed knowledge of:** **Method used:** The student should explain how to • develop data models using ERDs **Principles**: The student should explain the principles of: • eliminating redundancy and ensuring data integrity. **Theories**: The student should explain: • database design principles and data modeling techniques. **Circumstantial knowledge**: Detailed knowledge about: • Best practices in designing ERD	The following tools and equipment are to be available: • Computers with ERD tools (e.g., MySQL Workbench, Lucidchart). • Examples of ERDs and relational schemas for practice.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Peer Teaching: Students exchange designs, review each other's work, and provide constructive feedback. **Think-Pair-Share:** Students individually design a partial data model, pair up to refine it, and share results with the class.					
		c) Designing user interface	**Demonstration:** Demonstrates how to sketch wireframes and prototypes for a basic interface (e.g., login page). **Activity and Practice:** Students sketch wireframes for specific scenarios, like an online booking system. **Role-Playing:** Students act as users to test peers' prototypes and give feedback. **Group Work:** Teams collaborate to create low-fidelity and high-fidelity prototypes using design tools. **Brainstorming:** Groups discuss and	• Create wireframes or prototypes based on given requirements. • Address user-centered design principles (e.g., usability, accessibility). • Refine designs based on peer/user feedback.	Completed wireframes or prototypes that demonstrate functionality, usability, and aesthetic appeal.	**Detailed knowledge of:** **Method used:** The student should explain how to • create system wireframes and prototypes • interpret wireframes and prototypes **Principles**: The student should explain the principles of: • user-centered design, accessibility, and consistency. **Theories**: The student should explain: • UI/UX design principles and heuristics **Circumstantial knowledge**: Detailed knowledge about: • common design mistakes and how to fix them (e.g., cluttered interfaces, poor navigation).	The following tools and equipment are to be available: • Computer with prototyping tools (e.g., Figma, Adobe XD, Balsamiq). • Examples of wireframes and UI designs. • Sample user personas.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			propose ideas for improving an existing interface (e.g., a sample webpage).					
2. Web programming frameworks	2.1. Identifying and setting up a web development environment	a) Identifying web frameworks	**Brainstorm**: Guide students to to identify popular web frameworks (e.g., Django, Laravel, React, Angular). **Group Discussion:** Students discuss the pros and cons of different frameworks based on specific project requirements. **Case Studies:** Analyze case studies of real-world applications built with different frameworks.	• Identify key features of various frameworks. Compare frameworks in terms of usability, scalability, and popularity.	A list or report comparing at least three frameworks, detailing features, advantages, and drawbacks.	**Detailed knowledge of:** **Method used:** The student should explain how to • categorize frameworks (e.g., front-end, back-end, full-stack). **Principles:** The student should explain the principles of: • considerations when choosing frameworks, such as community support, ease of learning, and performance. **Theories:** The student should explain: • the concepts of web development frameworks • MVC architecture and its role in web development frameworks. **Circumstantial knowledge:** Detailed knowledge about: • project needs considerations	The following tools and equipment are to be available: • Computer with internet access to official documentation of frameworks (e.g., Django, Laravel, React, Angular). • Online tools for popularity comparison (e.g., Google Trends).	45
		b) Setting up web development	**Demonstration:** Demonstrate installation of tools like code editors (VS Code), package	• Install and configure tools like editors,	Fully configured development environment, including installed	**Detailed knowledge of:** **Method used:** The student should explain • popular development	The following tools and equipment are to be available: • Computer	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		tools	managers (npm, pip), and version control (Git). **Activity and Practice:** Students install and configure essential tools on their systems. **Peer Teaching:** Students share tips and tricks for configuring tools efficiently. **Interactive Workshops:** Hands-on sessions for configuring development environments in different operating systems.	package managers, and version control. • Use tools to manage basic projects.	tools and initial settings.	tools and their functionalities. **Principles:** The student should explain the principles of: • cross-platform compatibility and ease of use. **Theories:** The student should explain: • the concepts of web development tools • role of version control and dependency management in team-based projects. **Circumstantial knowledge:** Detailed knowledge about: • environment compatibility issues and solutions (e.g., Docker).	• Tutorials and guides for tool installation and configuration of tools such as VS Code, Git, npm • Virtual environments.	
		c) Performing installation and configuration of a framework	**Demonstration:** Illustrate step-by-step framework installation and configuration (e.g., Django setup with pip, Laravel with Composer). **Activity and Practice:** Students install and configure a selected framework individually. **Role-Playing:** Students take turns acting as instructors to guide peers	• install the chosen framework. • Configure basic settings such as project structure and dependencies.	A functional development environment with a framework installed	**Detailed knowledge of:** **Method used:** The student should explain how to • setup procedures for frameworks like Django, React, or Laravel. **Principles:** The student should explain the principles of: • proper installation practices, including version management **Theories:** The student should	The following tools and equipment are to be available: • Computer • Framework documentation. • Sample installation guides.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			through the setup process. **Troubleshooting Sessions:** Guide students to identify and resolve common errors during installation.			explain: • frameworks interaction with underlying platforms and dependencies. **Circumstantial knowledge:** **Detailed knowledge about:** • troubleshooting tips for common errors (e.g., dependency conflicts, missing configurations).		
		d) Creating a basic project	**Demonstration:** Create a simple project (e.g., a "Hello World" web app) using a selected framework. **Activity and Practice:** Students create their own basic projects following instructions. **Group Work:** Students work in teams to enhance their basic projects by adding simple features (e.g., form handling, basic styling). **Think-Pair-Share:** Students discuss challenges faced during project creation and share solutions.	• Create and run a basic project within the framework. • Apply basic framework features (e.g., routing, rendering) to the created project.	A simple working project demonstrating the use of the framework's features (e.g., rendering a page, handling routes).	**Detailed knowledge of:** **Method used:** The student should explain • steps to start a project, including setting up directories, configurations, and initial code. • Verify functionality of the installed framework by running a test project **Principles:** The student should explain the principles of: • simplicity and functionality in the initial project. **Theories:** The student should explain: • basic framework features (e.g., routing, rendering) **Circumstantial knowledge:** Detailed knowledge about:	The following tools and equipment are to be available: • Computer with installed framework • Examples of basic projects. • Development tools with live server capabilities.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						• debugging, testing, and iteration phases of the SDLC		
	2.2. Building the front-end using a framework	a) Implementing components and layouts	**Demonstration:** Showcase how to create components and define layouts in a framework (e.g., React, Angular, Vue). **Activity and Practice:** Students build predefined components and layouts individually. **Group Work:** Students collaborate to design a page layout using reusable components. **Case Study Analysis:** Analyze real-world websites and discuss their component structure and layout strategies. **Project-Based Learning:** Students implement components for specific use cases, such as navigation bars or carousels.	• Create reusable components within the framework. • Design and implement responsive layouts using framework-specific features.	Functional webpage with reusable components (e.g., headers, footers, cards) and responsive layouts.	**Detailed knowledge of:** **Method used:** The student should explain how • front-end frameworks (e.g., React, Angular) use component-based architecture. **Principles:** The student should explain the principles of: • modularity, reusability, and responsiveness. **Theories:** The student should explain: • concept of component hierarchies • the role of state management in layouts. **Circumstantial knowledge:** Detailed knowledge about: • challenges such as maintaining consistency across components and optimizing layout performance.	The following tools and equipment are to be available: • Computer with front-end web framework tools (e.g., React, Angular, Vue). • Sample web systems for practice.	90
		b)Performin	**Demonstration**:	• Customize	A webpage with	**Detailed knowledge of:**	The following tools	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		g customisation of styles	Illustrate how to use CSS frameworks (e.g. Bootstrap), preprocessors (e.g., Sass, LESS), or framework-specific style configurations. **Interactive Workshops:** Hands-on sessions to customize themes, colors, and typography. **Brainstorming:** Guide students to design improvements on existing systems and implement them through style customization. **Role-Playing:** Students act as developers addressing client requests for customizations.	styles using CSS, inline styles, or framework-specific utilities. • Apply consistent themes across components.	customized styles, including modified colors, typography, and spacing, adhering to a theme or branding.	**Method used:** The student should explain how to • use of CSS frameworks (e.g., Tailwind, Bootstrap) for styling. **Principles:** The student should explain the principles of: • best practices for styling. **Theories:** The student should explain: • concepts of separation of concerns in design (content vs. presentation). **Circumstantial knowledge:** Detailed knowledge about: • trade-offs between inline styles and external stylesheets • handling conflicting styles.	and equipment are to be available: • Computer with front-end web framework tools (e.g., React, Angular, Vue). • Tools like Sass/LESS compilers. • Prebuilt templates.	
		c) Implementing user interactions	**Demonstration:** Demonstrate how to add interactivity using event listeners, states, and hooks (e.g., React useState/useEffect). **Activity and Practice:** Students implement interactivity such as form validation, modals, or	• Add interactivity features like hover effects, event-driven updates, and user feedback mechanism. • Handle user interactions	A fully interactive webpage with features such as dynamic content updates, event handling, and smooth user experiences.	**Detailed knowledge of:** **Method used:** The student should explain how • DOM handles events, state changes, and animations. **Principles:** The student should explain the principles of: • usability and performance	The following tools and equipment are to be available: • Computer with front-end web framework tools (e.g., React, Angular, Vue). • Debugging	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			dropdown menus. **Think-Pair-Share:** Students pair up to discuss their implementation challenges and solutions. **Game-Based Learning:** Use interactive coding challenges to test event handling and dynamic updates.	with dynamic responses. • Debug issues related to interactivity implementation.		in interactive designs. **Theories:** The student should explain: • event propagation, binding, and state management. **Circumstantial knowledge:** Detailed knowledge about: • accessibility concerns • cross-browser compatibility issues in user interactions.	tools (e.g., Chrome DevTools). • Framework-specific event-handling guides. • Libraries for animations (e.g., GSAP).	
	2.3. Building the back-end using a framework	a) Implementing routing and handling of user requests	**Demonstration:** Demonstrate how to define routes and handle HTTP methods (GET, POST, PUT, DELETE) using a back-end framework (e.g., Laravel, Express.js, Django, Flask). **Activity and Practice:** Students create APIs with basic routes and test them with tools like Postman. **Case Study Analysis:** Analyze the routing structure of an existing sample web based system. **Group Work:** Students work in pairs to	• Create and define routes for handling user requests. • Test routes with sample requests and ensure correct response handling.	Functional server-side application that handles user requests and sends appropriate responses (e.g., JSON, HTML).	**Detailed knowledge of:** **Method used:** The student should explain how • routing works in frameworks and the importance of RESTful principles. **Principles:** The student should explain the principles of: • clean URL structures, efficient routing, and error handling. **Theories:** The student should explain: • client-server architecture and HTTP protocol. **Circumstantial knowledge:** Detailed knowledge about: • debugging routing errors and managing dynamic	The following tools and equipment are to be available: • Computer with a working front-end of the system • Frameworks like Laravel, Express.js, Django. • Framework documentation. • Local/remote server setups	135

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			debug routing issues and optimize route performance.			routes with parameters.		
		b) Implementing interaction with databases	**Demonstration:** Illustrate how to connect back-end frameworks with relational databases and use Object Relational Mapping (ORM) tools like Sequelize, Django ORM, or Mongoose. **Activity and Practice:** Students build CRUD functionality to interact with a database. **Game-Based Learning:** Students participate in coding challenges to create optimized database queries.	• Establish database connections using back-end frameworks features. • Implement basic CRUD operations. • Optimize database queries for performance.	Database-driven web applications with fully functional CRUD operations and efficient query handling.	**Detailed knowledge of:** **Method used:** The student should explain • role of databases in web applications and the integration process. **Principles:** The student should explain the principles of: • Modular, secure, and scalable back-end architecture **Theories:** The student should explain: • concepts of database schemas, relationships, and query optimization. **Circumstantial knowledge:** Detailed knowledge about: • challenges like handling large datasets, maintaining data integrity, and resolving connection issues.	The following tools and equipment are to be available: • Computer • Database servers (e.g., MySQL). • ORM libraries. • Sample datasets for practice.	
		c) Implementing authentication and authorisation	**Demonstration:** Demonstrate how to set up user authentication and role-based access control using frameworks like	• Create secure login and registration systems. • Implement role-based	A secure application that authenticates users, handles sessions or tokens and enforces role-	**Detailed knowledge of:** **Method used:** The student should explain how • authentication and authorization in securing applications is carried out.	The following tools and equipment are to be available: • Computer with a web-based systematically	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Passport.js (Node.js) or Django's authentication system. **Activity and Practice:** Students implement secure login, registration, and role-based access systems. **Case Study Analysis:** Review the security architecture of an existing sample web application.	access control for different user types.	based access.	**Principles:** The student should explain the principles of: • security best practices like hashing, salting, and avoiding hardcoding credentials. **Theories:** The student should explain: • the concepts of techniques of Authentication and Authorization • session management, token-based authentication. **Circumstantial knowledge:** Detailed knowledge about: • real-world Authentication and Authorization challenges like brute force attacks, session hijacking	• Libraries like Passport.js, JWT. • Security testing tools. • Online resources for best practices.	
3. Basics of object-oriented programming	3.1 Creating basic programs using classes and objects	a) Defining classes and creating objects	**Demonstration:** Show how to define classes and create objects in a programming language. **Hands-On Practice:** Assign tasks requiring students to define their own classes, add attributes and methods, and create multiple objects from those	The student should be able to: • Define classes with appropriate attributes and methods. • Use constructors to initialize objects. • Create and	Classes are defined with logically structured attributes and methods, and objects are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Define classes and initialize attributes. • Create and manipulate objects to perform specific tasks. • Use methods within classes to implement functionality.	The following tools and equipment are to be available: • Computers with programming environments (e.g., Python, Java, C++). • Sample problems for defining and	54

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			classes. **Interactive Tutorials:** Provide guided examples demonstrating the syntax and structure of classes and objects **Case Studies:** Analyze real-world applications of object-oriented programming (OOP) to highlight the significance of classes and objects in program design.	use objects effectively to perform tasks.		**Principles:** The student should explain the principle of: • Object-oriented programming. • The relationship between classes and objects in designing reusable code. • Code organisation and reuse through object-oriented design. **Theories:** The trainee should explain: The trainee should explain: • How objects represent real-world entities in programming. • The role of classes as blueprints for objects. **Circumstantial knowledge** Detailed knowledge about: • Common errors in defining and using classes and objects and how to resolve them. • Best practices for naming classes, attributes, and methods.	using classes and objects. • Tutorials on class and object definitions in OOP languages. • Case studies showcasing object-oriented program design. • Exercises focusing on creating and debugging classes and objects.	
		b) Organising	**Demonstration:** Show how constructors are	The student should be able to:	Constructors are correctly	**Detailed knowledge of:** **Methods:** The trainee should	The following tools and equipment are	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		programs using constructors	defined and used in a programming language to initialise objects. **Hands-On Practice:** Assign exercises where students create classes with constructors to initialise attributes and ensure the objects function correctly. **Interactive Tutorials:** Provide step-by-step examples illustrating parameterised and default constructors. **Problem-Solving Activities:** Challenge students to design classes that use constructors to handle different initialisation scenarios, such as passing data dynamically.	• Define constructors with proper syntax and initialise attributes. • Use constructors to create and initialise objects dynamically. • Incorporate constructors effectively in organizing class functionality	implemented and used to initialise objects to meet problem requirements and adhere to object-oriented principles.	explain how to: • Define constructors in classes with and without parameters. • Use constructor overloading to handle multiple initialisation scenarios. • Incorporate constructors in program design for object creation. **Principles:** The student should explain the principle of: • Object initialization using constructors. **Theories:** The trainee should explain: The trainee should explain: • The role of constructors in managing object states. • The advantages of using constructors over direct attribute assignment. **Circumstantial knowledge** Detailed knowledge about: • Common errors in constructor implementation and how	to be available: • Computers with programming environments installed (e.g., Python, Java, C++). • Sample problems for defining and using constructors effectively. • Tutorials on constructors and their applications in OOP. • Case studies showcasing real-world uses of constructors. • Exercises focusing on constructor overloading, debugging, and advanced usage.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						to resolve them. • Best practices for designing and organizing constructors in classes.		
		c) Constructing real-world models using classes and objects	**Demonstration:** Show how to design and implement real-world models using classes and objects. **Hands-On Practice:** Assign tasks where students create real-world class models, such as a Car, Student, or BankAccount, incorporating attributes and methods. **Interactive Tutorials:** Provide guided examples that demonstrate modeling real-world entities on the relationships between classes and objects. **Problem-Solving Activities:** Present scenarios where students need to design object-oriented solutions for real-world problems	The student should be able to: • Identify and translate real-world entities into classes and objects. • Design classes with appropriate attributes and methods representing real-world functionality. • Implement relationships between classes, such as inheritance and associations.	Classes and objects that represent real-world entities accurately per object-oriented principles are constructed.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Design class diagrams for real-world models. • Implement attributes and methods representing real-world properties and behaviors. • Build relationships between classes to represent associations and inheritance. **Principles:** The student should explain the principle of: • Object-oriented programming (OOP) concepts. • Designing reusable code using classes and objects. • Real-world problem-solving through object modeling. **Theories:** The trainee should explain: The trainee should explain:	The following tools and equipment are to be available: • Computers with programming environments installed (e.g., Python, Java, C++). • Diagramming tools for class and object modeling (e.g., Lucidchart, Draw.io). • Sample problems for designing and implementing real-world models. • Tutorials on object-oriented modeling and design. • Case studies showcasing real-world	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						• How real-world entities can be abstracted into classes and objects. **Circumstantial knowledge** Detailed knowledge about: • Common design patterns for modeling real-world scenarios in OOP. • Applications of OOP in real-world industries.	implementations of OOP principles. • Exercises focusing on multi-class systems and real-world problem-solving.	
	3.2 Creating programs with inheritance and method overriding	a) Implementing inheritance	**Demonstration:** Show how to create base and derived classes, including the use of extends or inherit keywords. **Hands-On Practice:** Assign tasks where students create inheritance hierarchies to model real-world relationships **Interactive Tutorials:** Provide guided examples illustrating single and multilevel. **Problem-Solving Activities:** Present scenarios where	The student should be able to: • Create base and derived classes with appropriate attributes and methods.	Programs demonstrating the proper use of inheritance are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Define base and derived classes using inheritance. • Use constructors in inheritance hierarchies to initialize base and derived class attributes. **Principles:** The student should explain the principle of: • Code reuse and modularity through inheritance. **Theories:** The trainee should explain: The trainee should explain: • How inheritance facilitates hierarchical	The following tools and equipment are to be available: • Computers with programming environments installed (e.g., Python, Java, C++). • Sample problems for practicing inheritance and method overriding. • Tutorials on inheritance and overriding techniques in different programming	36

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			inheritance improves code reuse, and ask students to implement solutions. **Case Studies:** Analyze programs that effectively use inheritance and discuss the advantages			relationships in object-oriented programming. • The importance of constructors in ensuring correct object initialization across inheritance hierarchies. **Circumstantial knowledge** Detailed knowledge about: • Common pitfalls in implementing inheritance, such as incorrect overriding or constructor chaining errors. • Real-world use cases of inheritance, such as modeling organizational structures or system components.	languages. • Case studies showcasing effective use of inheritance in software systems. • Exercises focusing on creating and debugging inheritance-based programs.	
		b) Implementing overriding parent class methods	**Demonstration:** Show how to override methods in derived classes. **Hands-On Practice:** Assign tasks requiring students to override parent class methods and customise their behavior in derived classes. **Interactive Tutorials:** Provide guided exercises	The student should be able to: • Define and override methods in derived classes. • Use super or base class references to access parent class functionality.	Parent class attributes and methods are appropriately overridden in derived classes.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Override methods in derived classes. • Use the super keyword or equivalent to access parent class methods or attributes. **Principles:** The student should explain the principle of:	The following tools and equipment are to be available: • Computers with programming environments installed (e.g., Python, Java, C++). • Sample programs and problem sets for	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			focusing on method overriding, including the use of super or base class references.			• Method overriding as a means to extend or customize parent class functionality **Theories:** The trainee should explain: The trainee should explain: • How method overriding enables dynamic behaviour in object-oriented programming. • The impact of method overriding on code reusability. **Circumstantial knowledge** Detailed knowledge about: • Common pitfalls in implementing inheritance, such as incorrect overriding or constructor chaining errors. • Real-world use cases of inheritance, such as modeling organizational structures or system components.	practicing method overriding. • Tutorials on inheritance and method overriding in OOP. • Case studies highlighting effective use of method overriding. • Exercises focusing on implementing and testing overridden methods.	
	3.3 Implementing encapsulation and	a) Implementing encapsulation of class	**Demonstration:** Show how to encapsulate class attributes and methods using access modifiers (private, protected,	The student should be able to: • Apply appropriate access	Attributes and methods are correctly encapsulated using access	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Encapsulate class attributes and methods	This element can be achieved at school workshop and The following tools and equipment are to be	36

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
	abstraction	attributes and methods	public). **Hands-On Practice:** Assign exercises requiring students to encapsulate attributes and methods in a class and access them via getter and setter methods. **Interactive Tutorials:** Provide guided examples on encapsulation and demonstrate its role in securing data. **Problem-Solving Activities:** Present real-world scenarios where encapsulation ensures secure and organized program design.	modifiers to class attributes and methods. • Use getter and setter methods to manage attribute access and updates. • Implement encapsulation to enhance data security	modifiers.	using appropriate access modifiers. • Use getter and setter methods to provide controlled access to class attributes. **Principles:** The student should explain the principle of: • Encapsulation as a means to protect data and reduce dependency between components. • Data hiding and the separation of implementation from interface. **Theories:** The trainee should explain: The trainee should explain: • How encapsulation improves code security and maintainability. • The importance of access modifiers in implementing encapsulation. • The role of getter and setter methods in managing data access.	available: • Computers with programming environments installed (e.g., Python, Java, C++). • Sample problems for practicing encapsulation in class design. • Tutorials on encapsulation and abstraction in OOP. • Case studies demonstrating secure and modular program designs using encapsulation. • Exercises focusing on implementing and debugging encapsulation.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						Circumstantial knowledge Detailed knowledge about: • Common mistakes in encapsulation and how to avoid them. • Best practices for designing encapsulated classes in real-world applications.		
		b)Implementing abstraction of functionalities using abstract classes and method	**Demonstration:** Show how to define and use abstract classes and methods. **Hands-On Practice:** Assign tasks where students create abstract classes representing general concepts and implement concrete methods in derived classes. **Interactive Tutorials:** Provide guided examples to illustrate the role of abstraction in separating functionality from implementation. **Problem-Solving Activities:** Present	The student should be able to: • Define abstract classes and methods with appropriate syntax. • Implement concrete methods in derived classes, adhering to abstract class requirements. • Demonstrate the application of abstraction to improve code structure and consistency.	Abstract classes and methods are correctly defined and implemented in derived classes.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Define abstract classes and abstract methods. • Use derived classes to implement the functionality defined by abstract methods. **Principles:** The student should explain the principle of: • Abstraction to hide unnecessary implementation details and expose essential features. • Enforcing a consistent interface through abstract classes and methods. **Theories:** The trainee should	The following tools and equipment are to be available: • Computers with programming environments (e.g., Python, Java, C++). • Sample projects and problem sets for abstract class and method implementation. • Tutorials on abstraction in object-oriented programming. • Case studies showcasing the role of abstraction in	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			scenarios where abstraction ensures a consistent interface for derived classes, such as designing a payment processing system.			explain: The trainee should explain: • How abstraction separates "what" from "how" in program design. • The role of abstract classes in creating extensible and scalable code. • The importance of adhering to interfaces defined by abstract classes in derived classes **Circumstantial knowledge** Detailed knowledge about: • Common challenges in implementing abstraction and how to address them. • Best practices for designing abstract classes in real-world applications	software design. • Exercises focusing on creating, implementing, and debugging abstract classes and methods.	
	3.4 Implementing polymorphism with interfaces and abstract classes	a) Designing abstract classes for common behaviour	**Demonstration:** Show how to define abstract classes and abstract methods that enforce common behaviour in derived classes. **Hands-On Practice:** Assign exercises where students design abstract classes to encapsulate	The student should be able to: • Define abstract classes with appropriate attributes and methods for common behavior. • Implement	Abstract classes and their methods are logically designed to encapsulate shared behaviour.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Design and define abstract classes for shared behaviors. • Implement and override abstract methods in derived classes.	The following tools and equipment are to be available: • Computers with programming environments (e.g., Python, Java, C++). • Sample problems and	54

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			shared behaviours and implement these in multiple derived classes. **Interactive Tutorials:** Provide step-by-step examples to demonstrate the relationship between abstract classes and polymorphism. **Problem-Solving Activities:** Present scenarios where abstract classes simplify the design of complex systems, such as defining a *Shape* class with a *draw()* method for various shapes.	derived classes that inherit from abstract classes and provide specific functionality.		**Principles:** The student should explain the principle of: • Abstraction and its role in enforcing consistent behaviours across multiple implementations. • Code reusability through the use of abstract classes. **Theories:** The trainee should explain: The trainee should explain: • How abstract classes promote code organization and reduce duplication. • The role of polymorphism in enabling objects to interact through common interfaces. • The benefits of separating interface definitions from implementations **Circumstantial knowledge** Detailed knowledge about: • Best practices for designing abstract classes to maximize reusability	projects for practising abstract class design and polymorphism. • Tutorials on abstract class design and usage in OOP. • Case studies demonstrating the use of abstract classes in software systems. • Exercises focusing on designing, and debugging systems using abstract classes.	
		b) Implementing polymorphism with	**Demonstration:** Show how polymorphism is achieved using abstract classes and interfaces	The student should be able to: • Define abstract	Abstract classes and interfaces are correctly defined and implemented	**Detailed knowledge of:** **Methods:** The trainee should explain how to:	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		abstract classes or interfaces	**Hands-On Practice:** Assign exercises requiring students to define abstract classes or interfaces and implement polymorphic behaviours in derived classes. **Interactive Tutorials:** Provide guided examples to demonstrate dynamic method invocation using polymorphism. **Problem-Solving Activities:** Present real-world scenarios requiring polymorphism, such as designing a payment system with multiple payment methods	classes or interfaces and their associated methods. • Implement methods in derived classes adhering to the interface.	to exhibit polymorphism behaviour.	• Define and implement abstract classes or interfaces. • Use polymorphism to invoke methods from derived classes **Principles:** The student should explain the principle of: • Polymorphism as a means to simplify complex systems through consistent interfaces. • Abstraction and its role in achieving polymorphism using abstract classes and interfaces **Theories:** The trainee should explain: The trainee should explain: • How polymorphism enhances flexibility and extensibility. • The role of abstract classes and interfaces in defining common behaviours **Circumstantial knowledge** Detailed knowledge about: • Common errors in	• Computers with programming environments installed (e.g., Python, Java, C++). • Sample projects and exercises for implementing polymorphism. • Tutorials on polymorphism, abstract classes, and interfaces in OOP. • Case studies showcasing polymorphism in large-scale software systems. • Exercises focusing on designing, implementing, and debugging polymorphic behavior.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						implementing polymorphism and how to address them. • Best practices for designing systems using abstract classes and interfaces		
		c) Performing overriding methods to implement polymorphic behaviour	**Demonstration:** Show how to override methods in derived classes to achieve polymorphism. **Hands-On Practice:** Assign exercises requiring students to override methods in real-world scenarios. **Interactive Tutorials:** Provide guided examples illustrating the dynamic dispatch mechanism used in polymorphism. **Problem-Solving Activities:** Present tasks where overriding methods in a hierarchy of classes implements specific behavior for different objects.	The student should be able to: • Override methods in derived classes adhering to the parent class or interface contracts. • Demonstrate the consistent behavior of overridden methods across different derived classes.	Overridden methods are correctly implemented to demonstrate polymorphic behavior implementation.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Override methods in derived classes while maintaining consistency with parent class **Principles:** The student should explain the principle of: • Method overriding to implement polymorphic behaviour. • Encapsulation and inheritance as enablers of polymorphism. **Theories:** The trainee should explain: The trainee should explain: • How overriding enhances flexibility and customizability.	The following tools and equipment are to be available: • Computers with programming environments installed (e.g., Python, Java, C++). • Sample projects and exercises for practicing method overriding and polymorphism. • Tutorials on polymorphism and method overriding in OOP. • Case studies showcasing polymorphism in real-world applications.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						• The impact of polymorphism on reducing code duplication. **Circumstantial knowledge** Detailed knowledge about: • Common challenges and pitfalls in method overriding and their solutions. • Best practices for designing and implementing polymorphic behaviour.	• Exercises focusing on designing, implementing, and debugging overridden methods.	
4. Event-driven programming	4.1 Working with IDE and event-driven language	a) Performing exploration of features and tools of the event-driven language IDE	**Demonstration:** Walk students through the features and tools of a popular event-driven IDE. **Hands-On Practice:** Assign tasks requiring students to explore and use the key features of the IDE **Interactive Tutorials:** Provide guided examples demonstrating how to create, test, and debug an event-driven program within the IDE.	The student should be able to: • Navigate and use the various tools and features of the event-driven IDE. • Configure the IDE for optimal event-driven programming.	Features and tools of the IDE are correctly explored.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Navigate the IDE interface and use its tools effectively for event-driven programming **Principles:** The student should explain the principle of: • Event-driven programming and its reliance on an IDE for efficient development. • The role of IDEs in improving code quality and reducing development	The following tools and equipment are to be available: • Computers with event-driven IDEs installed (e.g., Visual Studio, NetBeans, PyCharm). • Sample projects for exploring IDE features and practicing event-driven programming. • Tutorials on using IDEs for	68

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						time. **Theories:** The trainee should explain: The trainee should explain: • How IDE tools and features contribute to the software development lifecycle **Circumstantial knowledge** Detailed knowledge about: • Common challenges in using IDEs for event-driven programming and how to overcome them.	event-driven programming. • Case studies showcasing event-driven application development using IDEs. • Exercises focusing on debugging, UI design, and event handling.	
		b) Creating basic programs using event-driven language	**Demonstration:** Show how to create simple event-driven programs using an IDE with built-in event-driven language support. **Hands-On Practice:** Assign exercises to develop basic programs using common event-driven constructs	The student should be able to: • Write and execute basic event-driven programs in an IDE.	Basic programs principled on event-driven programming paradigm are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use an IDE to create event-driven applications. • Design and implement event-driven programs that respond to user inputs. **Principles:** The student should explain the principle of: • Understanding the role of event listeners, callbacks, and triggers in creating responsive applications.	The following tools and equipment are to be available: • Computers or laptops equipped with an IDE supporting event-driven programming (e.g., Visual Studio, NetBeans, PyCharm, or Eclipse). • Access to	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						Theories: The trainee should explain: The trainee should explain: • How events are queued and processed in sequence to handle asynchronous user interactions. • The phases of event propagation and their impact on event handling. **Circumstantial knowledge** Detailed knowledge about: • Common challenges in using IDEs for event-driven programming and how to overcome them.	sample event-driven projects for hands-on practice. • Internet connectivity to access online resources, documentation, and tutorials. • Tutorials or instructional videos on event-driven programming, emphasizing real-world use cases. • Case studies highlighting event-driven applications such as graphical user interfaces (GUIs) and mobile apps. • Workbooks with practical exercises to create basic event-driven programs.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
							• Reference materials and documentation for the specific programming language used (e.g., Python, Java, VB.NET)	
	4.2 Developing programs using functions and arrays	a) Constructing reusable functions	**Demonstration:** Show how to create reusable functions and integrate them into event-driven applications. **Hands-On Practice:** Assign exercises to design and implement reusable functions for specific tasks, such as input validation or mathematical calculations. **Interactive Tutorials:** Provide guided examples to demonstrate best practices for function design, such as parameterisation and return values. **Problem-Solving Activities:** Present	The student should be able to: • Write functions with appropriate parameters and return values. • Use functions effectively to organize code and reduce redundancy.	Reusable functions that meet task requirements are correctly implemented, and exhibit expected behaviour.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Define and call reusable functions in a program. • Pass parameters to functions and handle return values effectively. **Principles:** The student should explain the principle of: • Function modularity to simplify code structure and reduce duplication. • Parameterization and how it enhances function reusability. **Theories:** The trainee should explain: The trainee should explain: • The role of functions in improving software	The following tools and equipment are to be available: • Computers with IDEs installed (e.g., Visual Studio, PyCharm, NetBeans). • Sample problems for creating and testing reusable functions. • Tutorials on function design and implementation in programming languages. • Case studies showcasing the importance of reusable	99

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			challenges where students must identify repetitive code and refactor it into reusable functions.			scalability and maintainability. • How function reusability contributes to efficient programming workflows. **Circumstantial knowledge** Detailed knowledge about: • Common errors in function design, such as incorrect parameter usage or scope issues, and how to resolve them	functions in large-scale projects. • Exercises to practice function modularity and parameterization.	
		b) Implementing single-dimensional and multi-dimensional arrays	**Demonstration:** Illustrate the declaration, initialisation, and manipulation of single-dimensional and multi-dimensional arrays. **Hands-On Practice:** Assign exercises requiring students to create, populate, and manipulate arrays for real-world applications. **Interactive Tutorials:** Guide students step-by-step through examples like processing tabular data, or handling matrix operations.	The student should be able to: • Declare and initialize arrays using correct syntax. • Access, modify, and process elements within single-dimensional and multi-dimensional arrays.	Arrays are implemented accurately and meet the task requirements.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Declare and initialize single-dimensional and multi-dimensional arrays. • Access and manipulate array elements using loops and conditional statements. **Principles:** The student should explain the principle of: • Data storage and organization using arrays. • Indexing in single-dimensional and multi-dimensional arrays for efficient data access.	The following tools and equipment are to be available: • Computers with IDEs installed (e.g., Visual Studio, Python, Java, C++). • Sample datasets for practising array manipulations. • Tutorials and guides on arrays in event-driven programming. • Exercises and problem sets focusing on	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Problem-Solving Activities: Present challenges where students must use arrays for specific tasks.			• Iterative processes for array traversal and manipulation. **Theories:** The trainee should explain: The trainee should explain: • How arrays represent structured data in programming. • The role of arrays in efficient data handling and storage. **Circumstantial knowledge** Detailed knowledge about: • Common errors in array operations, such as off-by-one errors or out-of-bound accesses, and their solutions. • Practical applications of arrays in event-driven programming and real-world scenarios.	single-dimensional and multi-dimensional arrays. • Case studies showcasing the use of arrays in real-world applications.	
		c) Using control structures to traverse and process arrays	**Demonstration:** Show how to use control structures to traverse and process single-dimensional and multi-dimensional arrays. **Hands-On Practice:**	The student should be able to: • Write control structures to traverse arrays efficiently using loops	Programs that use control structures effectively to traverse and process arrays are created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use loops like for, while, and foreach to traverse arrays. • Implement conditional statements to process	The following tools and equipment are to be available: • Computers with programming environments (e.g., Python,	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Assign exercises requiring students to implement control structures to perform tasks such as summing array elements, searching for specific values, and modifying data in arrays. **Interactive Tutorials:** Provide guided examples demonstrating common operations like finding the maximum/minimum value, sorting arrays, or calculating averages. **Problem-Solving Activities:** Present real-world challenges where students need to use control structures for array traversal and processing.	and conditionals. • Implement logic to process arrays, such as filtering or updating data.		specific elements in arrays. **Principles:** The student should explain the principle of: • Array indexing and its role in traversal. • Error handling during array traversal to avoid runtime issues. **Theories:** The trainee should explain: The trainee should explain: • The impact of loop structures on the efficiency of array operations. • The relationship between control structures and array manipulation in solving complex problems. **Circumstantial knowledge** Detailed knowledge about: • Common errors in array traversal, such as off-by-one errors, and strategies to resolve them. • Best practices for using nested loops with multi-	Java, C++) installed. • Sample datasets for practicing array traversal and processing. • Tutorials and examples on using control structures with arrays. • Exercises focusing on tasks like searching, sorting, and data aggregation using arrays. • Case studies showcasing the use of control structures and arrays in event-driven applications.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						dimensional arrays.		
	4.3 Designing form with menus controls	a) Designing forms with basic controls	**Demonstration:** Show how to design forms using an IDE with built-in form designer tools. Demonstrate how to add and configure basic controls like text boxes, labels, buttons, and dropdown menus. **Hands-On Practice:** Assign exercises where students create simple forms with various controls and define their properties (e.g., size, position, color, and event handlers). **Interactive Tutorials:** Provide step-by-step tutorials that guide students in designing forms with real-world use cases, such as login forms, contact forms, or search interfaces. **Problem-Solving Activities:** Present scenarios requiring students to design and	The student should be able to: • Add, configure, and align basic controls on a form.	Forms with basic controls are designed with intuitive, user-friendly layouts tailored to meet specific task requirements.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Add and configure controls like text boxes, labels, buttons, and menus on a form. • Use properties and events to customize control behavior. **Principles:** The student should explain the principle of: • Form design, focusing on usability, accessibility, and user interaction. • Layout alignment and spacing for clear and organized forms. **Theories:** The trainee should explain: The trainee should explain: • How event-driven programming enables interactive forms. • The importance of user-centered design in form development.	The following tools and equipment are to be available: • Computers with IDEs supporting form design. • Sample projects and exercises for practising form design. • Tutorials and guides on designing forms and adding controls. • Exercises and problem sets focusing on form layout and alignment. • Case studies showcasing best practices in form design for real-world applications.	81

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			implement forms for specific applications, such as a product order form or a customer feedback interface.			**Circumstantial knowledge** Detailed knowledge about: • Common challenges in form design, such as misaligned controls or unhandled events, and their solutions. • Best practices for designing forms that enhance user experience. • Real-world applications of form design in data entry, user registration, and search interfaces.		
		b) Incorporating menus to forms	**Demonstration:** Show how to add menus to an already created form. Highlight menu customisation options (e.g. renaming menu items, adding separators. **Hands-On Practice:** Assign exercises where students add menus to existing forms (e.g., a File menu or an Edit menu). **Peer Reviews:** Encourage students to review and provide	The student should be able to: • Add and customize menus on an existing form	Appropriate menus are properly incorporated to forms.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Add menus to an existing form. • Define and configure properties for menu items. **Principles:** The student should explain the principle of: • Organizing menu items to improve usability and navigation. • Linking menus to form functionality for a cohesive user experience.	The following tools and equipment are to be available: • Computers with IDEs that support form and menu integration. • Sample forms for adding and testing menus. • Tutorials and guides on menu integration into existing forms. • Exercises focusing on	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			feedback on each other's menu designs for an existing form. **Interactive Tutorials:** Provide step-by-step tutorials on integrating menus into a form. **Problem-Solving Activities:** Present challenges requiring students to enhance existing forms by incorporating menus			**Theories:** The trainee should explain: The trainee should explain: • The role of menus in enhancing form interactivity and functionality. • The relationship between event-driven programming and menu interactions. • The importance of logical grouping and hierarchy in menu design. **Circumstantial knowledge** Detailed knowledge about: • Common challenges in integrating menus into existing forms • Best practices for maintaining consistency between menus and form functionality	adding and customizing menus for enhanced functionality. • Case studies showcasing applications with well-designed menus.	
		c) Implementing menu items to execute specific tasks	**Brainstorming:** Facilitate group brainstorming sessions to identify potential user interactions and their corresponding event-handling logic for specific controls.	The student should be able to: • Define menu items and link them to event handlers to execute specific	Menu items are appropriately linked to specific task for execution.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Create and configure menu items to execute tasks. • Attach event handlers to menu items for task execution.	The following tools and equipment are to be available: • Computers with IDEs supporting menu creation and event	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Demonstration: Showcase how to attach event handlers to controls (e.g. buttons, checkboxes, and dropdown menus). Demonstrate how event handlers respond to specific actions (e.g., clicking, selecting, or hovering). **Problem-Solving Activities:** Present real-world scenarios where students must design forms with controls and implement event handling. **Interactive Tutorials:** Guide students step-by-step in creating event-driven programs with functional controls. **Hands-On Practice:** Assign exercises requiring students to implement event handlers for different controls on forms	tasks. • Optimise menu structures for usability and task accessibility.		**Principles:** The student should explain the principle of: • Event-driven programming in linking user interactions (menu clicks) to actions. • Usability in designing menu structures and naming menu items. **Theories:** The trainee should explain: The trainee should explain: • How event handlers enable task execution in response to menu interactions. • The role of logical grouping of menu items to enhance user navigation. **Circumstantial knowledge** Detailed knowledge about: • Challenges in implementing menu tasks, such as unresponsive items or incorrect task execution, and their solutions. • Best practices for organising menu items to	handling (e.g., Visual Studio, NetBeans). • Sample projects for practicing menu implementation and task linking. • Tutorials on designing menus and programming their tasks. • Exercises focusing on creating menus for specific real-world applications. • Case studies illustrating effective menu implementation in popular software.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						maximise efficiency and accessibility		
		d) Implementing event handling for controls	**Brainstorming:** Facilitate group brainstorming sessions where students ideate menu functionalities and how they can link to real-world application needs. **Demonstration:** Show how to create menu items and link them to specific event handlers for executing tasks, such as opening files, saving data, or exiting the application, using an IDE. **Hands-On Practice:** Assign exercises where students design forms with menu items and program them to execute specific tasks **Interactive Tutorials:** Guide students step-by-step in implementing menus for real-world tasks.	The student should be able to: • Attach and configure event handlers for various controls. • Implement logic within event handlers to perform specific actions. • Test event-handling mechanisms to ensure correct functionality	Controls are handled to respond accurately to user interactions with appropriate actions.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Attach event handlers to controls programmatically or using GUI tools. • Write logic within event handlers to perform specific tasks. **Principles:** The student should explain the principle of: • Event-driven programming and its reliance on event listeners and handlers. • User interaction and control responsiveness for creating interactive forms. **Theories:** The trainee should explain: The trainee should explain: • How event-handling logic enables dynamic interaction between users and controls. • The importance of separating UI elements from event-handling logic	The following tools and equipment are to be available: • Computers with IDEs supporting event handling (e.g., Visual Studio, NetBeans, Eclipse). • Sample forms with pre-designed controls for practice. • Reference documentation for programming languages (e.g., Python, Java, C++). • Tutorials on event handling for various controls in event-driven programming. • Case studies	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Problem-Solving Activities: Present scenarios where students must design menus to manage specific tasks **Peer Reviews:** Have students present their menu designs and functionality to peers for constructive feedback and improvement suggestions.			to maintain program clarity. **Circumstantial knowledge** Detailed knowledge about: • Programming language-specific syntax and libraries for event handling. • Strategies for preventing and handling errors. • Designing user-friendly and intuitive interactions. • Handling multiple events on a single control, such as mouse clicks and key presses.	demonstrating effective event-driven form designs. • Exercises to practice attaching and testing event handlers.	
	4.4 Creating database connectivity options and reporting	a) Establishing a connection of event-driven programs to a database	**Demonstration:** Show how to set up database connections using programming languages and libraries. **Hands-On Practice:** Assign exercises where students establish connections to databases. **Interactive Tutorials:** Guide students through the process of integrating databases with event-driven programs step-by-	The student should be able to: • Configure database drivers and establish a connection in the programming environment.	A stable and secure connection to the database is created.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Set up database drivers and integrate them into the development environment. **Principles:** The student should explain the principle of: • Database connectivity, including connection pooling and driver configuration. • Secure access to	The following tools and equipment are to be available: • Computers with IDEs supporting event-driven programming (e.g., Visual Studio, NetBeans, Eclipse). • Database Management Systems like	67

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			step. **Problem-Solving Activities:** Present real-world challenges requiring database integration. **Brainstorming:** Facilitate group discussions to identify potential use cases for database connectivity in event-driven programs.			databases, including authentication and permissions. **Theories:** The trainee should explain: The trainee should explain: • Concepts on handling exceptions during database connections and operations to avoid application crashes • How database drivers (e.g., JDBC, ODBC) bridge the application and the database. **Circumstantial knowledge** Detailed knowledge about: • Managing connections in applications and handling multiple concurrent users. • Configuring timeouts and retries to maintain connection stability.	MySQL, PostgreSQL, SQLite, or MS SQL Server for hands-on practice. • Integrated Development Environments (IDEs) tools such as PyCharm, Visual Studio, or NetBeans that support database integration.	
		b) Performing basic CRUD operations using SQL queries	**Demonstration:** Show how to perform CRUD operations (Create, Read, Update, Delete) using SQL queries through an IDE. **Hands-On Practice:** Assign exercises	The student should be able to: • Write and execute SQL queries for CRUD operations. • Link CRUD operations to	CRUD operations using SQL are performed accurately to meet user needs.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Write SQL queries for basic CRUD operations. • Use event-driven programming to integrate CRUD functionality.	The following tools and equipment are to be available: • Computers installed with programming IDEs (e.g., PyCharm,	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			requiring students to perform CRUD operations. **Problem-Solving Activities:** Present scenarios where students need to design a database schema and implement CRUD operations for real-world applications. **Brainstorming:** Facilitate discussions on the types of data commonly stored in databases and how CRUD operations can be designed for specific application needs. **Group Projects:** Encourage students to collaboratively design and implement CRUD-enabled applications, such as an online registration system.	event handlers in an event-driven program.		**Principles:** The student should explain the principle of: • Data manipulation and its importance in application functionality. • SQL as a language for interacting with relational databases. **Theories:** The trainee should explain: The trainee should explain: • How CRUD operations enable dynamic and interactive application behaviour. **Circumstantial knowledge** Detailed knowledge about: • Various data types used in database tables and their application. • Designing user-friendly forms to trigger CRUD operations.	Visual Studio, NetBeans). • Database Management Systems (e.g., MySQL, PostgreSQL, SQLite, MS SQL Server). • Sample databases for practising CRUD operations (e.g., student records, inventory systems). • SQL query-building tools (e.g., MySQL Workbench, phpMyAdmin, DBeaver). • Tutorials on SQL and CRUD operations. • Exercises focusing on performing CRUD operations and integrating	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
							them with event-driven programs. • Case studies showcasing applications leveraging CRUD functionality.	
		c) Showing database records on forms	**Brainstorming:** Facilitate group discussions to explore innovative ways to represent database records visually, such as charts, tables, or summary panels. **Demonstration:** Show how to fetch records from a database and display them dynamically on forms using event-driven programming techniques. Demonstrate examples using controls like data grids, list views, or text fields. **Hands-On Practice:** Assign exercises where	The student should be able to: • Write SQL SELECT queries to retrieve data from databases. • Bind retrieved data to form controls dynamically. • Format and organize displayed data for clarity and usability.	Records are displayed accurately and in real time on forms.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Write SQL queries to fetch records based on user input or predefined conditions. • Bind data dynamically to various form controls like grids, lists, or labels. **Principles:** The student should explain the principle of: • Data representation for readability and user-friendliness. • Error handling during database queries and data binding. **Theories:** The trainee should explain: The trainee should explain:	The following tools and equipment are to be available: • Computers installed with programming IDEs (e.g., PyCharm, Visual Studio, NetBeans). • Database Management Systems such as MySQL, PostgreSQL, SQLite, or MS SQL Server for database operations. • Pre-configured databases, such as customer management or	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			students display database records in forms using various controls, such as tables or dropdown menus. **Interactive Tutorials:** Provide guided examples that integrate SQL SELECT queries with form controls to display real-time data from a database. **Problem-Solving Activities:** Present real-world challenges where students need to design forms for displaying data, such as showing employee details or product catalogs. **Peer Reviews:** Have students present their forms displaying database records for feedback on functionality and user interface design.			• How event-driven programming synchronizes user actions with database interactions. • The role of SQL in filtering and organizing data for display. **Circumstantial knowledge** Detailed knowledge about: • Techniques for designing user-friendly data displays using form controls. • Configuring and using data-binding tools or libraries in IDEs.	inventory systems, for practice. • Libraries or frameworks (e.g., JavaFX, .NET DataGridView) for creating interactive data displays. • Guides on SQL SELECT queries and data binding in event-driven programming. • Tasks for fetching and displaying records on forms, including conditional queries and pagination. • Examples of applications that display database records effectively.	
		d) Generatin	**Brainstorming:**	The student	Reports that	**Detailed knowledge of:**	The following tools	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		g reports from database data	Facilitate group discussions to explore different report formats, layouts, and how to present data effectively for various audiences. **Peer Reviews:** Have students present their reports to peers for design, clarity, and data representation feedback. **Demonstration:** Show how to fetch data from a database and generate reports. **Hands-On Practice:** Assign tasks where students create simple reports by retrieving and formatting data from databases. **Problem-Solving Activities:** Present scenarios requiring report generation, such as creating monthly financial summaries or inventory status reports.	should be able to: • Write SQL queries to retrieve data required for reports. • Format and organize data into meaningful and readable report layouts.	accurately reflect database data and that meet specified requirements are generated from database.	**Methods:** The trainee should explain how to: • Write and execute SQL queries to extract data for reporting. **Principles:** The student should explain the principle of: • Data aggregation and summarization for report generation. • Visual representation of data for clarity and decision-making. **Theories:** The trainee should explain: The trainee should explain: • How reporting enhances decision-making in data-driven applications. • The role of database queries in providing accurate data for reports. **Circumstantial knowledge** Detailed knowledge about: • Designing reports for specific use cases (e.g. financial summaries or attendance records). • Formatting data for	and equipment are to be available: • Database Management Systems such as MySQL, PostgreSQL, SQLite, or MS SQL Server. • Pre-configured datasets for generating reports, such as sales records or employee attendance. • Guides on SQL queries and report generation using reporting tools. • Tasks focusing on retrieving data, formatting, and exporting reports. • Examples of applications that use database-driven	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Case Studies: Analyze real-world applications of database-driven reports.			graphs, tables, and charts to improve visual appeal. • Integrating reports into event-driven applications for real-time data updates.	reporting for decision-making. • Pre-designed report templates for tasks like sales summaries or monthly financial overviews.	

Form Four

Table 6: *Detailed contents for Form Four*

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
1. Mobile application development	1.1. Working with mobile apps and development platforms	(a) Setting up the development environment	**Brainstorm** Guide students to discuss various current applications and areas that require mobile apps. Also identify common tools and environments for mobile app development. **Demonstration:** Demonstrate how to install and configure IDEs (e.g., Android Studio, Xcode) **Activity and Practice:** Students follow step-by-step guides to set up their own development environments. **Peer Review:** Students collaborate to troubleshoot setup issues.	• Install and configure IDEs and SDKs. • Test the installation setup by running a simple mobile application to display "Hello World".	Fully functional mobile app development environment ready to create and run basic mobile apps.	**Detailed knowledge of:** **Method used:** The student should explain • tools and processes required for mobile development. **Principles:** The student should explain the principles of: • compatibility considerations (e.g., OS versions). **Theories:** The student should explain: • concepts of mobile app, characteristics, advantages, etc. • role of SDKs, IDEs, and build tools in mobile development. **Circumstantial knowledge:** **Detailed knowledge about:** • Troubleshoot installation errors and version mismatches	The following tools and equipment are to be available: • Personal computer. • IDEs: Android Studio, Xcode. • SDKs for Android, iOS. • Tutorials or guides. • Mobile devices or emulators.	122

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		(b) Performing exploration of basic mobile user interface components	**Brainstorm:** Guide students to explore UI components like buttons, text inputs, and labels using a visual example. **Activity and Practice:** Students create a sample UI with basic components. **Role-Playing:** Act as users to test and provide feedback on UI usability. **Game-Based Learning:** Timed challenges to replicate UI designs.	• Identify and implement basic UI components. • Test functionality and responsiveness of components.	Sample mobile app with functional buttons, text fields, and labels.	**Detailed knowledge of:** **Method used:** The student should explain • the common UI components and their purposes. **Principles:** The student should explain the principles of: • usability and user-centric design. **Theories:** The student should explain: • the key UI components. • the Model-View-Controller (MVC) design pattern. **Circumstantial knowledge:** Detailed knowledge about: • platform-specific component behavior and accessibility considerations.	The following tools and equipment are to be available: • Personal computer. • IDEs: Android Studio, Xcode. • Mobile design guide books. • Tools for UI prototyping (e.g. Figma). • Sample app design templates.	
		(c) Using layouts, views, and containers	**Demonstration:** Illustrate how to use layout managers (e.g., ConstraintLayout, LinearLayout) to design mobile screens.	• Organize app screens by applying layouts and views • Create responsive and	Mobile apps with properly structured layouts and consistent navigation.	**Detailed knowledge of:** **Method used:** The student should explain • the role of layouts in mobile app design. **Principles:** The student should explain the	The following tools and equipment are to be available: • Personal computer. • IDEs. • Mobile phones	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Scenario-Based Learning: Students work on given app layout requirements. **Activity and Practice:** Students create multi-screen layouts using views and containers.	dynamic designs using a combination of containers		principles of: • responsive design and screen size adaptation. **Theories:** The student should explain: • the concept of hierarchy and nesting of views **Circumstantial knowledge:** Detailed knowledge about: • screen orientation changes	with varying screen sizes or emulators.	
		(d) Performing addition and customisation of basic user interface components	**Brainstorming:** Discuss creative ways to enhance UI design for specific use cases. **Demonstration:** Demonstrate how to add and style UI components using XML and programmatically. **Activity and Practice:** Students customize buttons, and inputs. **Peer Review:** Students present their customized UIs for feedback from their peers.	• Add components to an app. • Customize styles and interactions (e.g., button, color themes).	Mobile app with interactive and visually appealing UI elements.	**Detailed knowledge of:** **Method used:** The student should explain • styling and theming in mobile apps. **Principles:** The student should explain the principles of: • consistency in design and platform guidelines. **Theories:** The student should explain: • state management and component customization. **Circumstantial knowledge:** Detailed knowledge about:	The following tools and equipment are to be available: • Personal computer. • Style guides for mobile platforms. • Resources for designing accessible apps. • Tools for UI animation.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						• handling touch events, animations, and cross-platform compatibility of customized components.		
	1.2. Handling user input and basic interactions	(a) Performing handling of user input	**Demonstration:** Illustrate how to use input components like text fields, sliders, and switches in a mobile app. **Activity and Practice:** Students create a sample app that captures input (e.g., name, age, preferences). **Role-Playing:** Students act as users to test various input forms.	• Implement input components and collect data from users. • Write scripts to handle different types of input (e.g., text, numbers, dates).	Mobile app that successfully captures and stores user inputs.	**Detailed knowledge of:** **Method used:** The student should explain how to • handle different types of user input **Principles:** The student should explain the principles of: • accessibility and user-friendly input design. **Theories:** The student should explain: (e) different types of user input components and their application. **(f)** concepts like input validation and platform-specific input management. **Circumstantial knowledge:** Detailed knowledge about: • Handle errors such as invalid input formats and device-specific input issues.	The following tools and equipment are to be available: • Computer • IDEs: Android Studio, Xcode. • Input documentation for iOS and Android platforms. • Testing mobile apps or emulators.	148

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		(b) Performing handling of events	**Brainstorming:** Discuss examples of interactive features in apps. **Demonstration:** Teacher explains event-driven programming by creating event handlers (e.g., for button clicks, gestures). **Activity and Practice:** Students write code to handle events triggered by user actions.	• Attach event listeners to UI components. • Trigger appropriate actions based on user interaction (e.g., navigation on a button click).	Mobile app with interactive components responding appropriately to user events.	**Detailed knowledge of:** **Method used:** The student should explain how • events are handled in mobile app development. **Principles:** The student should explain the principles of: • responsiveness and efficiency in event handling. **Theories:** The student should explain: • types of events in mobile apps • event lifecycle and event bubbling. **Circumstantial knowledge:** **Detailed knowledge about:** • challenges like unresponsive components or misaligned event triggers in complex layouts.	The following tools and equipment are to be available: • Computer • Documentation on event handling. • Sample code snippets for event listeners. • Emulators or devices to simulate user actions.	
		(c) Performing validation and processing	**Demonstration:** Illustrate methods to validate user inputs (e.g., format	• Validate inputs for accuracy and completeness.	A functional app that rejects invalid inputs and processes valid	**Detailed knowledge of:** **Method used:** The student should explain • validation techniques	The following tools and equipment are to be available: • Computer set.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		user of input	checking, required fields). **Scenario-Based Learning:** Students develop an app that processes validated inputs (e.g., a registration form). **Activity and Practice:** Implement client-side and server-side validation.	• Process data effectively and securely (e.g., encrypt sensitive data).	data seamlessly.	and their relevance. **Principles:** The student should explain the principles of: • data integrity, security, and usability. **Theories:** The student should explain: • client-side vs. server-side validation and their trade-offs. **Circumstantial knowledge:** Detailed knowledge about: • edge cases like unexpected user behavior and malicious inputs.	• Guides for secure data handling. • Case studies on validation challenges in mobile apps.	
		(d) Showing feedback to users	**Demonstration:** Demonstrate how to implement feedback mechanisms like toasts, alerts, and progress indicators. **Activity and Practice:** Students add feedback features to their apps (e.g., "Submission successful"	• Provide visual or textual feedback for user actions.	Mobile app that displays clear, appropriate, and timely feedback to users.	**Detailed knowledge of:** **Method used:** The student should explain how to • provide timely feedback in mobile app. **Principles:** The student should explain the principles of: • user satisfaction and app usability. **Theories:** The student should explain:	The following tools and equipment are to be available: • Computer • User feedback design templates. • Documentation on human-computer interaction principles.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			messages). **Role-Playing:** Students provide feedback as users testing the apps.			• feedback types and best practices and the importance of feedback loops. • human-computer interaction principles **Circumstantial knowledge:** Detailed knowledge about: • Handling issues like delayed feedback or unclear messages leading to user confusion.		
	1.3. Working with data and multimedia	(a) Using local data storage	**Brainstorm**: Guide students to discuss various local storage approaches **Demonstration:** Illustrate how to use SQLite, SharedPreferences, or file storage for saving local data. **Scenario-Based Learning:** Students create a small app (e.g., a notes app) that stores and retrieves data locally. **Activity and**	• Save, retrieve, and update data using local storage techniques • Persist data between app sessions.	A functional app that efficiently uses local storage to handle data persistence.	**Detailed knowledge of:** **Method used:** The student should explain how • various local storage options and when to use each. **Principles:** The student should explain the principles of: • data integrity and security in local storage. **Theories:** The student should explain: • various local storage options and their appropriateness • data serialization,	The following tools and equipment are to be available: • Computer • SQLite and Shared Preferences documentation. • Emulators or devices for testing data persistence. • Sample apps with local storage examples.	122

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Practice: Students integrate local storage into an app.			storage limits, and file organization. **Circumstantial knowledge:** Detailed knowledge about: • issues like limited storage space and data conflicts.		
		(b) Showing images and videos	**Brainstorming:** Discuss app ideas where multimedia data is needed to be displayed. **Demonstration:** Demonstrate how to load images (e.g., using Glide or Picasso libraries) and play videos using native video players or APIs. **Activity and Practice:** Students develop an app that displays an image gallery or streams videos.	• Display images from local storage, URLs, or APIs. • Integrate video playback features (e.g., play, pause, seek).	Mobile app that effectively displays images and plays videos, with smooth transitions and user-friendly controls.	**Detailed knowledge of:** **Method used:** The student should explain how to • integrate multimedia in mobile apps **Principles:** The student should explain the principles of: • responsiveness, caching, and smooth playback. **Theories:** The student should explain: • Types of media supported by mobile apps • the role of compression formats and streaming protocols. **Circumstantial knowledge:** Detailed knowledge about: • challenges like slow	The following tools and equipment are to be available: • Computer • Libraries like Glide, Picasso, and ExoPlayer. • Multimedia file examples (images, videos). • Network tools for testing streaming content.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						loading, unsupported formats, or resource-intensive multimedia content.		
	1.4. Optimising and deploying mobile apps	(a) Performing optimisation of mobile app	**Brainstorm**: Facilitate a discussion on the reasons and cases that require optmization in mobile apps. **Demonstration:** Show different techniques for enhancing mobile app performance (e.g., reducing memory usage, improving load times). **Case Study:** Students analyze poorly optimized apps and identify areas of improvement. **Activity and Practice:** Optimize an existing app for speed, battery, and resource efficiency.	• Reduce app size by removing unused resources. • Improve app responsiveness and memory usage. • Test battery performance.	A mobile app that is lightweight, responsive, and energy-efficient.	**Detailed knowledge of:** **Method used:** The student should explain how to • optimize the mobile app using methods such as profiling, refactoring, and caching **Principles:** The student should explain the principles of: • reducing resource use and ensuring responsiveness. **Theories:** The student should explain: • the impact of algorithms, threading, and garbage collection in mobile app development. **Circumstantial knowledge:** **Detailed knowledge about:** • platform-specific performance bottlenecks.	The following tools and equipment are to be available: • Computer • Profiling tools like Android Studio Profiler. • Sample apps for optimization practice. • Mobile devices for real-world testing.	76

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		(b) Deploying the mobile app	**Demonstration:** Guide students through the process of signing and building apps for deployment on platforms (e.g., Google Play, App Store). **Activity and Practice:** Students deploy a test app to a mobile store or as an APK.	• Build and package the app for deployment. • Test the app in a production-like environment. Configure store listings.	A successfully deployed mobile app available for download or as a functional APK.	**Detailed knowledge of:** **Method used:** The student should explain • the deployment lifecycle and necessary configurations. **Principles:** The student should explain the principles of: • security (e.g., app signing) and proper versioning. **Theories:** The student should explain: • mobile app deployment options (e.g. as an APK or via app stores) • app store guidelines and approval processes. **Circumstantial knowledge:** Detailed knowledge about: • common deployment errors such as missing certificates or compatibility issues.	The following tools and equipment are to be available: • Computer • Developer accounts for app stores. • Signing tools and certificates. • Documentation on deployment best practices.	
2. Integrating cybersecurity	2.1 Identifying and	a) Identifying common cybersecurity	**Demonstration:** Show how cybersecurity	The student should be able to: • Identify	Common threats are correctly identified and	**Detailed knowledge of:** **Methods:** The trainee should explain how to:	The following tools and equipment are to be available:	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
measures into applications	controlling cybersecurity threats	threats	threats manifest in applications using simulated environments or real-life scenarios. Elaborate on Confidentiality, Integrity and Availability. **Interactive Tutorials:** Guide students through identifying security breaches by analysing application behaviour and logs. **Brainstorming:** Facilitate group discussions to explore potential cybersecurity threats in various application contexts. **Peer Reviews:** Have students present identified threats and	• cybersecurity threats. • Use systematic approaches to document cybersecurity threats	documented with sufficient detail.	• Document identified threats systematically for reporting or further action. **Principles:** The student should explain the principle of: • Application security fundamentals, such as confidentiality, integrity, and data encryption. **Theories:** The trainee should explain: The trainee should explain: • How cybersecurity threats exploit application vulnerabilities. **Circumstantial knowledge** Detailed knowledge about: • Types of cybersecurity threats, such as phishing, ransomware, SQL injection, and insider threats. • Security best	• Virtual labs for simulating and detecting cybersecurity threats. • Mock applications with deliberate vulnerabilities for hands-on threat identification. • Guides on identifying cybersecurity threats and analysing application vulnerabilities. • Reports of actual security breaches and lessons learned. • Industry standards and guidelines for application security	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			receive constructive feedback on their analyses.			practices, such as secure coding standards. • Social engineering tactics that attackers use to compromise systems.		
		b) Controlling cybersecurity threats	**Brainstorming:** Facilitate group discussions to identify creative approaches for mitigating cyber threats. **Interactive Tutorials:** Guide students through step-by-step examples of mitigating specific cybersecurity threats. **Case Studies:** Analyze historical cybersecurity breaches to evaluate the effectiveness of various mitigation strategies.	The student should be able to: • Identify appropriate mitigation strategies for specific cybersecurity threats. • Confirm that the application aligns with industry best practices and regulatory standards after mitigation.	Effective mitigation (control) measures that aligns with industry best practices for addressing cybersecurity threats are identified.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Identify threats and vulnerabilities using established frameworks. **Principles:** The student should explain the principle of: • Layered security to provide multiple barriers against threats. **Theories:** The trainee should explain: The trainee should explain: • The importance of incident response in mitigating the impact of cybersecurity threats.	The following tools and equipment should be made available: • Virtual setups to test and implement mitigation measures safely. • Mock applications with vulnerabilities for students to apply mitigation techniques. • Online platforms offering cybersecurity challenges and exercises.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Peer Reviews: Have students present their mitigation strategies for feedback and improvement suggestions. **Simulation Activities:** Use virtual labs to replicate cyber threats and allow students to practice real-time mitigation.			• The theory of encryption and how it secures data against unauthorised access. **Circumstantial knowledge** Detailed knowledge about: • Emerging threats like ransomware, zero-day attacks, and advanced persistent threats. • Best practices for securing cloud-based applications and services.	• Guides on mitigating various types of threats and securing applications. • Detailed analyses of cybersecurity incidents and responses.	
	2.2 Practising safe programming practices	a) Performing sanitisation and validation of user inputs	**Brainstorming:** Facilitate group discussions on potential threats posed by unvalidated user inputs and strategies to counter them. **Case Studies:** Analyze incidents where unsanitised inputs led to security breaches	The student should be able to: • Identify and implement appropriate sanitisation and validation techniques for different input types. • Integrate validation checks at critical points in the application.	User inputs are appropriately sanitised and validated against predefined rules.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use regular expressions and data validation libraries to verify input format. • Implement sanitisation methods to remove or neutralise harmful data. • Apply input validation at both client and server sides	The following tools and equipment are to be available: • Computer installed with programming tools (e.g. Visual Studio, PHP, Java PyCharm) for coding exercises. • Libraries and frameworks for	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			and identify lessons learned. **Demonstration:** Show how improper user input can lead to vulnerabilities like SQL injection or cross-site scripting (XSS) and demonstrate sanitisation techniques. **Interactive Tutorials:** Guide students step-by-step on how to implement input validation and sanitisation using programming libraries or frameworks. **Hands-On Practice:** Assign exercises where students sanitize inputs in sample applications to prevent attacks.			to ensure robust security. **Principles:** The student should explain the principle of: • Whitelisting allowed input formats to ensure strict validation. • Escaping special characters to prevent injection attacks. **Theories:** The trainee should explain: The trainee should explain: • How improper input validation can lead to critical vulnerabilities like SQL injection, XSS, or buffer overflow. • The role of input validation in preventing the propagation of malicious data. **Circumstantial knowledge** Detailed knowledge about:	secure coding. • Step-by-step guides on implementing input validation and sanitisation techniques. • Exercises focused on identifying vulnerabilities and applying sanitisation methods. • Case Studies analyzing past breaches due to improper input handling.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Peer Reviews: Have students review each other's code for input handling to identify potential vulnerabilities and offer suggestions.			• Data validation techniques specific to application contexts, such as numeric, date, and string inputs. • Threats posed by unescaped characters in SQL queries, HTML, or script tags. • Best practices for validating user inputs in web applications, mobile apps, and APIs.		
		b) Performing prevention of injections attacks	**Brainstorming:** Facilitate group discussions to identify potential injection attack vectors and strategies to secure applications. **Case Studies:** Analyze real-world injection attack incidents and evaluate how they could have been prevented. **Demonstration:**	The student should be able to: • Identify injection attack risks • Implement suitable preventive measures against injection attacks.	Applications are protected against injection attacks and perform securely under various input conditions.	**Detailed knowledge of: Methods:** The trainee should explain how to: • Use parameterized queries and stored procedures to secure database interactions. • Validate and sanitize user inputs to prevent injection vulnerabilities. **Principles:** The student should explain the principle of: • Input validation to prevent malicious code execution.	The following tools and equipment are to be available: • Computer installed with programming tools (e.g. Visual Studio, PHP, Java PyCharm) for coding exercises. • Libraries and frameworks for secure coding. • Step-by-step guides on	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Show examples of injection attacks, such as SQL injection or XSS, and how to mitigate them with secure coding practices. **Interactive Tutorials:** Guide students step-by-step through implementing parameterized queries and escaping user inputs to prevent injection vulnerabilities. **Hands-On Practice:** Assign tasks where students identify injection vulnerabilities in sample applications and fix them using secure techniques. **Peer Reviews:**			**Theories:** The trainee should explain: The trainee should explain: • How injection attacks exploit poorly coded applications and unsecured input fields. **Circumstantial knowledge** Detailed knowledge about: • SQL injection, XSS, LDAP injection, and other types of injection vulnerabilities.	implementing input validation and sanitisation techniques. • Exercises focused on identifying vulnerabilities and applying sanitisation methods. • Case Studies analyzing past breaches due to improper input handling.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Have students present their secured code to peers for feedback on potential vulnerabilities and improvements.					
	2.3 Encrypting and decrypting data	a) Implementing data encryption	**Brainstorming:** Facilitate group discussions on the importance of encryption for protecting personal data, like passwords or messages. **Demonstration**: Show how basic encryption techniques like Caesar cipher and XOR encryption work using step-by-step examples. **Interactive Tutorials:** Guide students to write and test encryption scripts. **Case Studies:**	The student should be able to: • Understand and apply basic encryption techniques like substitution or XOR operations. • Assess accuracy in encrypting messages or data.	Encrypted data is accurate and functions as intended.	**Detailed knowledge of: Methods:** The trainee should explain how to: • Apply basic encryption algorithms like Caesar cipher and substitution cipher. • Encode data using simple encoding schemes like Base64. **Principles:** The student should explain the principle of: • Substituting or shifting characters to create encrypted messages. • Protecting sensitive data using encryption and encoding. **Theories:** The trainee should explain: The trainee should explain:	The following tools and equipment are to be available: • Computer installed with programming tools (e.g. Visual Studio, PHP, Java PyCharm) for coding exercises. • Encryption libraries or frameworks. • Sample applications requiring encryption for data protection, such as login systems or payment platforms. • Guides on	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Analyze relatable scenarios, such as securing a secret message or protecting a personal file, to illustrate encryption's value. **Hands-On Practice:** Assign tasks where students implement and test basic encryption algorithms. Group Projects: Encourage students to design a simple messaging system that encrypts messages. **Peer Reviews:** Allow students to review and test each other's encryption solutions for feedback and			• The concept of keys in encryption and their role in encoding data. • How encryption makes intercepted data unreadable to unauthorised individuals. • The role of simple algorithms in the foundation of cybersecurity measures. **Circumstantial knowledge** Detailed knowledge about: • Applications of encryption in securing personal files and messages. • How encryption protects data transmitted over networks, like emails or chats. • Challenges of weak encryption and the importance of algorithm choice. • The difference	implementing encryption methods and managing encryption keys. • Tasks focusing on encrypting data, as well as key management. • Analyses of incidents highlighting the role of encryption in cybersecurity.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			improvement.			between encoding, encryption, and hashing at a basic level.		
		b) Implementing data decryption	**Demonstration:** Show how encrypted data is decrypted using algorithms like Caesar cipher or AES, highlighting the role of keys. **Case Studies:** Analyze scenarios where encrypted data was successfully decrypted to prevent data loss or security breaches. **Interactive Tutorials:** Guide students step-by-step to implement decryption techniques. **Brainstorming:** Facilitate group discussions on	The student should be able to: • Identify the appropriate decryption technique for a given encrypted data set. • Assess their accuracy in decrypting data using predefined keys and ensuring data integrity.	Decrypted data matches the original plaintext data without errors or loss.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Use symmetric and asymmetric decryption techniques to retrieve original data. • Test the accuracy and integrity of decrypted data. **Principles:** The student should explain the principle of: • Key management, including the role of public and private keys in decryption. • Decrypting data securely while preventing unauthorized access. **Theories:** The trainee should explain: The trainee should explain:	The following tools and equipment are to be available: • Computer installed with programming tools (e.g. Visual Studio, PHP, Java PyCharm) for coding exercises. • Decryption libraries. • Sample data consisting of encrypted files or text messages for students to decrypt during practical exercises. • Tutorials and standards on decryption techniques and best practices.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			real-world applications of data decryption (e.g. secure communications and file protection). **Hands-On Practice:** Assign tasks for decrypting data using predefined keys and algorithms. **Peer Reviews:** Have students test and review each other's decryption implementations for accuracy and security.			• How encryption and decryption processes work together to secure data. • The role of algorithms in ensuring the accuracy and reliability of decryption. **Circumstantial knowledge** Detailed knowledge about: • Applications of decryption in securing personal and business data. • Managing encryption and decryption keys in real-world scenarios like cloud storage.	• Step-by-step guides on decrypting data using basic algorithms. • Practical tasks to decrypt data using predefined keys and validate the results.	
	2.4 Handling sensitive information in programming	a) Implementing prevention of hardcoding sensitive information in source code	**Brainstorming:** Facilitate discussions on the potential consequences of hardcoding sensitive data and alternative secure practices.	The student should be able to: • Identify hardcoded sensitive information in source code. • Implement prevention of hardcoding sensitive	Sensitive information is properly coded and not exposed in the source code.	**Detailed knowledge of:** **Methods:** The trainee should explain how to: • Implement secure practices for prevention of hardcoding sensitive information **Principles:** The student	The following tools and equipment are to be available: • Computer installed with programming tools (e.g. Visual Studio, PHP, Java	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Demonstration: Show how sensitive information, such as passwords, can be hardcoded and the security risks involved. **Case Studies:** Analyze real-world scenarios where hardcoding sensitive data led to breaches, focusing on lessons learned. **Hands-On Practice:** Assign tasks where students identify and replace hardcoded sensitive information with secure alternatives.	information		should explain the principle of: • Minimising exposure of sensitive data by using secure approach. **Theories:** The trainee should explain: The trainee should explain: • How exposing sensitive information in source code increases security risks, such as unauthorised access. **Circumstantial knowledge** Detailed knowledge about: • Techniques for encrypting sensitive data before storage or transmission. • Best practices for managing passwords and cryptographic keys securely.	PyCharm) for coding exercises. • Step-by-step guides on removing hardcoded sensitive data. • Practical tasks to identify and mitigate hardcoded sensitive information in sample code. • Real-world examples of security breaches due to exposed sensitive data.	
		b) Implementing protection of sensitive information	**Brainstorming:** Facilitate group discussions to explore the risks	The student should be able to: • Identify sensitive	Sensitive information is securely stored using appropriate	**Detailed knowledge of:** **Methods:** The trainee should explain how to:	The following tools and equipment are to be available:	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		in storage	of insecure data storage and strategies for mitigating them. **Demonstration:** Show how sensitive information, such as passwords and personal data, can be securely stored using encryption and hashing techniques. **Interactive Tutorials:** Guide students through implementing secure storage practices using tools like database encryption or file encryption libraries. **Case Studies:** Analyze incidents where sensitive data was compromised due to poor storage	• information that needs protection in storage. • Implement encryption, hashing, or other secure storage techniques.	storage solution that meets industry standards and protects data from unauthorised access.	• Encrypt sensitive data before storing it in a database or file system. **Principles:** The student should explain the principle of: • Data encryption and decryption to secure information in storage. • Hashing for storing non-reversible sensitive data securely. **Theories:** The trainee should explain: The trainee should explain: • The role of encryption algorithms in securing data at rest. • The importance of hashing for password protection and why it differs from encryption. **Circumstantial knowledge**	• Computer installed with programming tools (e.g. Visual Studio, PHP, Java PyCharm) for coding exercises. • Encryption libraries for secure storage. • Database management systems that support encryption, such as PostgreSQL, or MySQL. • Key management tools. • Step-by-step guides on implementing encryption and hashing. • Practical tasks for encrypting, hashing, and securely	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			practices and identify preventive measures. **Hands-On Practice:** Assign tasks for implementing and testing secure data storage techniques in sample applications. **Peer Reviews:** Have students review and evaluate each other's data storage solutions to identify strengths and weaknesses.			Detailed knowledge about: • Tools and frameworks for implementing encryption and hashing. • Standards and guidelines for data storage security. • Using secure storage solutions for cloud-based applications.	storing sensitive data. • Examples of data breaches caused by insecure storage and their mitigation. • Documentation on secure storage practices.	
3. Integrating programming with data science	3.1. Creating basic programs	a) Performing exploration of basic programming constructs	**Brainstorm**: Facilitate students to define workshop service bay and discuss key components of a workshop service bay and the importance of proper organization and safety.	• The student should be able to: • Use of correct syntax for variables, loops, and conditionals. • Identify and	A functional Python script showcasing proper use of variables, conditionals, and loops.	**Detailed knowledge of:** **Method used:** The student should explain: • Step-by-step explanation of Python basics (syntax, semantics). **Principles:** The student should explain the principles of:	The following tools and equipment are to be available: • Computer • Python IDE (e.g., PyCharm, VS Code, or Jupyter Notebook). • Python	112

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			- **Demonstration:** Demonstrate basic constructs such as variables, conditionals, loops, and data types. **Hands-On Practice:** Students write small programs to understand constructs like if statements, for loops, and while loops. **Guided Inquiry:** Students analyze sample code snippets for functionality and errors.	correct logical errors in code.		• clean and efficient coding practices. **Theories:** The student should explain: • basic concepts and applications of Python programming language • data types and variables in Python • flow control (sequential, conditional, iterative). **Circumstantial knowledge:** Detailed knowledge about: • debugging computer programs.	documentation or tutorials.	
		b) Implementing handling of user inputs and outputs	**Brainstorm**: Guide students to discuss real-world input/output scenarios. **Activity and Practice:** Create programs that take user input (e.g., input()) and display outputs with formatted text (e.g., print() with formatting).	• Implement user inputs and outputs in a mobile app. • Handle both app's expected and unexpected inputs.	A Python script that interacts effectively with users via prompts and displays formatted outputs.	**Detailed knowledge of:** **Method used:** The student should explain how to: • Handle inputs and producing outputs. **Principles:** The student should explain the principles of: • user-friendly interaction in programs. **Theories:** The student should explain: • the role of data	The following tools and equipment are to be available: • Computer • Python IDE (e.g., PyCharm, VS Code, or Jupyter Notebook). • Real-world examples for input/output interaction.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						validation and error handling in user interactions. **Circumstantial knowledge:** Detailed knowledge about: • Handling edge cases in input/output.		
		c) Using basic libraries for data science	**Demonstration:** Introduce libraries such as NumPy, pandas, and matplotlib for basic data analysis. **Hands-On Practice:** Students perform simple tasks like creating arrays, dataframes, or plotting graphs. **Brainstorming:** Discuss how these libraries simplify data science tasks.	• Install and import relevant libraries for data analysis. • Use Python libraries to manipulate and analyze small datasets.	A Python program using libraries to load, process, and visualize basic data.	**Detailed knowledge of:** **Method used:** The student should explain how to: • explore different data analysis libraries **Principles:** The student should explain the principles of: • modular programming using libraries. **Theories:** The student should explain: • Common examples of Python libraries • Concept of data manipulation and visualization. **Circumstantial knowledge:** Detailed knowledge about: • Data science libraries	The following tools and equipment are to be available: • Computer • Python library documentation. • Small sample datasets.	
	3.2. Collecting and	a) Collecting data from	**Brainstorm**: Guide the students to	• Collect field using	Collected and stored data of high	**Detailed knowledge of:** **Method used:** The student	The following tools and equipment are	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
	storing data	multiple sources	discuss real-world examples of multi-source data collection (e.g., APIs, web scraping, CSV files). **Demonstration:** Demonstrate data collection using APIs and web scraping tools like requests and BeautifulSoup. **Activity and Practice:** Students collect datasets from online open sources (e.g., Kaggle, government portals) and from the field using field visit approaches such as survey	traditional data collection methods • Extract data from diverse online sources using automated tools. • Use API calls or web scraping techniques. • Store the collected data in suitable format such as Excel, CSV, JSON	quality using either traditional field visits or online sources.	should explain how to: • collect data using traditional methods • us Python tools for online data collection. **Principles:** The student should explain the principles of • ethical and legal considerations in data collection. **Theories:** The student should explain: • Types of data (field data, online data) • dealing with structured and unstructured data. **Circumstantial knowledge:** Detailed knowledge about: • dealing with missing or inconsistent data across sources.	to be available: • Traditional data collection tools such as notebooks • Electronic data collection tools such as Google Forms • Internet access for working with APIs. • Tools like requests, Beautiful Soup, or Selenium. • Sample APIs or websites for practice.	
		b) Implementing retrieval of data from files	**Demonstration:** Demonstrate reading data from CSV, Excel, or JSON files using libraries like pandas or csv. **Activity and Practice:**	• Retrieve data correctly from various file types. • Handle file errors correctly.	A Python program that reads, cleans, and processes data from files such as CSV, Excel, or JSON.	**Detailed knowledge of:** **Method used:** The student should explain how to: • read and write files using Python **Principles:** The student should explain the principles of	The following tools and equipment are to be available: • Computer installed with libraries such as pandas, csv, or openpyxl.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			Students retrieve and process data from provided datasets. **Problem Solving:** Tasks involve handling edge cases (e.g., missing values, incorrect formats).			• data integrity during retrieval. **Theories:** The student should explain: engineering processes. **Circumstantial knowledge:** Detailed knowledge about: • handling large or corrupted files.	• Sample datasets in CSV, JSON, or Excel format.	
		c) Implementing local data storage	**Demonstration:** Illustrate how to save processed data to flat files using Python (e.g., writing to CSV or JSON with pandas or json). **Hands-On Practice:** Students create scripts to save cleaned datasets locally in their computers. **Discussion:** Compare formats (CSV, JSON, Excel) for storing data and their trade-offs.	• Write processed data using appropriate file formats. • Proper naming and directory organization for stored files.	A well-structured local directory with saved datasets in appropriate formats.	**Detailed knowledge of:** **Method used:** The student should explain how to: • save cleaned and formatted files to a local computer. **Principles:** The student should explain the principles of • file format selection for data analysis and sharing. **Theories:** The student should explain: • Concepts of file formats. **Circumstantial knowledge:** Detailed knowledge about: • Handling large datasets that might	The following tools and equipment are to be available: • Computer • Python libraries (pandas, json). • Varied datasets (text-heavy, numeric-heavy, mixed types).	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						exceed memory limits during saving.		
		d) Performing connection to remote databases	**Demonstration:** Illustrate how to connects to a remote database using libraries like psycopg2 for PostgreSQL or mysql-connector-python for MySQL. **Activity and Practice:** Students practice retrieving and storing data in remote databases. **Role-Playing:** Students simulate client-server interactions.	• Read and proces data from flat files. • Handle errors and clean data. • Handle credentials securely.	A Python program that retrieves and processes data from local flat files (e.g., CSV, Excel, or JSON).	**Detailed knowledge of:** **Method used:** The student should explain how to: • read process data from remote storage. **Principles:** The student should explain the principles of • storage based on accessibility and security needs. **Theories:** The student should explain: • Trade-offs between local storage, shared network storage, and cloud solutions. **Circumstantial knowledge:** Detailed knowledge about: • Limitations in storage or retrieval due to connectivity issues.	The following tools and equipment are to be available: • Computer • Optional introduction to tools like Google Drive API or Amazon S3 for remote storage	
	3.3. Cleaning and organizing data	a) Performing basic data cleaning	**Demonstration:** Illustrate cleaning techniques like handling missing values, removing duplicates, and correcting data types	• Identify and handle missing data (e.g., imputation or removal). • Handle duplicate	A cleaned dataset free from missing values, duplicates, or data type issues.	**Detailed knowledge of:** **Method used:** The student should explain how to: • handle common data cleaning issues. **Principles:** The student should explain the	The following tools and equipment are to be available: • Computer wit tools: pandas, numpy. • Messy datasets	63

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			using libraries such as pandas. **Activity and Practice:** Students work with messy datasets to clean data. **Discussion:** Ethical implications of modifying raw data.	records or invalid entries.		principles of • consistency, reliability, and accuracy. **Theories:** The student should explain: • Data quality dimensions (e.g., completeness, validity). **Circumstantial knowledge:** Detailed knowledge about: • Data quality issues	(e.g., CSVs with missing, duplicated, or invalid values).	
		b) Performing basic data transformations	**Demonstration:** Illustrate data transformation techniques like filtering, sorting, renaming columns, and deriving new features. **Hands-On Practice:** Students create derived variables, reorder datasets, and filter data.	• Manipulate data structures and apply transformations like filtering and sorting. • Create meaningful derived variables.	A transformed dataset that meets specified requirements (e.g., sorted, filtered, or with additional features).	**Detailed knowledge of:** **Method used:** The student should explain how to: • apply transformations on data **Principles:** The student should explain the principles of • Reproducibility in data transformations. **Theories:** The student should explain: • Feature engineering and its role in improving data utility. **Circumstantial knowledge:** Detailed knowledge about:	The following tools and equipment are to be available: • Computer • Tools: pandas, numpy. • Sample scenarios requiring transformations (e.g., filtering outliers, creating summary columns).	6

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
						• Handling errors when performing complex transformations (e.g., applying functions to entire columns).		
		c) Performing basic data normalisation and standardisation	**Demonstration:** Demonstrate the concepts of normalization (scaling data to a range) and standardization (z-scores) using sklearn. **Activity:** Normalize and standardize datasets for a machine learning task. **Discussion:** Advantages of scaling in improving model performance.	• Scale numerical data effectively. • Apply normalization or standardization based on context.	A preprocessed dataset with numerical features normalized or standardized.	**Detailed knowledge of: Method used:** The student should explain how to: • normalize data using Python • standardize data using Python. **Principles:** The student should explain the principles of • Reducing feature bias in data analysis. **Theories:** The student should explain: • Normalization versus standardization (e.g., min-max scaling vs. z-score). **Circumstantial knowledge:** Detailed knowledge about: • Deciding when scaling is necessary (e.g., for algorithms sensitive to data ranges).	The following tools and equipment are to be available: • Computer • Tools: sklearn.preprocessing. • Datasets with diverse ranges for scaling and standardization exercises.	
		d) Using	**Demonstration:**	• Utilize	Scripts that		The following tools	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		specialised libraries for data cleaning	Highlight advanced libraries like pyjanitor and dataprep for cleaning large datasets. **Activity:** Explore time-saving features of specialized libraries (e.g., chaining operations). **Discussion:** When to use built-in tools versus specialized libraries.	advanced libraries for repetitive cleaning tasks. • Apply chaining operations for efficiency.	demonstrate efficient cleaning workflows using specialized libraries.	**Detailed knowledge of:** **Method used:** The student should explain how to: • choose between built-in tools versus specialized libraries. **Principles:** The student should explain the principles of • Efficiency in handling repetitive tasks. **Theories:** The student should explain: • Role of automation in reducing errors during data cleaning. **Circumstantial knowledge:** Detailed knowledge about: • chaining methods versus performing operations sequentially.	and equipment are to be available: • Computer • Tools: pyjanitor, dataprep, pandas-profiling. • Datasets requiring complex cleaning (e.g., merging, reshaping).	
	3.4. Analysing data	a) Performing basic statistical analysis	**Brainstorm**: Guide students to discuss and recap different types of basic statistics **Demonstration:** Illustrtae statistical measures (mean, median, mode,	• Compute descriptive statistics using Python libraries. • Identify appropriate statistical measures for	A report summarizing key statistical insights (e.g., central tendency, variability, distribution).	**Detailed knowledge of:** **Method used:** The student should explain • demonstrate step-by-step the statistical computations. **Principles:** The student should explain the principles of:	The following tools and equipment are to be available: • Computer • Tools: numpy, scipy, pandas. • Example datasets (e.g., weather, sales,	95

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			variance, standard deviation) using libraries like numpy and scipy. **Hands-On Activity:** Students calculate descriptive statistics for datasets. **Discussion:** Importance of understanding data distribution.	given data.		• Central tendency, dispersion, and inferential statistics. **Theories:** The student should explain: • Different statistical methods • Importance of statistical significance. **Circumstantial knowledge:** Detailed knowledge about: • Patterns in datasets and interpreting variability.	or student performance data).	
		b) Using specialised libraries to analyse data	**Demonstration:** Illustrate specialized libraries like statsmodels and scipy for advanced statistical testing (e.g., t-tests, ANOVA). **Activity:** Students use libraries for hypothesis testing and correlation analysis. **Discussion:** Choosing the right statistical test for the	• Apply specialized libraries to solve statistical problems. • Interpret statistical test results.	Python scripts demonstrating use of specialized libraries for statistical analysis and visualizations.	**Detailed knowledge of:** **Method used:** The student should explain how to • apply specialized libraries for real-world data analysis problems. **Principles:** The student should explain the principles of: • Statistical testing for hypothesis validation **Theories:** The student should explain: • Fundamentals of correlation,	The following tools and equipment are to be available: • Computer • Tools: statsmodels, scipy, seaborn. • Datasets requiring hypothesis testing or advanced statistical computations.	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
			data type.			regression, and p-values. **Circumstantial knowledge:** Detailed knowledge about: • assumptions and limitations of statistical methods used.		
	3.5. Visualising data	a) Creating basic plots using libraries	**Demonstration:** Illustrate how to use libraries like matplotlib and seaborn for basic plots (e.g., line, bar, scatter, histogram). **Activity:** Students create plots for datasets. **Discussion:** Choosing the right type of plot for the data.	• Use data analysis libraries to generate plots. • Label and style plots appropriately.	Visualizations (e.g., bar charts, scatter plots) that effectively communicate data insights.	**Detailed knowledge of:** **Method used:** The student should explain how to: • Demonstrate plotting techniques and customization. **Principles:** The student should explain the principles of: • Data visualization. **Theories:** The student should explain: • Axes, scales, and color schemes. **Circumstantial knowledge:** Detailed knowledge about: • Selecting visualizations that align with the data type and target audience.	The following tools and equipment are to be available: • Computer • Tools: matplotlib, seaborn. • Example datasets (e.g., sales, population, or survey data).	27
		b) Creating	**Demonstration:**	• Add	Interactive charts	**Detailed knowledge of:**	The following tools	

Module Title (Main Competence)	Unit Title (Specific Competences)	Elements (Learning Activities)	Suggested Teaching and Learning Methods	Assessment Criteria			Training Requirements/ Suggested Resources	Number of Periods per Unit
				Process Assessment	Product/Services Assessment	Knowledge Assessment		
		basic interactive visualisations	Demonstrate using libraries like plotly and bokeh for interactive visualizations. **Activity:** Students create dashboards or interactive plots.	interactivity (e.g., zoom, hover) to visualizations. • Integrate widgets in visualization.	(e.g., interactive scatter plots, dashboards) that allow user interaction and exploration.	**Method used:** The student should explain how to: • create interactive elements. **Principles:** The student should explain the principles of: • user engagement through interactivity. **Theories:** The student should explain: • Concepts of responsiveness and dynamic data updates. **Circumstantial knowledge:** Detailed knowledge about: • interactivity features based on the target audience's needs and technical environment.	and equipment are to be available: • Computer. • Tools: plotly, bokeh, dash. • Sample scenarios for creating dashboards (e.g., financial or health data).	

Bibliography

1. Vocational Education and Training Authority. (2023). *Curriculum for Computer Programming*. Vocational Education and Training Authority

2. Ministry of Education, Science and Technology. (2023). *Curriculum for Ordinary Secondary Education, Form I-IV*. Dar Es Salaam: Tanzania Institute of Education