GCE ADVANCED LEVEL · COMPUTER SCIENCE (0795)

A-Level Computer Science Syllabus

A unified, non-overlapping 7-module curriculum arranged chronologically from Lower Sixth foundations to Upper Sixth mastery, structured into Modules, Chapters, and Lessons.

SOURCE: MINESEC NATIONAL HARMONISED PROGRESSION SHEETS (LOWER SIXTH & UPPER SIXTH) · ORIGIN TAGS SHOW PROGRESSION (L6 → U6)

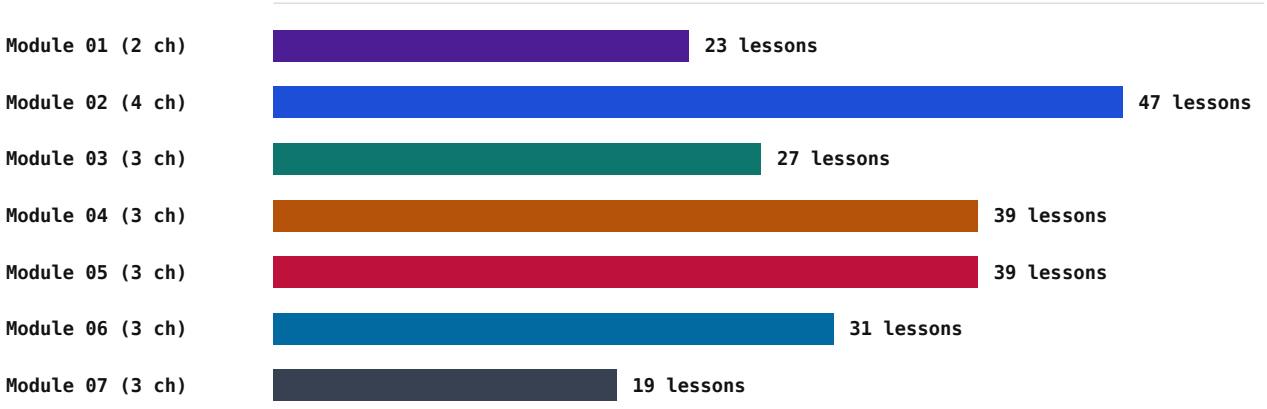
225

7	21	0
SELF-CONTAINED MODULES	CHAPTERS	CROSS-MODULE SPLITS

Every Computer Science domain lives inside a single self-contained module with zero cross-class splitting, while lessons inside each chapter progress chronologically from Lower Sixth (L6) through Upper Sixth (U6).

CURRICULUM BREAKDOWN

Lessons by Thematic Module



Module 01 — Computer Applications, AI, Emerging Tech & Socio-Economic Implications (2 Chapters · 23 Lessons)

Module 02 — Data Representation, Digital Logic & Computer Architecture (4 Chapters · 47 Lessons)

Module 03 — Operating Systems & System Software (3 Chapters · 27 Lessons)

Module 04 — Data Structures, Algorithms & Complexity (3 Chapters · 39 Lessons)

Module 05 — Programming Paradigms, C/Python Coding & Software Engineering (3 Chapters · 39 Lessons)

Module 06 — Information Systems, Databases & SQL (3 Chapters · 31 Lessons)

Module 07 — Computer Networks, Cybersecurity & Web Development (3 Chapters · 19 Lessons)

MODULE 01 · 2 CHAPTERS · 23 LESSONS

Computer Applications, AI, Emerging Tech & Socio-Economic Implications

Unifies Artificial Intelligence, Machine Learning, data collection/analysis, digital collaboration, manual/automated simulation systems, productivity tools, computer crime, cyber law, and socio-economic/ethical implications into one complete module.

CHAPTER 1.1 · 12 LESSONS (CHRONOLOGICAL L6 → U6)

Artificial Intelligence, Machine Learning & Data Analysis

Competency: Define AI/ML, ethical AI, intelligent systems, data collection/analysis, and Python AI development.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
1.1.1 L6 · L1	**Introduction to Artificial Intelligence — Part 1**	Define AI and explain its history/evolution; identify main types of AI and explain common applications.
1.1.2 L6 · L2	**Introduction to Artificial Intelligence — Part 2**	Explain ethical issues in AI; examine risks and benefits; discuss principles for responsible AI use.
1.1.3 L6 · L3	**AI Techniques and Machine Learning**	Describe AI techniques and intelligent systems; define Machine Learning; distinguish supervised, unsupervised, and reinforcement learning.
1.1.4 L6 · L4	**Developing AI Systems & Importance of Data — Part 1**	Explain stages in developing an AI system; identify AI languages/tools; explain why Python is widely used.
1.1.5 L6 · L5	**Developing AI Systems & Importance of Data — Part 2**	Distinguish primary vs. secondary data; classify quantitative, qualitative, structured, and unstructured data; relate data quality to AI reliability.
1.1.6 L6 · L6	**Data Collection and Data Analysis**	Use questionnaires, interviews, observations, and digital forms; describe data analysis, tools (ML, spreadsheets, DBMS), and objectives.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
1.1.7 L6 · L7	**Integration Activity (AI & Data)**	Consolidate AI concepts, data classification, and data analysis through applied exercises.
1.1.8 U6 · L1	**Introduction to AI & AI Ethics**	Define AI, history, types, and applications; explain ethical issues, risks/benefits, and principles for responsible AI use.
1.1.9 U6 · L2	**AI Techniques and Intelligent Systems**	Describe AI techniques and applications; explain and compare intelligent systems.
1.1.10 U6 · L3	**Machine Learning**	Define Machine Learning; explain ML principles; distinguish supervised, unsupervised, and reinforcement learning.
1.1.11 U6 · L4	**Developing AI Systems**	Explain stages of AI development; identify AI languages/tools; explain why Python is widely used.
1.1.12 U6 · L5	**Integration Activity (AI & Emerging Tech)**	Consolidate AI and ML concepts through applied exercises.

CHAPTER 1.2 · 11 LESSONS (CHRONOLOGICAL L6 → U6)

Digital Systems, Productivity Tools & Socio-Economic/Ethical Impact

Competency: Use collaboration/productivity tools, compare manual vs. automated/AI systems, and evaluate security, cybercrime, and legal/ethical duties.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
1.2.1 L6 · L11	**Communication and Collaboration Tools**	Describe and select communication/collaboration tools; set up a learning environment; collaborate and share via networks and social media.
1.2.2 L6 · L12	**Digital Systems: Manual, Automated, Processing & Simulation — Part 1**	Compare manual vs. automated systems; describe monitoring/control systems and batch vs. online processing.
1.2.3 L6 · L13	**Digital Systems: Manual, Automated, Processing & Simulation — Part 2**	Describe simulation/modelling systems; outline pros/cons of manual vs. automated systems; explain AI condition monitoring and prediction.
1.2.4 L6 · L14	**Integration Activity (Digital Resources)**	Applied digital collaboration and system classification exercises.
1.2.5 L6 · L15	**Computer Applications and Productivity Tools**	Identify domains of general-purpose applications (commerce, industry, science, education, media); use word processors, DTP, spreadsheets, DBs, slides.
1.2.6 L6 · L16	**Social & Economic Implications, Security — Part 1**	Identify social/economic effects of computerisation on employment and work; evaluate pros/cons; explain security, reliability, and resilience.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
1.2.7 L6 · L17	**Social & Economic Implications, Security — Part 2**	Explain CIA + privacy; analyse consequences of system failure/breaches; evaluate social, economic, privacy, and security implications of AI.
1.2.8 L6 · L18	**Computer Crimes, Professional Ethics and Conduct**	Define and combat computer crimes; apply professional codes of conduct; explain computer misuse, copyright, and data protection laws.
1.2.9 L6 · L19	**Legislation, Global Communication & Ethical Use — Part 1**	Analyse global communication effects on citizenship and culture; evaluate the digital divide and promote responsible participation.
1.2.10 L6 · L20	**Legislation, Global Communication & Ethical Use — Part 2**	Examine health/workplace effects of computing; differentiate legislation vs. ethics; emulate professional computing obligations.
1.2.11 L6 · L21	**Integration Activity (Ethics & Implications)**	Case-study analysis on computing implications, ethics, and cyber law.

MODULE 02 · 4 CHAPTERS · 47 LESSONS

Data Representation, Digital Logic & Computer Architecture

Complete hardware and data foundations: number bases, signed & IEEE 754 fractional arithmetic, character/image/sound encoding, combinational/sequential logic, K-map minimisation, Von Neumann/Harvard/RISC/CISC/Flynn architectures, assembly addressing, memory hierarchy, and I/O.

CHAPTER 2.1 · 11 LESSONS (CHRONOLOGICAL L6 → U6)

Number Systems, Signed/Fractional Arithmetic & Data Encoding

Competency: Convert bases, perform two's complement and IEEE 754 floating-point arithmetic, and encode characters, images, and sound.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.1.1 L6 · L24	**Base Conversions & Operations on Number Systems — Part 1**	Describe number systems and positional values; convert between binary, decimal, octal, and hexadecimal.
2.1.2 L6 · L25	**Base Conversions & Operations on Number Systems — Part 2 (Arithmetic & Units)**	Carry out arithmetic in bases 2, 8, 10, and 16; distinguish kibi/kilo, mebi/mega, gibi/giga, tebi/tera.
2.1.3 L6 · L26	**Operations on Number Systems & Data Units — Part 2 (BCD & AI)**	Describe practical applications of BCD and hexadecimal; explain how binary/hex support AI data storage and processing.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.1.4 L6 · L27	**Representation of Signed Integers**	Describe and represent signed integers using sign-magnitude, one's complement, and two's complement.
2.1.5 L6 · L28	**Signed Integer Arithmetic**	Perform addition and subtraction using sign-magnitude, one's complement, and two's complement; identify overflow and underflow.
2.1.6 L6 · L29	**Representation of Characters, Images & Sound — Part 1**	Explain ASCII, EBCDIC, Unicode; describe image digitisation (pixels, resolution, colour depth).
2.1.7 L6 · L30	**Representation of Characters, Images & Sound — Part 2**	Describe sound digitisation (sampling rate, bit depth); explain how character, image, and sound data serve as AI inputs.
2.1.8 L6 · L31	**Integration Activity (Data Representation)**	Number base, signed arithmetic, and multimedia encoding drills.
2.1.9 U6 · L12	**Representation of Fractional Numbers — Part 1**	Explain and represent fractional numbers in fixed-point and floating-point binary; relate numerical precision to AI systems.
2.1.10 U6 · L13	**Representation of Fractional Numbers — Part 2**	Convert floating-point binary to decimal; normalise representations; explain precision errors and why precision matters in AI.
2.1.11 U6 · L14	**Integration Activity (Fractional Representation)**	Fixed-point and floating-point conversion and normalisation drills.

CHAPTER 2.2 · 10 LESSONS (CHRONOLOGICAL L6 → U6)

Digital Logic Gates, Circuits, Flip-Flops & K-Map Minimisation

Competency: Design combinational, arithmetic, and sequential (SR/JK) circuits, and simplify expressions via Boolean algebra and Karnaugh maps.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.2.1 L6 · L32	**Digital Logic and Logic Gates**	Explain role of digital logic circuits; describe symbols and functions of basic, universal, and exclusive gates; construct truth tables.
2.2.2 L6 · L33	**Logic Circuit Design 1 — Part 1**	Define combinational logic circuits; explain gate/input/output relationships; design simple combinational circuits.
2.2.3 L6 · L34	**Logic Circuit Design 1 — Part 2**	Analyse digital circuits to derive outputs and truth tables; explain how logic circuits underpin AI digital hardware.
2.2.4 L6 · L35	**Binary Arithmetic Circuits**	Explain and design half-adder and full-adder circuits using logic gates; explain half-subtractor and full-subtractor circuits.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.2.5 L6 · L36	**Sequential Logic Circuits**	Differentiate combinational vs. sequential circuits; describe flip-flops; derive truth tables and outputs of SR and JK flip-flops.
2.2.6 L6 · L37	**Practical Digital Logic & Sequential Circuits**	Implement universal NAND/NOR logic, design circuits (e.g., traffic lights), use De Morgan's laws, and relate sequential storage to AI hardware.
2.2.7 L6 · L38	**Integration Activity (Digital Logic)**	Combinational and sequential logic design lab.
2.2.8 U6 · L15	**Boolean Algebra Principles**	Derive Boolean expressions from circuits/truth tables; simplify using Boolean laws; relate Boolean logic to rule-based AI.
2.2.9 U6 · L16	**Standard Forms and Logic Minimisation**	Derive SOP and POS forms; use Karnaugh maps to minimise Boolean expressions and logic circuits for rule-based AI.
2.2.10 U6 · L17	**Integration Activity (Boolean Minimisation)**	K-map minimisation and logic circuit synthesis.

CHAPTER 2.3 · 18 LESSONS (CHRONOLOGICAL L6 → U6)

Computer Organisation, Processor Architectures & Machine Cycle

Competency: Explain hardware components, embedded systems, buses, fetch-decode-execute cycle, RISC/CISC, Flynn's taxonomy, and assembly labs.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.3.1 L6 · L8	**Components of a Computer & Basic Operations — Part 1**	Connect I/O and processing devices; perform soft/warm booting; transfer data; print; identify hardware needed for AI applications.
2.3.2 L6 · L9	**Components of a Computer & Basic Operations — Part 2**	Describe types of computers, how computer systems function, and identify input and output devices.
2.3.3 L6 · L10	**Integration Activity (Computer Systems)**	Practical hardware setup and basic operations review.
2.3.4 L6 · L22	**Computer Hardware Components & Embedded Systems — Part 1**	Connect hardware and outline embedded system pros/cons; describe laser/3D printers, mics, speakers, disks, flash, optical, touch, VR headsets.
2.3.5 L6 · L23	**Computer Hardware Components & Embedded Systems — Part 2**	Explain roles of motherboard, PSU, video card, and sound card; relate hardware devices to applications and users.
2.3.6 L6 · L39	**Introduction to Computer Organisation & Architecture — Part 1**	Define and differentiate computer organisation vs. architecture; identify major computer system components.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.3.7 L6 · L40	**Introduction to Computer Organisation & Architecture — Part 2**	Explain CPU, memory, and I/O roles; explain stored-program concept; illustrate Von Neumann and Harvard architectures.
2.3.8 L6 · L41	**Advanced Processor Architectures — Part 1**	Compare RISC vs. CISC (pipelining, registers); compare Von Neumann, Flynn (SISD, SIMD, MISD, MIMD), CRAY, Dataflow, Harvard; analyse parallel processing.
2.3.9 L6 · L42	**Advanced Processor Architectures — Part 2**	Outline virtual machines; compare Von Neumann vs. Harvard pros/cons; explain why parallel/specialised architectures matter for AI.
2.3.10 L6 · L43	**System Communication and Timing**	Explain data, address, and control buses; illustrate CPU-memory-I/O transfers; explain system clock and Immediate Access Store (IAS).
2.3.11 L6 · L44	**The Processor, Its Components & Machine Cycle**	Explain Control Unit, ALU, and registers; describe and trace fetch-decode-execute stages and registers used.
2.3.12 L6 · L45	**Interrupts, Polling and Parallel Processing**	Compare polling vs. interrupts and pipelining vs. parallel processing; describe interrupt detection and cycle efficiency for AI systems.
2.3.13 L6 · L46	**Assembly Instructions and Addressing Modes**	Trace assembly programs (data movement, I/O, arithmetic, jumps, compare); explain instruction formats and addressing modes (immediate, direct, indirect, register, relative, indexed).
2.3.14 L6 · L47	**Integration Activity (Architecture & Assembly)**	Fetch-execute tracing and assembly addressing exercises.
2.3.15 U6 · L89	**Synthesis: Computer Architecture & Hardware Organisation**	Analyse Von Neumann, buses, fetch-decode-execute cycle, and memory hierarchy (Cache, RAM, Registers).
2.3.16 U6 · L90	**Synthesis: ISA & Low-Level Processing**	Convert Binary, Hex, Two's Complement, IEEE 754; analyse assembly instructions, addressing modes, and CPU registers.
2.3.17 U6 · L91	**Hands-on C Project: CPU Instruction Cycle & Memory Simulation**	Simulate CPU register changes and instruction steps; trace memory access times and cache hit/miss rates (2-hour lab).
2.3.18 U6 · L92	**Hands-on Project: Assembly Code & Register Manipulation**	Write/execute low-level routines (arithmetic, loops, jumps) and debug registers/memory in an assembly simulator.

CHAPTER 2.4 · 8 LESSONS (CHRONOLOGICAL L6 → U6)

Memory Hierarchy, Storage Technologies & I/O Communication

Competency: Evaluate RAM/ROM/Cache, address-line calculations, secondary storage, peripheral interfaces, and I/O controllers.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
2.4.1 L6 · L48	**Primary Storage — Part 1**	Explain memory hierarchy (speed, capacity, cost); explain RAM, ROM, Cache functions; differentiate RAM and ROM types.
2.4.2 L6 · L49	**Primary Storage — Part 2**	State RAM, ROM, Cache applications; differentiate SRAM vs. DRAM; explain how memory hierarchy affects AI performance.
2.4.3 L6 · L50	**Memory Addressing Concepts**	Explain memory addressing; perform memory capacity calculations; calculate memory locations from address lines.
2.4.4 L6 · L51	**Secondary Storage Technologies**	Describe HDD, SSD, and optical media; classify storage technologies and analyse performance factors.
2.4.5 L6 · L52	**Integration Activity (Memory Management)**	Propose and justify a storage arrangement for an AI application.
2.4.6 L6 · L53	**Peripheral Devices and Interfaces**	Analyse peripheral performance characteristics; explain interfaces and communication methods for data exchange.
2.4.7 L6 · L54	**The I/O Controller**	Explain I/O controller roles; describe polling and interrupt-driven I/O operations.
2.4.8 L6 · L55	**Integration Activity (Peripheral I/O)**	I/O controller and peripheral interface scenarios.

MODULE 03 · 3 CHAPTERS · 27 LESSONS

Operating Systems & System Software

Consolidates system software, utilities, OS evolution and interfaces, process states, pre-emptive/non-pre-emptive CPU scheduling, synchronisation, deadlock, contiguous/paged/virtual memory, I/O transfer mechanisms, and file systems.

CHAPTER 3.1 · 7 LESSONS (CHRONOLOGICAL L6 → U6)

Software Classification, System Utilities & OS Interfaces

Competency: Classify system/application software, describe utilities and linking loaders, and compare OS types and interfaces.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
3.1.1 U6 · L37	**Classification of Software**	Identify good software attributes; classify system vs. application software; evaluate software acquisition and suitability.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
3.1.2 U6 · L38	**Introduction to System Software**	Explain system software purpose; differentiate system vs. application software and classify major categories.
3.1.3 U6 · L39	**System and Utility Software**	Describe device drivers, file/transfer utilities, linking loaders, subroutine libraries, and system resources for AI apps.
3.1.4 U6 · L40	**Types and Evolution of Operating Systems**	Describe OS evolution from batch to modern systems; relate OS types to computing and AI requirements.
3.1.5 U6 · L41	**Operating System Interfaces**	Compare OS interfaces (CLI, GUI, etc.), select by context, and identify how interfaces support AI-enabled applications.
3.1.6 U6 · L42	**Operating System Installation and File Storage**	Explain OS installation and file-storage methods; explain how storage supports AI datasets.
3.1.7 U6 · L43	**Integration Activity (System Software)**	OS selection, installation, and utility configuration review.

CHAPTER 3.2 · 9 LESSONS (CHRONOLOGICAL L6 → U6)

Process Management, CPU Scheduling, Synchronisation & Deadlock

Competency: Apply FCFS, SJF, SRTF, Round Robin, locks/semaphores, and RAG deadlock strategies, including OS scheduling simulation.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
3.2.1 U6 · L44	**Process Management Basics**	Distinguish process vs. program; illustrate process states/transitions; explain creation, termination, context switching, and call stacks.
3.2.2 U6 · L45	**Process Scheduling**	Explain scheduler and dispatcher roles; distinguish pre-emptive vs. non-pre-emptive scheduling and evaluation metrics.
3.2.3 U6 · L46	**Non-Preemptive Scheduling**	Apply FCFS, SJF, non-pre-emptive Priority; construct Gantt charts, calculate metrics, and explain starvation and ageing.
3.2.4 U6 · L47	**Pre-emptive Scheduling**	Apply SRTF, pre-emptive Priority, and Round Robin with varying time quanta; construct Gantt charts and calculate metrics.
3.2.5 U6 · L48	**Process Synchronisation**	Explain critical section, race condition, mutual exclusion; apply locks, semaphores, and monitors to concurrent processes.
3.2.6 U6 · L49	**Deadlock Handling**	Explain deadlock conditions; sketch Resource-Allocation Graphs (RAG); distinguish prevention, avoidance, detection, and recovery.
3.2.7 U6 · L50	**Integration Activity (Process Management)**	CPU scheduling, synchronisation, and deadlock exercises.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
3.2.8 U6 · L94	**Synthesis: Operating Systems & System Software**	Solve FCFS, SJF, Round Robin with Gantt charts; evaluate First-fit, Best-fit, Paging, and fragmentation.
3.2.9 U6 · L96	**Hands-on OS Project: CPU & Memory Allocation Simulator**	Build a spreadsheet/programmatic simulator for FCFS/Round Robin scheduling and First-fit vs. Best-fit allocation.

CHAPTER 3.3 · 11 LESSONS (CHRONOLOGICAL L6 → U6)

Memory Management, Virtual Memory, I/O & File Systems

Competency: Apply placement policies, paging/segmentation, demand paging, DMA/interrupt I/O, and compare FAT, NTFS, and ext file systems.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
3.3.1 U6 · L51	**Memory Management Basics**	Differentiate logical vs. physical addresses; describe process loading, address binding, and swapping.
3.3.2 U6 · L52	**Contiguous Memory Allocation**	Describe fixed/variable partitioning, internal/external fragmentation; apply first-fit, best-fit, and worst-fit policies.
3.3.3 U6 · L53	**Non-Contiguous Memory Allocation**	Describe and compare paging and segmentation as non-contiguous memory allocation schemes.
3.3.4 U6 · L54	**Virtual Memory and Demand Paging**	Distinguish virtual vs. physical memory; describe demand paging and how the OS handles page faults.
3.3.5 U6 · L55	**Integration Activity (OS Memory Management)**	Memory allocation and paging simulation exercises.
3.3.6 U6 · L56	**I/O Device Management**	Explain OS I/O management; describe device drivers, buffering, and spooling; coordinate apps, devices, and CPU.
3.3.7 U6 · L57	**I/O Transfer Mechanisms**	Compare programmed I/O (polling), interrupt-driven I/O, and DMA in terms of CPU involvement and efficiency.
3.3.8 U6 · L58	**File Systems & Management**	Describe file/directory organisation, attributes, operations; compare FAT, NTFS, and ext file systems.
3.3.9 U6 · L59	**Mass Storage and File Access Methods**	Explain mass-storage characteristics; create/manage files; differentiate sequential vs. direct access.
3.3.10 U6 · L60	**OS Security and Error Management**	Describe OS security strategy, error types/sources, and OS error-management strategies.
3.3.11 U6 · L61	**Integration Activity (I/O & File Systems)**	I/O transfer and file system comparison lab.

MODULE 04 · 3 CHAPTERS · 39 LESSONS

Data Structures, Algorithms & Complexity

Complete algorithmic foundations: design strategies, control structures, subroutines, recursion, primitive types, arrays, records, linked lists, hash tables, stacks/queues/BSTs, RPN, searching/sorting, Big-O, P/NP, and computability.

CHAPTER 4.1 · 18 LESSONS (CHRONOLOGICAL L6 → U6)

Algorithm Design Strategies, Control Structures, Subroutines & Recursion

Competency: Design algorithms using stepwise refinement, sequence/selection/loops, parameter passing, call stacks, and recursion.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.1.1 L6 · L56	**Algorithm Design Approaches 1**	Explain algorithm design strategies (brute force, divide-and-conquer, stepwise refinement, greedy).
4.1.2 L6 · L57	**Algorithm Design Approaches 2**	Differentiate top-down vs. bottom-up design; apply stepwise refinement to develop an algorithm.
4.1.3 L6 · L58	**Representation of Algorithms — Part 1**	Describe forms/characteristics of good algorithms; use control constructs to determine actions under conditions.
4.1.4 L6 · L59	**Representation of Algorithms — Part 2**	Interpret and convert between algorithmic representations; represent a simple AI decision/classification process.
4.1.5 L6 · L60	**Algorithm Design Strategies**	Identify strategies used in existing algorithms; select an appropriate strategy for a given problem.
4.1.6 L6 · L61	**Sequence and Selection Structures 1**	Explain need for control structures; explain sequence and selection constructs.
4.1.7 L6 · L62	**Sequence and Selection Structures 2**	Define dry running and trace tables; design and trace selection algorithms including rule-based AI decisions.
4.1.8 L6 · L63	**Sequence and Selection Structures 3 (Multiple Selection)**	Explain multiple selection; design and trace algorithms using nested IF and CASE constructs.
4.1.9 L6 · L64	**Repetition Structures 1a (FOR Loop)**	Explain repetition types and count-controlled FOR loop syntax/execution; construct and trace FOR loop algorithms.
4.1.10 L6 · L65	**Repetition Structures — Part 1 (WHILE & REPEAT)**	Distinguish pre-test vs. post-test condition-controlled loops; describe WHILE and REPEAT...UNTIL syntax.
4.1.11 L6 · L66	**Repetition Structures — Part 2 (Loop Tracing & AI)**	Construct and trace WHILE/REPEAT...UNTIL algorithms; develop a loop-based dataset processing algorithm for AI.
4.1.12 L6 · L67	**Subroutines**	Explain subroutines; differentiate procedures vs. functions; demonstrate subroutine calls in algorithms.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.1.13 L6 · L68	**Parameter Passing**	Explain parameters; differentiate formal vs. actual parameters and pass-by-value vs. pass-by-reference.
4.1.14 L6 · L69	**Processing Subroutine Calls 1 (Call Stack)**	Explain how the processor executes subroutine calls using the stack for return addresses; modularise AI data tasks.
4.1.15 L6 · L70	**Processing Subroutine Calls 2 (Trace & Recursion Intro)**	Trace call/return sequences; explain the concept and characteristics of recursion.
4.1.16 L6 · L71	**Recursion**	Define recursion; trace recursive algorithms; develop simple recursive solutions; explain OS stack usage.
4.1.17 L6 · L72	**Recursion Limitations and Complexity**	Differentiate iteration vs. recursion; dry-run recursive algorithms; assess runtime space limits and compare for AI tasks.
4.1.18 L6 · L73	**Integration Activity (Modular Algorithms)**	Modular algorithm design, recursion, and trace-table lab.

CHAPTER 4.2 · 14 LESSONS (CHRONOLOGICAL L6 → U6)

Data Structures, Linked Lists, Hash Tables & ADTs (Stacks, Queues, BSTs)

Competency: Manipulate arrays, records, linked lists, hash tables, stacks (RPN), queues, and Binary Search Trees.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.2.1 L6 · L93	**Primitive Data Types**	Explain primitive data types and applications; explain the need for data structures in managing collections.
4.2.2 L6 · L94	**Arrays**	Distinguish 1D and 2D arrays; represent and manipulate data using arrays for AI dataset inputs.
4.2.3 L6 · L95	**Records**	Explain records; differentiate arrays vs. 2D records; represent and manipulate heterogeneous data using records.
4.2.4 L6 · L96	**Data Types, Memory Allocation & Data Structures**	Calculate memory allocated per data type; declare char-array strings, records, lists/enums; compare pointer-driven vs. fixed structures.
4.2.5 L6 · L97	**Linked Lists**	Explain linked lists; represent and manipulate linked lists in code; explain dynamic data management for AI datasets.
4.2.6 L6 · L98	**Hash Tables**	Explain hash tables; handle collisions via linear probing and separate chaining with linked lists.
4.2.7 L6 · L99	**Using Repetition Structures to Process Arrays & Lists**	Develop loop algorithms to process arrays, validate input, generate tables, and print patterns.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.2.8 L6 · L100	**Integration Activity (Data Structures)**	Design an efficient data-structure solution for a simple AI application.
4.2.9 U6 · L6	**Abstract Data Types (ADTs)**	Differentiate ADTs vs. data structures; explain stack and queue ADTs, operations, and applications including AI.
4.2.10 U6 · L7	**Binary Tree Concepts and Representation**	Represent binary trees; perform preorder/inorder/postorder traversals; perform search, insert, delete on BSTs; relate trees to AI decisions.
4.2.11 U6 · L8	**Applications of ADTs**	Evaluate postfix (RPN) and convert infix/prefix/postfix using stacks and expression trees; relate stacks/queues to AI data processing.
4.2.12 U6 · L9	**Integration Activity (ADTs & Trees)**	ADT, stack expression, and BST implementation exercises.
4.2.13 U6 · L93	**Synthesis: Data Structures & C Programming Foundations**	Trace search/sort algorithms; declare and manipulate stacks, queues, linked lists, and binary trees dynamically in C.
4.2.14 U6 · L95	**Hands-on C Project: Data Indexing & Search Module**	Implement dynamic record insertion, searching, and sorting in C using a Linked List or Binary Search Tree.

CHAPTER 4.3 · 7 LESSONS (CHRONOLOGICAL L6 → U6)

Searching, Sorting, Big-O Complexity & Computability

Competency: Trace search/sort algorithms, analyse time/space complexity in Big-O, and distinguish P/NP, decidable/undecidable, and Turing machines.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.3.1 L6 · L101	**Complexity of Algorithms**	Explain time and space complexity; estimate resource requirements by counting operations/memory; compare algorithms.
4.3.2 L6 · L102	**Big-O Notation**	Express time complexity using Big-O notation; compare searching/sorting complexities; explain P and NP classes.
4.3.3 L6 · L103	**Computational Problem Types**	Distinguish tractable/intractable, decidable/undecidable (Halting, TSP), computable/non-computable, Turing machines, heuristics, and AI difficulty.
4.3.4 L6 · L104	**Algorithm Complexity, Dry Runs and Optimisation**	Determine complexity from operation counts and dry runs; optimise algorithms to improve step efficiency.
4.3.5 L6 · L105	**Integration Activity (Algorithm Efficiency)**	Big-O analysis and algorithm optimisation lab.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
4.3.6 U6 · L10	**Searching Algorithms**	Trace linear and binary search; compare efficiency; explain how search techniques explore AI problem spaces.
4.3.7 U6 · L11	**Sorting Algorithms**	Trace bubble, selection, and insertion sort; compare efficiency and suitability; explain how sorting prepares AI data.

MODULE 05 · 3 CHAPTERS · 39 LESSONS

Programming Paradigms, C/Python Coding & Software Engineering

End-to-end software construction: programming paradigms (OOP, declarative, procedural), translators, debugging, C and Python programming from basics to structs/subroutines, SDLC models, requirements, design, quality, V&V, and FSM/automata.

CHAPTER 5.1 · 6 LESSONS (CHRONOLOGICAL L6 → U6)

Programming Languages, Paradigms, Translators & Debugging

Competency: Compare imperative/OOP vs. declarative paradigms, assemblers/compilers/interpreters, and debugging tools.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
5.1.1 L6 · L74	**Programming Language Basics**	Explain role of programming languages; categorise and compare low-level vs. high-level languages.
5.1.2 L6 · L75	**Programming Paradigms 1**	Describe and compare imperative (procedural, OOP) vs. declarative (logic, functional, SQL) paradigms for AI/software tasks.
5.1.3 L6 · L76	**Programming Paradigms 2 (OOP)**	Explain OOP concepts (class, object, attributes, methods) and characteristics (encapsulation, abstraction, inheritance, polymorphism, coupling, cohesion).
5.1.4 L6 · L77	**Language Translators**	Describe assembler, compiler, and interpreter; explain main stages of program translation and compilation.
5.1.5 L6 · L78	**Program Errors and Debugging**	Explain syntax, semantic, and runtime errors; use debuggers, breakpoints, watchpoints, and step execution on AI data programs.
5.1.6 L6 · L79	**Integration Activity (Language Concepts)**	Paradigms, translation, and debugging exercises.

CHAPTER 5.2 · 23 LESSONS (CHRONOLOGICAL L6 → U6)

Programming in C & Python

Competency: Write, test, and debug C and Python programs using variables, selection, loops, strings,

1D/2D arrays, structs, and subroutines.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
5.2.1 L6 · L80	**Introduction to C & Python Programming — Part 1**	Describe C and Python features/applications; explain basic syntax and program structure.
5.2.2 L6 · L81	**Introduction to C & Python Programming — Part 2**	Write simple C/Python programs using print and variables; format input data for an AI application.
5.2.3 L6 · L82	**Introduction to C & Python Programming — Part 3**	Identify and use C and Python data types (integer, float, string, Boolean).
5.2.4 L6 · L83	**Python/C Operators and Expressions — Part 1**	Identify common C/Python operators; explain expression evaluation and construct expressions.
5.2.5 L6 · L84	**Python/C Operators and Expressions — Part 2 (I/O)**	Use input() in Python and scanf() in C; combine I/O to collect, validate, and display data for an AI system.
5.2.6 L6 · L85	**Python/C Selection Structures — Part 1**	Explain selection in C/Python; construct programs using simple and multiple selection.
5.2.7 L6 · L86	**Python/C Selection Structures — Part 2**	Apply selection structures to solve computational problems in C and Python.
5.2.8 L6 · L87	**Python/C Iterative Structures**	Construct C/Python programs using for and while loops; process datasets for repeated AI computations.
5.2.9 L6 · L88	**Writing and Debugging C Programs**	Apply algorithmic design in C to solve a problem; debug syntax and logic errors.
5.2.10 L6 · L89	**Writing and Debugging Python Programs**	Apply algorithmic design in Python to solve a problem; debug syntax and logic errors.
5.2.11 L6 · L90	**Python Program Development & Debugging Project — Part 1**	Apply algorithmic design to develop a complete Python solution to a contextual problem.
5.2.12 L6 · L91	**Python Program Development & Debugging Project — Part 2**	Test, debug systematically, and present a Python solution involving data or a simple AI feature.
5.2.13 L6 · L92	**Integration Activity (C & Python Programming)**	Practical C and Python coding synthesis.
5.2.14 U6 · L18	**Setting Up a C Programming Environment**	Install compiler/IDE; explain C compile-run cycle and components; write/run Hello World; relate tools to AI programs.
5.2.15 U6 · L19	**Implementing Basic Instructions in C**	Declare variables/constants, format I/O safely, evaluate arithmetic/relational/logical expressions for data tasks.
5.2.16 U6 · L20	**Implementing Selection in C**	Write decision programs, handle multi-condition and nested selection, trace variables, and implement simple AI decision rules.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
5.2.17 U6 · L21	**Implementing Iteration in C**	Differentiate pre-test, post-test, and counting loops; choose efficient loops, trace variables, and process data.
5.2.18 U6 · L22	**Strings & Character Handling in C**	Use string.h and ctype functions (strcmp, strcpy, strlen, strcat); manipulate strings and text data for AI.
5.2.19 U6 · L23	**Implementing 1D Arrays with Loops in C**	Declare/initialise 1D arrays; compute sum/average/min/max with loops; store/process data for AI systems.
5.2.20 U6 · L24	**Implementing 2D Arrays with Loops in C**	Declare/initialise 2D arrays; use nested loops for matrix addition/multiplication and 2D AI data representation.
5.2.21 U6 · L25	**Implementing Records and Lists in C**	Create structs, manipulate arrays of records, declare lists, and organise structured input for AI applications.
5.2.22 U6 · L26	**Implementing Subroutines in C**	Write/call functions, use pass-by-value and pass-by-reference, write recursive functions, and design reusable AI helpers.
5.2.23 U6 · L27	**Integration Activity (C Programming)**	Comprehensive C programming lab.

CHAPTER 5.3 · 10 LESSONS (CHRONOLOGICAL L6 → U6)

Software Engineering, SDLC Models, Testing & Automata

Competency: Evaluate SDLC models, SRS, modular design, CASE/VCS tools, V&V testing, PERT/Gantt, maintenance, and Finite State Machines.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
5.3.1 U6 · L28	**Software Development Activities and Models**	Describe SDLC activities; compare development models (pros/cons, applicability); identify where AI tools support SDLC.
5.3.2 U6 · L29	**Requirements Analysis and Specification**	Outline elicitation, validation, specification, verification; write an SRS; compare technical vs. user requirements and AI requirements.
5.3.3 U6 · L30	**Software Design — Stages and Documentation**	Apply modular/data/process/interface design; analyse cohesion, coupling, reuse, APIs; produce design docs with AI components.
5.3.4 U6 · L31	**Software Development Tools**	Differentiate CASE tools, IDEs, and version control; explain software quality characteristics and AI-assisted dev tools.
5.3.5 U6 · L32	**Software Quality, Acquisition and Evaluation**	Evaluate software quality, acquisition costs/legal/security, suitability/limitations, and accuracy/reliability of AI software.
5.3.6 U6 · L33	**Software Verification, Validation and Testing**	Explain V&V, code review, static analysis; differentiate testing methods; identify tests for AI-supported features.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
5.3.7 U6 · L34	**Software Project Management and Reporting**	Explain proposal writing, planning, monitoring, staffing, and reporting; produce a software management report.
5.3.8 U6 · L35	**Software Reuse, Teamwork and Maintenance**	Explain reuse, teamwork, user feedback, debugging, corrective/preventive maintenance, and project documentation.
5.3.9 U6 · L36	**Integration Activity (Software Engineering)**	Software development lifecycle and design case study.
5.3.10 U6 · L102	**Synthesis: Software Engineering, SDLC & Theoretical CS**	Compare Waterfall/Agile, testing, PERT/Gantt (critical path), and analyse Finite State Machines (FSM) and automata.

MODULE 06 · 3 CHAPTERS · 31 LESSONS

Information Systems, Databases & SQL

Unifies organisational information systems, data capture/validation, SDLC/DFD/UML/GIS, database models, 3-level architecture, ER modelling, 1NF–BCNF normalisation, ACID/BASE, SQL (DDL/DML/DQL/Joins/Aggregates), RDBMS packages, and PHP/HTML web-DB integration.

CHAPTER 6.1 · 9 LESSONS (CHRONOLOGICAL L6 → U6)

Information Systems, Organisational Levels, SDLC & GIS

Competency: Evaluate organisational IS, data capture/validation, UML/DFD system design, GIS layers, and IS maintenance.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
6.1.1 U6 · L62	**Introduction to Information Systems**	Define IS, components, good information characteristics; define a community IS problem and illustrate management info flow.
6.1.2 U6 · L63	**Data Capture**	Define data capture vs. data processing; describe common data capture methods and applications.
6.1.3 U6 · L64	**Data Verification and Validation**	Define and apply data verification and validation methods to ensure data accuracy and integrity.
6.1.4 U6 · L65	**Organisational Levels and Information Systems**	Identify organisational levels, people, information needs, and corresponding information systems at each level.
6.1.5 U6 · L66	**Integration Activity (Information Systems)**	Organisational IS and data validation case study.
6.1.6 U6 · L67	**System Development Process**	Identify SDLC stages, activities, and deliverables; transform an existing system into a computerised community IS.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
6.1.7 U6 · L68	System Design Tools (UML, DFD, GIS)	Interpret UML and DFDs; use pseudocode/structured English; build a library/school MIS DFD; explore GIS layers and queries.
6.1.8 U6 · L69	System Testing	Explain testing purpose; describe test data and system testing types; select test data for system functionality.
6.1.9 U6 · L70	Information System Management	Describe IS maintenance, backup, security, documentation, and user support procedures.

CHAPTER 6.2 · 12 LESSONS (CHRONOLOGICAL L6 → U6)

Database Models, ER Modelling, Normalisation (1NF–BCNF) & Integrity

Competency: Design ERDs, normalise schemas through 3NF/BCNF, enforce integrity constraints, and explain ACID/BASE.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
6.2.1 U6 · L71	Database Fundamentals and Data Organisation	Explain database purpose, DBMS functions, and pros/cons of computerised vs. non-computerised databases.
6.2.2 U6 · L72	Database Models	Compare hierarchical, network, relational, and object-oriented models; outline multi-file advantages over flat-file systems.
6.2.3 U6 · L73	Relational Database Models	Explain relational structure, flat-file problems, primary/candidate/foreign keys; organise and validate data.
6.2.4 U6 · L74	Database Architecture (3-Level ANSI-SPARC)	Describe external, conceptual, and internal levels and how they provide data abstraction and independence.
6.2.5 U6 · L75	Entity-Relationship Modelling	Explain E-R components, degree, and cardinality; construct and interpret E-R diagrams for database scenarios.
6.2.6 U6 · L76	Relational Database Design and Relationships	Organise data into related tables, choose data types, set up joins, enforce constraints, and handle modification anomalies.
6.2.7 U6 · L77	Database Normalisation & Functional Dependencies	Explain insertion/update/deletion anomalies, functional dependencies, and normalise tables through 1NF, 2NF, 3NF, and BCNF.
6.2.8 U6 · L78	Database Design Stages and Data Modelling	Apply conceptual, logical, normalisation, and physical stages; decompose M:N relationships; build a data dictionary and ERD.
6.2.9 U6 · L79	Database Integrity & Transactions	Describe entity, referential, and domain/check constraints; describe ACID and BASE transaction approaches.
6.2.10 U6 · L80	Integration Activity (Database Modelling)	ERD design, 3NF/BCNF normalisation, and integrity lab.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
6.2.11 U6 · L97	**Synthesis: Relational Database Modelling & Normalisation**	Construct enterprise ERDs and normalise relational tables from 1NF through 3NF/BCNF.
6.2.12 U6 · L99	**Hands-on Database Project: ERD & Normalisation Case Study**	Analyse a real-world case (e.g., Hospital/School) and produce a normalised ERD, relational schema, and data dictionary.

CHAPTER 6.3 · 10 LESSONS (CHRONOLOGICAL L6 → U6)

SQL, RDBMS Implementation & Web-Database Integration

Competency: Write DDL, DML, DQL, JOIN, and aggregate queries, build Access/Base apps, and connect SQL databases to HTML/PHP forms.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
6.3.1 U6 · L81	**SQL — Data Definition Language (DDL)**	Explain DDL; use CREATE, ALTER, and DROP with data types, keys, and integrity constraints.
6.3.2 U6 · L82	**SQL — Data Manipulation Language (DML)**	Explain DML; use INSERT, UPDATE, and DELETE with appropriate conditions.
6.3.3 U6 · L83	**SQL — Data Query Language (DQL)**	Use SELECT, WHERE, logical/comparison operators, and ORDER BY to filter and sort data.
6.3.4 U6 · L84	**SQL — Joins**	Explain SQL join types and construct multi-table JOIN queries.
6.3.5 U6 · L85	**SQL — Aggregate Functions**	Use COUNT(), SUM(), AVG(), MIN(), MAX() with GROUP BY and HAVING for summary results.
6.3.6 U6 · L86	**RDBMS Packages and Database Implementation**	Use MS Access/Base to create tables, forms, reports, import/export, and run SQL queries, joins, and subqueries.
6.3.7 U6 · L87	**Web and Object-Oriented Database Integration**	Use PHP/HTML to connect a database-driven website; link an OO database to store/retrieve data via web.
6.3.8 U6 · L88	**Integration Activity (SQL & RDBMS)**	End-to-end SQL and web-database practical.
6.3.9 U6 · L98	**Synthesis: Database Querying & Information Retrieval with SQL**	Write DDL/DML and complex DQL queries using inner/outer JOINs, GROUP BY, HAVING, and aggregates.
6.3.10 U6 · L100	**Hands-on Database Project: SQL Creation & Web Form Binding**	Execute SQL DDL scripts, build core queries, and test data insertion via an HTML/PHP form template.

MODULE 07 · 3 CHAPTERS · 19 LESSONS

Computer Networks, Cybersecurity & Web Development

Consolidates network topologies, media, OSI/TCP-IP models, IPv4 addressing, CIDR/subnetting, data communication modes, error-detecting codes (Hamming/CRC), network security, safe online services, and HTML/CSS web authoring.

CHAPTER 7.1 · 10 LESSONS (CHRONOLOGICAL L6 → U6)

Computer Networks, Topologies, OSI Model, IP Addressing & Subnetting

Competency: Configure LANs, compare topologies/media, map protocols to OSI layers, and implement IPv4 CIDR/subnetting.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
7.1.1 L6 · L106	**Introduction to Computer Networks — Part 1**	Define networks; identify device roles and geographic scopes; explain how connectivity enables AI and cloud services.
7.1.2 L6 · L107	**Introduction to Computer Networks — Part 2**	Distinguish peer-to-peer vs. client-server; describe star, bus, ring, mesh topologies and guided/unguided media.
7.1.3 L6 · L108	**Introduction to Computer Networks — Part 4 (Transmission Media)**	Compare common transmission media and select suitable media for given applications.
7.1.4 L6 · L109	**Network Connections, Architectures & Hardware**	Describe point-to-point/multipoint, NOS features, architectures; select/connect switches, routers, modems for AI/cloud data.
7.1.5 L6 · L110	**The OSI Model**	Explain communication protocols and standards; describe the 7 OSI layers and map protocols to layers.
7.1.6 L6 · L111	**IP Addressing**	Explain IP address structure, classes, types, CIDR, and subnetting; configure OS networking and explain mobile communications.
7.1.7 L6 · L112	**Network Configuration and Troubleshooting**	Compare wired/wireless transmission; outline network errors; set up and troubleshoot networks for AI/cloud connectivity.
7.1.8 L6 · L113	**Data Communication Modes and Transmission**	Describe simplex/half-duplex/duplex, serial/parallel, sync/async, mux/demux, baseband/broadband, analogue/digital signals.
7.1.9 U6 · L101	**Synthesis: Data Communication, Networks & Security**	Distinguish OSI/TCP-IP layers, IPv4 subnetting/CIDR, routing, firewalls, encryption, and attack prevention.
7.1.10 U6 · L103	**Hands-on Networking Lab: Subnetting & LAN Simulation**	Implement IP subnetting, configure IP/mask/gateway on a virtual network, and test with ping/traceroute.

CHAPTER 7.2 · 4 LESSONS (CHRONOLOGICAL L6 → U6)

Network Security, Internet Standards & Safe Online Services

Competency: Apply error-detecting codes (Hamming/CRC), firewalls, encryption, compare OSI vs. TCP/IP, and use internet services safely.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
7.2.1 L6 · L114	**Network and Data Security**	Describe error detection (parity, Hamming, CRC); protect systems with antivirus, firewalls, encryption, passwords, backup, and cloud storage.
7.2.2 L6 · L115	**Network Standards, Protocols & Internet Development**	Compare OSI vs. TCP/IP models; discuss TCP, UDP, IP, FTP; explain Internet history, intranet, extranet, and AI/cloud protocols.
7.2.3 L6 · L116	**Internet Services and Safe Online Use**	Use e-commerce, e-learning, e-banking, email, and search safely; manage privacy settings and evaluate AI-enabled Internet risks.
7.2.4 L6 · L117	**Integration Activity (Networking & Security)**	Contextual networking, troubleshooting, and security lab.

CHAPTER 7.3 · 5 LESSONS (CHRONOLOGICAL L6 → U6)

Web Development Technologies (HTML & CSS)

Competency: Evaluate online information literacy and build structured, styled webpages using HTML and CSS.

LESSON	LESSON TITLE	LEARNING OBJECTIVES
7.3.1 L6 · L118	**Web Technology, Connectivity & Information Literacy**	Explain browsers, search engines, email, markup languages; apply online and assess online information for bias and reliability.
7.3.2 L6 · L119	**Introduction to HTML 1**	Introduce HTML document structure; use basic tags for text, headings, lists, links, and images.
7.3.3 L6 · L120	**Introduction to HTML 2**	Use HTML tags to create tables; assess AI-generated/assisted online information for bias, validity, and reliability.
7.3.4 L6 · L121	**Introduction to CSS 1**	Explain CSS role in styling; apply CSS to style text, colours, and layout; combine HTML and CSS into a structured webpage.
7.3.5 L6 · L122	**Integration Activity (Web Development)**	Structured HTML and CSS webpage project.

SOURCE DOCUMENTS

Official Progression Sheets

1. Republic of Cameroon (MINESEC) — *National Harmonised Progression Sheet for Computer Science, Lower Sixth* (122 lessons).
2. Republic of Cameroon (MINESEC) — *National Harmonised Progression Sheet for Computer Science, Upper Sixth* (103 lessons).