AC 28/5/2026/Sfc/02 ITEM NO: 1.4.2

Deccan Education Society's

Kirti M. Doongursee College of Arts, Science and Commerce (AUTONOMOUS)

Dadar (W), Mumbai -28

Affiliated to

UNIVERSITY OF MUMBAI

Title of the program

A- U.G. Certificate in Computer Science

B- U.G. Diploma in Computer Science

C- **B.Sc** (Computer Science)

Syllabus for

Semester – Sem I & II

Ref: GR dated 20th April, 2023 for Credit Structure of UG

(With effect from the academic year 2026-27 Progressively)

PROGRAM OUTCOMES

PO	Description
A student completing Bachelor's Degree in **Science** Program will be able to	
PO1	Disciplinary Knowledge: Demonstrate comprehensive knowledge of the disciplines that form a part of a graduate Programme. Execute strong theoretical and practical understanding generated from the specific graduate Programme in the area of work.
PO2	Critical Thinking and Problem solving: Exhibit the skills of analysis, inference, interpretation and problem-solving by observing the situation closely and design the solutions
PO3	Social competence: Display the understanding, behavioral skills needed for successful social adaptation, work in groups, exhibit thoughts and ideas effectively in writing and orally.
PO4	Research-related skills and Scientific temper: Develop the working knowledge and applications of instrumentation and laboratory techniques. Able to apply skills to design and conduct independent experiments, interpret, establish hypothesis and inquisitiveness towards research
PO5	Trans-disciplinary knowledge: Integrate different disciplines to uplift the domains of cognitive abilities and transcend beyond discipline-specific approaches to address a common problem.
PO6	Personal and professional competence: Performing dependently and collaboratively as a part of a team to meet defined objectives and carry out work across interdisciplinary fields. Execute interpersonal relationships, self-motivation and adaptability skills and commit to professional ethics.
PO7	Effective Citizenship and Ethics: Demonstrate empathetic social concern and equity centered national development, and ability to act with an informed awareness of moral and ethical issues and commit to professional ethics and responsibility.

PO8	Environment and Sustainability: Understand the impact of the scientific solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.

PROGRAM SPECIFIC OUTCOMES

PSO	Description
PSO1	**Core Knowledge** – Build strong foundations in programming, algorithms, databases, and operating systems
PSO 2	**Software Development** – Design and implement reliable software using modern tools and languages.
PSO 3	**Problem-Solving** – Apply computational thinking to analyze, debug, and optimize solutions
PSO 4	**Emerging Tech** – Gain exposure to AI, ML, cybersecurity, and data science.
PSO 5	**Professional Skills –** Develop teamwork, communication, and ethical responsibility for IT careers**.**
PSO 6	**Lifelong Learning –** Prepare for higher studies and adapt to evolving technologies.

PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8
PSO1: Core Knowledge & Software Development – Build strong foundations in programming, algorithms, databases, operating systems, and develop reliable software using modern tools.	3	3	1	2	2	2	1	1
PSO2: Problem-Solving & Research Skills – Apply computational thinking, critical analysis, debugging, and research skills to design and optimize solutions.	2	3	1	3	2	2	1	2
PSO3: Emerging Technologies & Lifelong Learning – Gain knowledge of AI, ML, cybersecurity, data science, and adapt to evolving technologies for higher studies and career growth.	2	3	1	3	3	2	1	3
PSO4: Professional Skills & Ethics – Develop teamwork, communication, leadership, ethical responsibility, and social awareness for professional success.	1	2	3	1	2	3	3	2

Mapping Scale:
3 – High | 2 – Moderate | 1 – Low

Deccan Education Society's

Kirti M. Doongursee College of Arts, Science and Commerce (AUTONOMOUS)

Sr. No.	Heading		Particulars
1	**Title of program** **O: _____A**	**A**	**U.G. Certificate in Computer Science**
	O: _____B	**B**	**U.G. Diploma in Computer Science**
	O: _____C	**C**	**B.Sc. (Computer Science)**
2	**Eligibility** **O: _____A**	**A**	HSC(Science) **OR** Passed Equivalent Academic Level 4.0
	O: _____B	**B**	Under Graduate Certificate in **Computer Science OR** Passed Equivalent Academic Level 4.5
	O: _____C	**C**	Under Graduate Diploma in **Computer Science OR** Passed Equivalent Academic Level 5.0
3	**Duration of Program** **O: _____A**	**A**	One Year
		B	Two Years
		C	Three Years
4	**Intake Capacity** **R: _____**		**120**

Sign of the HOD & Coordinator BOS in _____	**Sign of the NEP Coordinator, DES's Kirti M. Doongursee College**	**Sign of Principal DES's Kirti M. Doongursee College, Mumbai 28.**

Preamble

1. Introduction

The Bachelor of Science (B.Sc.) in Computer Science programme is designed to equip students with the knowledge, skills, and competencies required to thrive in the rapidly evolving digital era. In a world increasingly driven by data, automation, and intelligent systems, computer science plays a pivotal role in shaping innovation, economic growth, and societal transformation at both national and global levels. The programme recognizes the growing demand for skilled professionals in domains such as software development, data science, cybersecurity, artificial intelligence, and emerging technologies.

The primary objective of this programme is to build a strong foundation in core areas of computer science, including algorithms, programming, data structures, computer systems, and networks, while fostering analytical thinking and problem-solving abilities. It aims to bridge theoretical knowledge with practical application through hands-on learning, projects, and industry exposure. Emphasis is also placed on cultivating ethical computing practices, digital responsibility, and awareness of societal impacts.

Aligned with the principles of the National Education Policy (NEP), the curriculum adopts a multidisciplinary, flexible, and outcome-based approach that encourages innovation, research orientation, and lifelong learning. The programme is structured to enhance employability, support entrepreneurial initiatives, and prepare graduates to adapt to dynamic technological landscapes. By integrating contemporary tools, industry trends, and skill-based learning, the B.Sc. Computer Science programme aspires to develop competent, responsible, and forward-thinking professionals capable of contributing meaningfully to the digital future.

2. Aims and Objectives

Aims:

- To provide a strong foundation in the fundamental principles and concepts of computer science.
- To prepare students for careers in the IT industry, research, and higher education.
- To develop analytical thinking, problem-solving abilities, and computational skills.
- To foster innovation, creativity, and an entrepreneurial mindset in the field of computing.
- To promote ethical practices and social responsibility in the use of technology.

Objectives:

- To impart comprehensive knowledge in areas such as programming, data structures, algorithms, databases, computer networks, and operating systems.
- To develop practical skills through laboratory work, projects, and industry-oriented training.
- To enable students to design, develop, and evaluate software solutions for real-world

problems.
- To expose learners to emerging technologies such as artificial intelligence, data science, cybersecurity, and cloud computing.
- To encourage research aptitude, critical thinking, and continuous learning.
- To enhance employability skills including communication, teamwork, and professionalism.
- To align the curriculum with National Education Policy (NEP) guidelines, ensuring flexibility, multidisciplinary learning, and skill-based education.

3. Learning Outcomes

Upon successful completion of the programme, students will be able to:

- Demonstrate a strong understanding of fundamental concepts in computer science, including programming, data structures, algorithms, databases, computer networks, and operating systems.
- Apply theoretical knowledge to design, develop, and evaluate efficient software solutions for real-world problems.
- Analyze complex problems using computational thinking and select appropriate tools and techniques for effective problem-solving.
- Utilize modern computing tools, platforms, and programming languages to build scalable and reliable applications.
- Demonstrate proficiency in emerging areas such as data science, artificial intelligence, cybersecurity, and cloud computing.
- Exhibit research aptitude, innovation, and the ability to explore new technologies and methodologies.
- Adhere to professional ethics, cybersecurity practices, and social responsibilities in the development and use of computing technologies.
- Communicate effectively, work collaboratively in multidisciplinary teams, and manage projects efficiently.
- Pursue higher education, research, or professional careers in computer science and related domains with confidence and competence.

Semester	Course Code	Course Title	Vertical	Credit
I	26CSMJ111	Digital System and Architecture	Major	2
	26CSMJ112	Introduction to Database Systems	Major	2
	26CSMJP111	CS practical 1 (Part A+B)	Practical	2
	26CSOE131	Introduction to E-commerce using AI	OE	2
	26CSVC141	Introduction to Programming with Python	VSC	2
	26CSSCE151 26CSSCE152	Statistics with R Or Linux Operating System	SEC	2
II	26CSMJ211	Design and Analysis of Algorithm	Major	2
	26CSMJ212	Object Oriented Programming	Major	2
	26CSMJP21	CS practical 2 (Part A+B)	Practical	2
	26CSMR221	IT Tools and Digital Productivity	Minor	2
	26CSOE231	Cyber Law and Digital Security	OE	2
	26CSVC241	Advance Python Programming	VSC	2
	26CSSCE251 26CSSCE252	Web Technologies Or Web Development using Word Press	SEC	2

SEMESTER-I

Course Code	MAJOR SEM – I	Credits	Lectures/ Week
26CSMJ111	Paper I - Digital System and Architecture	2	2

Course Objectives (CO):

CO 1: To understand the fundamentals of Logic gates, Number system and Flip Flops.
CO 2: To have an understanding of Digital Systems and Operation of a Digital Computer.
CO 3: To Learn Different Architecture & Organization of memory system, processor organization and control unit.
CO 4: Basic understanding of 8085 microprocessor and its applications.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Recall fundamental concepts of number systems, Boolean algebra, logic gates, computer organization, memory types, and instruction set basics.
OC2: Explain the working principles of combinational and sequential circuits, memory hierarchy, cache mechanisms, instruction formats, and processor organization.
OC3: Apply logic simplification techniques, design combinational and sequential circuits, and use addressing modes and instruction sets in solving problems related to microprocessor operations.
OC4: Analyze memory system performance, cache design, instruction execution, pipelining, and compare architectural approaches.

Unit	Topics	No of Lectures
I	**Fundamentals of Digital Logic:** Boolean algebra, Logic Gates, Simplification of Logic Circuits: Algebraic Simplification, Karnaugh Maps. **Combinational Circuits:** Adders, Subtractors, Multiplexer, De-Multiplexer. **Sequential Circuits:** Flip- Flops (SR, JK & D), Counters: synchronous and asynchronous Counter. **Computer System:** Comparison of Computer Organization & Architecture, Computer Components and Functions, Interconnection Structures. Bus Interconnections, Input / Output: I/O Module Programmed I/O, Interrupt Driven I/O, Direct Memory Access.	15

II	**Memory System Organization:** Classification and design parameters, Memory Hierarchy, Internal Memory: RAM, SRAM and DRAM, Interleaved and Associative Memory. Cache Memory: Design Principles, Memory mappings, Replacement Algorithms, Cache performance, Cache Coherence. Virtual Memory, External Memory: Magnetic Discs, Optical Memory, Flash Memories, RAID Levels **Instructions:** Instruction Formats, Instruction Sets, Addressing Modes, Addressing Modes Examples with Assembly Language [8085/8086 CPU]. **Processor Organization:** Structure and Function. Register Organization [8085/8086 CPU]. Basic Microprocessor operations: Data Transfer (Register / Memory) Operations, Arithmetic & Logical Operations. Instruction Cycle, Instruction Pipelining. Introduction to RISC and CISC Architecture, Instruction Level Parallelism and Superscalar Processors, Design Issues.	**15**

Textbooks:

- M. Mano, Computer System Architecture 3rd edition, Pearson
- Carl Hamacher et al., Computer Organization and Embedded Systems, 6 ed., McGraw-Hill 2012
- R P Jain, Modern Digital Electronics, Tata McGraw Hill Education Pvt. Ltd. , 4th Edition, 2010

Additional References:

- William Stallings (2010), Computer Organization and Architecture-designing for performance, 8th edition, Prentice Hall, New Jersy.
- Anrew S. Tanenbaum (2006), Structured Computer Organization, 5th edition, PearsonEducation Inc,
- John P. Hayes (1998), Computer Architecture and Organization, 3rd edition, Tata McGrawHill
- Ramesh Gaonkar (2013), Microprocessor Architecture, Programming and Application with 8085, 6th edition, Penram.

Course Code	MAJOR SEM – I	Credits	Lectures/ Week
26CSMJ112	**Paper II - Introduction to Database Systems**	**2**	**2**

Course Objectives (CO):

CO1: Fundamental Principles: To provide a solid foundation in database concepts, architecture, and the importance of data abstraction and independence.

CO2: Design Methodology: To teach systematic database design using Entity-Relationship (ER) modeling and the principles of normalization to ensure data integrity.

CO3: Technical Proficiency: To develop expertise in writing efficient SQL queries (DDL, DML, DCL) and managing complex data retrievals through joins and subqueries.

CO4: Data Administration: To introduce the concepts of database security, view management, and transaction control to prepare students for real-world DBA responsibilities.

Course Outcomes (OC):

After successful completion of this course, students would be able to

OC1: Explain the fundamental concepts of DBMS architecture, data models (relational, hierarchical, network), and the levels of data abstraction.

OC2: Construct efficient SQL queries using DDL, DML, and built-in functions to manipulate data and enforce integrity constraints.

OC3: Analyze complex business requirements to design Entity-Relationship diagrams and transform them into optimized relational table structures.

OC4: Evaluate database designs for redundancy and anomalies using Normal Forms ($1NF, 2NF, 3NF, BCNF$) and implement security measures via DCL and views.

Unit	Topics	No of Lectures
I	**Introduction to DBMS:** Database, DBMS – Definition, Overview of DBMS, Advantages of DBMS, Levels of abstraction, Data independence, DBMS Architecture **Data models**: Client/Server Architecture, Object Based Logical Model, Record Based Logical Model (relational,hierarchical, network) **Entity Relationship Model and ER to Table:** Entities, attributes, entity sets, relations, relationship sets, Additional constraints(key constraints, participation constraints, weak entities, aggregation / generalization, Conceptual Design using ER (entities VS attributes, Entity Vs relationship, binary Vs ternary, constraints beyond	**15**

	ER) Entity to Table,Relationship to tables with and without key constraints. **DDL Statements**: Creating Databases, Using Databases, datatypes, Creating Tables (with integrity constraints – primary key, default, check, not null), Altering Tables, Renaming Tables, Dropping Tables, Truncating Tables **DML statements**: Viewing the structure of a table insert, update, delete, Select all columns, specific columns, unique records, conditional select, in clause, between clause, limit, aggregate functions (count, min, max, avg, sum), group by clause, having clause	
II	**Relational data model**: Domains, attributes, Tuples and Relations, Relational Model Notation, Characteristics of Relations, Relational Constraints - primary key, referential integrity, unique constraint, Null constraint, Check constraint **Functions:** String Functions (concat, instr, left, right, mid, length, lcase/lower, ucase/upper, replace, strcmp, trim, ltrim, rtrim), Math Functions (abs, ceil, floor, mod, pow, sqrt, round, truncate) Date Functions(adddate, datediff, day, month, year, hour, min, sec, now, reverse) **Joining Tables and Subqueries**: inner join, outer join (left outer, right outer, full outer) subqueries with IN, EXISTS, subqueries restrictions, Nested subqueries, ANY/ALL clause, correlated subqueries. **Normal forms:** Functional dependencies, first, second, third, and BCNF normal forms based on primary keys, lossless join decomposition. **Database Protection:** Security Issues, Threats to Databases, Security Mechanisms, Role of DBA, Discretionary Access Control, Backing Up and Restoring databases. **Views**: Creating, altering dropping, renaming and manipulating views **DCL Statements:** Creating/dropping users, privileges introduction, granting/revoking privileges, viewing privileges), Transaction control commands – Commit, Rollback	**15**

Textbooks:

- Fundamentals of Database System, ElmasriRamez, NavatheShamkant, Pearson Education, Seventh edition, 2017
- Database Management Systems, Raghu Ramakrishnan and Johannes Gehrke, 3rd Edition,2014
- Murach's MySQL, Joel Murach, 3rd Edition, 3rd Edition, 2019

Additional References:

- Database System Concepts, Abraham Silberschatz, HenryF.Korth, S.Sudarshan, McGraw Hill,2017
- MySQL: The Complete Reference, VikramVaswani , McGraw Hill, 2017
- Learn SQL with MySQL: Retrieve and Manipulate Data Using SQL Commands with Ease, Ashwin Pajankar, BPB Publications, 2020

Course Code	MAJOR SEM-I Practical	Credits	Lectures/ Week
26CSMJP111	CS Practical(A+B)	2	2

Part A : Digital System and Architecture

Course Objectives(CO):

CO1. To verify the truth tables of various logic gates
CO2. Develop proficiency in designing and implementing digital circuits.
CO3. Explore various components of digital systems, including processors, memory units, and input/output interfaces.

Course Outcomes (OC):

After successful completion of this course, students would be able to -
OC 1. Recall the truth tables, symbols, and basic functions of logic gates, flip-flops, and fundamental digital circuits
OC 2. Explain the working principles of combinational and sequential circuits, including adders, subtractors, comparators, flip-flops, counters, and shift registers.
OC 3. Apply Boolean algebra and logic design techniques to simplify expressions and implement digital circuits using Logisim. Design and implement digital circuits using multiplexers, demultiplexers, and flip-flop-based systems in Logisim.
OC 4. Analyze the behavior and performance of combinational and sequential circuits, including counters and shift registers, through simulation.

1	Study of discrete Logic gates: Study and verify the truth table of various logic gates (NOT, AND, OR, EX-OR, and EX-NOR).
2	Study of NAND and NOR gate as universal gate.
3	Simplify given Boolean expression
4	Design and verify a half/full adder
5	Design and verify half/full subtractor
6	Design and verify the operation of flip-flops using logic gates.
7	Verify the operation of a counter
8	Verification of De Morgan’s theorems
9	Using SPIM, write and test an adding machine program that repeatedly reads in integers and adds them into a running sum. The program should stop when it gets an input that is 0, printing out the sum at that point.

10	Using SPIM, write and test a program that reads in a positive integer using the SPIM system calls. If the integer is not positive, the program should terminate with the message “Invalid Entry;” otherwise, the program should print out the names of the digits of the integers, delimited by exactly one space. For example, if the user entered “528,” the output would be “Five Two Eight.”

PART B	**Introduction to Database Systems - Practical**	**Credit :1** **30 hrs**
Course Objectives (CO): CO1: **Systematic Design:** To master the transition from complex real-world requirements (ER Modeling) to structured data storage (Relational Modeling). CO2: **Constraint Enforcement:** To provide deep insight into maintaining data accuracy through various integrity constraints, transactions, and user access control. CO3: **Refinement & Optimization:** To teach students how to eliminate redundancy through normalization and improve retrieval performance using indexing techniques. CO4: **Operational Management:** To equip students with the skills to manage the full lifecycle of a database, including file loading, user privileges, and schema evolution.		
Course Outcomes (OC): After successful completion of this course, students would be able to OC1: Construct complex ER models including weak entities, class hierarchies, and aggregations, and perform algebraic operations to manipulate relational data. OC2: Convert advanced ER diagrams into relational schemas while mapping various constraints (Key, Participation, and Weak Entity) into table structures. OC3: Assess and refine database schemas through normalization (Refinement) and implement clustered indexing to optimize data storage and retrieval. OC4: Design a secure and functional database environment by managing file tables, enforcing transaction-based integrity, and configuring user-level security permissions.		
1	Perform Following: Create ER model with Key Constraints,Participation Constraints,Weak Entities, Class Hierarchies,Aggregation	
2	Perform Following: Create Relational Model Create and Modify Relations Integrity Constraints over Relations(Key Constraints, Foreign Key Constraints, General Constraints)	
3	Perform Following:	

	1. Create Relational Model 2. Enforcing Integrity Constraints(Transactions and Constraints) 3. Querying Relational Data
4	Perform Following: 1. Convert ER to Relational (Entity Sets to Tables ,Relationship Sets (without Constraints) to Tables 2. Translating Relationship Sets with Key Constraints 3. Translating Relationship Sets with Participation Constraints
5	Perform Following: 1. Convert ER to Relational (Entity Sets to Tables ,Relationship Sets (without Constraints) to Tables 2. Translating Weak Entity Sets 3. Translating Class Hierarchies 4. Translating ER Diagrams with Aggregation
6	Perform Relational algebra operations on above ER Model.
7	Perform Normalization and schema refinement on above ER and Relational Model.
8	Perform Following: Create Clustered index.
9	Perform Following: File Tables: Create, Alter and drop. Load Files.
10	Perform Following: Create database User, Grant and Deny.

Course Code	OPEN ELECTIVE SEM – II	Credits	Lectures/ Week
24CSOE231	Paper II -Introduction to E-Commerce using Artificial Intelligence	2	2

Course Objectives(CO):

CO1: Know how the business is carried out through electronic media
CO2: Understand the types and ways of doing business over internet
CO3: Apply the knowledge after becoming a professional or an entrepreneur
CO4: Analyze the security concerns while transacting using electronic media

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Define the concepts of e-business, e-commerce, types of e-commerce transactions, and e-business models.
OC2: Explain the role of e-payment systems, internet marketing, e-tailing, and AI tools used in digital commerce.
OC3: Apply AI applications such as chatbots, recommendation systems, and customer analytics in e-commerce business situations.
OC4:Examine security, ethical issues, and future opportunities of artificial intelligence in e-commerce.

Unit	Topics	No of Lectures
I	**E-Business and E-Commerce** • Definition: E-commerce and E-Business • Types of E Commerce transactions • E-Business Models • E-Business applications • Internet Marketing and E-Tailing. • E- Payment Systems	15
II	**AI in E-Commerce** • Introduction to Artificial Intelligence • AI tools used in Digital Commerce • AI Applications in E-Commerce • AI Applications in Indian E-Commerce • Security and Ethical Issues in AI-Based Commerce • Future Scope of AI in E-Commerce	15

References:

- Digital Business and E-Commerce, Dave Chaffey, Pearson
- E-Commerce: An Indian Perspective, P. T. Joseph, 7th Edition, PHI Learning, 2023.
- E-Commerce, Dr. Shivani Arora, Taxmann Publications, India.
- E-Commerce in India: Economic and Legal Perspectives, Pralok Gupta (Ed.)
- Artificial Intelligence for Business: A Roadmap for Getting Started with AI, Doug Rose, Wiley
- Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, Pearson, latest edition.,

Course Code	VOCATIONAL SKILL COURSE SEM – I	Credits	Lectures/ Week
26CSVC141	**Introduction to programming with Python (Practical Based)**	**2**	**2**

Course Objectives (CO):

CO1: Recall fundamental concepts, syntax, data types, operators, and standard libraries used in Python programming.
CO2: Understand Python programming constructs including control structures, functions, data structures, file handling, and regular expressions.
CO3: Apply Python programming techniques to develop programs using arrays, strings, lists, tuples, dictionaries, functions, and file operations.
CO4: Analyze program logic, data handling methods, and execution flow to solve computational problems and optimize Python-based solutions.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Identify Python syntax, programming elements, data types, operators, and standard functions.
OC2: Explain control structures, functions, data structures, file handling mechanisms, and regular expressions in Python.

OC3: Apply Python programming concepts to implement problem-solving solutions using data structures, functions, file handling, and modules.
OC4: Analyze Python programs and data processing techniques to interpret results and improve program efficiency.

Unit	Topics	No of Lectures
I	**Overview and Basic Elements of Python Programming:** Features of Python, Execution of a Python Program, Flavours of Python, Innards of Python, Python Interpreter, Comments, Docstrings, IDLE, Data types, Dictionary, Sets, Mapping, Basic Elements of Python, Variables, Input Function, Output Statements, Command Line Arguments. Operators, Precedence of Operators, Associativity of Operators **Control Statements:** The if statement, The if … else Statement, The if … elif … else Statement, Loop Statement- while loop, for loop, Infinite loop, Nested loop, The else suite, break statement, continue statement, pass statement, assert statement, return statement. **Arrays:** Creating Arrays, Indexing and Slicing of Arrays, Basic Array Operations, Arrays Processing, Mathematical Operations on Array, Aliasing Arrays, Slicing and Indexing in NumPy Arrays, Basic slicing, Advanced Indexing, Dimensions and Attributes of an Array **Functions:** Function definition and call, Returning Results, Returning Multiple Values from a Function, Built-in Functions, Difference between a Function and a Method, Pass Value by Object Reference, Parameters and Arguments, Recursive Functions, Anonymous or Lambda Functions. Modules in Python. **Strings:** Creating Strings, Functions of Strings, Working with Strings, Formatting Strings, Finding the Number of Characters and Words, Inserting Substrings into a String	15
II	**Exploring List, Tuples and Dictionaries:** Lists, List Functions and Methods, List Operations, List Slices, Nested Lists, Tuples, Functions in Tuple. **Working with Dictionaries:** Creating a Dictionary, Operators in Dictionary, Dictionary Methods, Using for Loop with	15

	Dictionaries, Operations on Dictionaries **Files in Python**: Opening and Closing a File, Working with Text Files, , Working with Binary Files, The 'with' statement, Pickle in Python, The seek() and tell() Methods, Random Accessing of Binary Files, Zipping and Unzipping Files, Working with Directories **Regular Expressions:** Introduction, Sequence Characters in Regular Expressions, Special Characters in Regular Expressions, Using Regular Expression on Files, Retrieving Information from an HTML File **Date And Time in Python:** Time, Date, Date and Time Now, combining date and times, formatting date and time, Finding and comparing dates, Sorting dates, Knowing the Time taken by a Program, Working with Calendar Module	

Textbooks:

- Practical Programming: An Introduction to Computer Science Using Python 3, Paul Gries , Jennifer Campbell, Jason Montojo, Pragmatic Bookshelf, 2nd Edition, 2014
- Programming through Python, M. T Savaliya, R. K. Maurya & G M Magar,Sybgen Learning India, 2020

Additional References:

- Python: The Complete Reference, Martin C. Brown, McGraw Hill, 2018
- Beginning Python: From Novice to Professional, Magnus Lie Hetland, Apress, 2017
- Programming in Python 3, Mark Summerfield, Pearson Education, 2nd Ed, 2018
- Python Programming: Using Problem Solving Approach, ReemaThareja, Oxford University Press, 2017
- Let Us Python, Yashwant. B. Kanetkar, BPB Publication, 2019

Course Code	**SEM I –Computer Science (Practical)**	**Credits**	**Practical /Week**
26CSVC141	**Introduction to Programming with Python (Practical Based)**	**2**	**2**

Course Objectives:

CO1: Recall Python syntax, basic commands, data types, and commonly used functions required for writing programs.

CO2: Understand the working of control structures, functions, data structures, file handling, and regular expressions through hands-on practice
CO3: Apply Python programming concepts to develop programs for problem-solving using arrays, strings, lists, tuples, dictionaries, functions, and file operations.
CO4: Analyze program logic, outputs, and errors to debug programs and improve efficiency of Python-based solutions.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Identify Python programming elements, syntax, and standard library functions during program development.
OC2: Explain the execution flow of Python programs involving control structures, functions, and data structures through practical implementation.
OC3: Apply Python programming skills to write, execute, and test programs for data handling, file processing, and problem-solving tasks.
OC4: Analyze program behavior, debug errors, and interpret outputs to ensure correctness and efficiency of solutions.

1	**Basic Programs:** 1.Write a program to display the message HELLO WORLD. 2.Write a program to declare some variables of type int, float and double. Assign some values to these variables and display these values. 3.Write a program to find the addition, subtraction, multiplication, and division of two numbers. 4.Write a program to swap two numbers without using a third variable. 5.Write a program to find the area of rectangle, square and circle. 6.Write a program to find the volume of a cube, sphere, and cylinder.
2	**Conditional statements:** 1.Write a program to enter a number from the user and display the month name. If number >13 then display invalid input using switch case. 2.Write a program to check whether the number is even or odd. 3.Write a program to check whether the number is positive, negative or zero. 4.Write a program to find the largest of three numbers.
3	**loops:** 1. Write a program to find the sum of squares of digits of a number. 2. Write a program to reverse the digits of an integer. 3. Write a program to find the sum of numbers from 1 to 100. 4. Write a program to print the Fibonacci series. 5. Write a program to find the reverse of a number. 6. Write a program to find whether a given number is palindrome or not. 7. Write a program to check whether the entered number is Armstrong or not. 8. Write a program to count the digit in a number. 9. Write a program to find the factorial of a number. 10.Write a program to check whether the entered number is prime or not.

4	**Functions & Recursive functions:** 1.Write a program to find the factorial of a number using recursive function. 2.Write a program to find the sum of natural number using recursive function.
5	**Arrays:** 1.Write a program to find the largest value that is stored in the array. 2.Write a program using pointers to compute the sum of all elements stored in an array. 3.Write a program to arrange the 'n'numbers stored in the array in ascending and descending order. 4.Write a program that performs addition and subtraction of matrices. 5.Write a program that performs multiplication of matrices
6	**Strings:** 1. Write a Python program to read a string from the user and display it. 2. Write a program to find the length of a given string. 3. Write a program to convert a string into uppercase and lowercase. 4. Write a program to concatenate two strings entered by the user. 5. Write a program to access and print each character of a string using a loop. **List:** 1. Write a Python program to create a list and display its elements. 2. Write a program to find the length of a list. 3. Write a program to add elements to a list (append, insert). 4. Write a program to remove elements from a list (remove, pop). 5. Write a program to access elements using indexing and slicing. 6. Write a program to find the largest and smallest element in a list. 7. Write a program to sort a list in ascending and descending order. 8. Write a program to reverse a list. 9. Write a program to count the frequency of an element in a list. 10. Write a program to remove duplicate elements from a list. 11. Write a program to merge two lists. 12. Write a program to find common elements between two lists. 13. Write a program to create a list of even numbers from a given list. 14. Write a program to find the second largest number in a list. 15. Write a program to flatten a nested list.
7	**Tuple:** 1. Write a Python program to create a tuple and display its elements. 2. Write a program to access elements of a tuple using indexing. 3. Write a program to find the length of a tuple. 4. Write a program to convert a list into a tuple. 5. Write a program to count occurrences of an element in a tuple. 6. Write a program to find the maximum and minimum elements in a tuple. 7. Write a program to check whether an element exists in a tuple. 8. Write a program to concatenate two tuples. 9. Write a program to slice a tuple.

	10. Write a program to unpack a tuple into variables. 11. Write a program to remove an element from a tuple (hint: convert to list). 12. Write a program to find the index of an element in a tuple. 13. Write a program to sort a tuple. 14. Write a program to convert a tuple of strings to a single string. 15. Write a program to create a tuple with different data types. **Dictionary:** 1. Write a Python program to create a dictionary and display its elements. 2. Write a program to access values using keys. 3. Write a program to add and update elements in a dictionary. 4. Write a program to remove elements from a dictionary. 5. Write a program to check whether a key exists in a dictionary. 6. Write a program to count the frequency of elements using a dictionary. 7. Write a program to merge two dictionaries. 8. Write a program to iterate through keys and values. 9. Write a program to find the maximum and minimum value in a dictionary. 10. Write a program to sort a dictionary by keys. 11. Write a program to sort a dictionary by values. 12. Write a program to create a dictionary from two lists (keys and values). 13. Write a program to invert a dictionary (swap keys and values). 14. Write a program to count occurrences of each word in a sentence using a dictionary. 15. Write a program to remove duplicate values from a dictionary.
8	**Files:** 1. Write a Python program to create a file and write data into it. 2. Write a program to read the contents of a file and display it. 3. Write a program to append data to an existing file. 4. Write a program to count the number of characters in a file. 5. Write a program to count the number of lines in a file.
9	**Regular Expression:** 1. Write a Python program to check whether a given string contains only alphabets. 2. Write a program to check whether a string starts with a specific word. 3. Write a program to find all digits in a given string. 4. Write a program to count the number of vowels in a string using regular expressions. 5. Write a program to check whether a string contains only numbers.
10	**Date and Time:** 1. Write a Python program to display the current date and time. 2. Write a program to display only the current date. 3. Write a program to display only the current time. 4. Write a program to print the current year, month, and day separately. 5. Write a program to create a specific date (e.g., 15 August 2025).

	6. Write a program to find the day of the week for a given date. 7. Write a program to add a certain number of days to the current date. 8. Write a program to subtract two dates and find the difference in days.

Course Code	SKILL ENHANCEMENT COURSE SEM – I - Course Title	Credits	Lectures/ Week
26CSSE151	**Statistics with R**	**2**	**2**

Course Objectives (CO):

CO1: Recall fundamental concepts, terminology, R environment components, data types, and basic statistical functions used in data analysis.
CO2: Understand R programming constructs, data structures, string operations, statistical methods, and graphical techniques for data analysis.
CO3: Apply R programming, data manipulation, statistical analysis, regression techniques, and visualization methods to solve data-driven problems
CO4: Analyze datasets and interpret statistical and graphical outputs to derive meaningful insights and support decision making.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Identify fundamental concepts, terminology, R environment components, data types, and basic statistical functions used in data analysis.
OC2: Explain R programming constructs, data structures, string operations, statistical methods, and graphical techniques for data analysis.

CO3: Apply R programming, data manipulation, statistical analysis, regression techniques, and visualization methods to analyze and represent data.
OC4: Analyze datasets and interpret results using statistical functions, contingency tables, regression models, and graphical representations to derive meaningful insights.

Unit	Topics	No of Lectures
I	**Exploring R Language and Setting Up environment:** Introduction to R, Terminologies in R, R Environment, Installing R, Studio, and R Commander, Customizing Studio, Data Management in Studio, R Graphical User Interface (R GUI), Working with R Scripts **Implementing ting Expression:** Expressions, assignment, Decision making, Loops, data and time options in R **Essential Data Structures in R:** Vectors, Matrix, Arrays, Lists, Data frames, Functions **Implementing Strings in R**: Character strings in R, Character Strings, , Strings and R objects, String Manipulation: Printing Characters, Basic String Manipulations, String Operations	15
II	**Built-in statistical functions in R:** mean() function, Median, Standard Deviation, Some other built-in statistical functions, **Regression Analysis:** Regression Analysis-Linear Regression and Multiple Regression, Normal Distribution-dnorm(),,pnorm(),qnorm(),rnorm() **Binomial Distribution:** dbinom(),pbinom(),qbinom(),rbinom() Functions, Time Series Analysis **Visualizing and analysing Data in R:** Tabulation, Contingency Tables, Making R Contingency Tables, Making R Custom Contingency Tables, Selection of Parts in a Table Object, Conversion of an Object into the Table, Testing Table Objects, Making R Complex Tables, Representing data through Cross Tabulation **Graphical Models & analysis**: Plots made of Single Plots made of Two Variables , Variable, Plots made of Multiple Variables, Special Plots, Storing Graphics	15

Textbooks:

- Statistical Programming in R, K.G. Srinivasa G.M. Siddesh,Chetan Shetty , Oxford

University Press, 2017
- Learning R: A Language for Data Analytics and Visualization, Sybgen Learning, R. K. Maurya, 2021
- Introduction to Statistics and Data Analysis With Exercises, Solutions and Applications in R: Heumann, Christian, Schomaker, Michael, Shalabh, Publisher" Springer 2016

Additional References:

- Learning R Programming, Kun Ren, Packt Publishing, 2018
- R Programming for Statistics and Data Science(Video), 365 Careers, Packt, 2018
- R Programming Fundamentals, Kaelen Medeiros, Oreily-Packt Publishing

Course Code	SEM I –Computer Science (Practical)	Credits	Practical /Week
26CSSE151	Statistics with R – Practical	2	4

Course Objectives:

CO1: Recall fundamental concepts, terminology, R environment components, data types, and basic statistical functions used in data analysis.
CO2: Understand R programming constructs, data structures, string operations, statistical methods, and graphical techniques for data analysis.
CO3: Apply R programming, data manipulation, statistical analysis, regression techniques, and visualization methods to solve data-driven problems.
CO4: Analyze datasets and interpret statistical and graphical outputs to derive meaningful insights and support decision making.

Course Outcomes:

After successful completion of this course, students would be able to
CO1: Identify fundamental concepts, terminology, R environment components, data types, and basic statistical functions used in data analysis.
CO2: Explain R programming constructs, data structures, string operations, statistical methods, and graphical techniques for data analysis.

CO3: Apply R programming, data manipulation, statistical analysis, regression techniques, and visualization methods to analyze and represent data. CO4: Analyze datasets and interpret results using statistical functions, contingency tables, regression models, and graphical representations to derive meaningful insights.	
1	**Setting Up R Environment:** 1. Write an R program in RStudio Script Editor to display your current R version, active session details, and current working directory. 2. Write an R program in RStudio to install and load the "ggplot2" package and verify if it has been successfully attached to the session. 3. Create a new R Project named Stats_Practicals in RStudio, set its working directory, and print the location. 4. Write an R program in RStudio to read a CSV file named "student.csv" and display the total number of records present in it.
2	**Customizing RStudio and Data Management:** 1. Write an R program in RStudio to create a dataset of 5 students with their roll numbers, names, and marks (as shown below). Convert it into a data frame and display its structure and summary. 2. Write an R program to import a CSV file named "products.csv" (containing the data shown below), display only the ProductName and Price columns. 3. Write an R program to create a data frame of employee details (with missing values as shown below). Remove rows containing missing salary values and display the cleaned data. 4. Write an R program to create a data frame of monthly sales and export it to a file named "monthly_sales.csv" in the working directory.
3	**Implementing Expressions and Control Structures:** 1. Write an R program in RStudio which display the decision of grade based on the following rules: ≥ 75 = Distinction, 60–74 = First Class, 40–59 = Pass, < 40 = Fail. Hence give decision for the marks 68, 80, 50. 2. Write an R program to find the largest number among three given numbers (45, 92, 78) using if-else statements. 3. Write an R program to calculate the factorial of a given number (n = 6) using a for loop. 4. Write an R program to print the Fibonacci series of the first 10 terms using a while loop.
4	**Essential Data Structures in R:** 1. Create a numeric vector of marks for five students: 12, 15, 20, 22, 18. Compute the sum, mean, median, sort the vector in descending order and display the result. 2. Create a data frame of 4 students with columns: Roll, Name, Age, Marks using the table below. Then show str(), summary() and select rows where Marks >= 70.

<table>
<tr><td></td><td>3. Create a list named mylist that contains:
● A numeric vector c(10,20,30),
● A 2×2 matrix with values 1:4, and
● A data frame with IDs and Names ID: 1,2 Name: "X","Y".
Then demonstrate how to access (a) the second element of the vector, (b) the element at row 2, column 1 of the matrix, and (c) the second row's Name from the data frame inside the list.</td></tr>
<tr><td>5</td><td>Implementing Strings in R:
1. Given the string "Welcome to RStudio Practical", write an R program to:
a) convert it to upper case and lower case.
b) extract the substring from character positions 9 to 15,
c) find the length (number of characters) of the string.
2. Create two character vectors:
first <- c("Dr.","Mr.","Ms.") and last <- c("Sharma","Patel","Singh").
Write an R program to concatenate them into a single vector of full names with a space between title and 24 | P a g e surname, and then collapse them into one comma-separated string.
3. Given the string "R is powerful. R is flexible. R is popular." write an R program to count how many times the letter R (case-insensitive) appears, and replace every occurrence of "R" (or "r") with "R-lang".
4. Write an R program in RStudio to take a sentence:
"RStudio makes data analysis easy and powerful" and perform the following operations:
a) Split the sentence into individual words.
b) Count how many words there are
c) Join the words back into a single string separated by hyphens (-).</td></tr>
<tr><td>6</td><td>Built-in Statistical Functions in R:
1. Write an R program to create a numeric vector c(20, 15, 12, 22, 13, 15, 14, 17, 15, 20, 21). Find and display its mean, median, variance, and standard deviation.
2. Write an R program to find the minimum, maximum, range, and summary of the following: 15, 14, 17, 15, 20, 21, 20, 15, 12, 22, 13.
3. Write an R program to correlation and covariance of the following:
<table><tr><td>X</td><td>2</td><td>3</td><td>5</td><td>6</td><td>9</td></tr><tr><td>Y</td><td>5</td><td>3</td><td>4</td><td>6</td><td>12</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>4. Write an R program to generate 10 random numbers from a normal distribution with mean = 50 and sd = 10. Display their mean, standard deviation, and plot a histogram.</td></tr>
<tr><td>7</td><td>Regression Analysis:
1. A small advertising study recorded monthly advertising spend (Ad) and monthly sales (Sales) as below. Fit a simple linear regression model Sales ~ Ad. Report the estimated intercept, slope, R-squared, and write the regression equation.

<table>
<tr><td>Ad</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td></tr>
<tr><td>Sales</td><td>14</td><td>25</td><td>35</td><td>45</td><td>56</td></tr>
</table>

2. create a scatter plot with the regression line for the following data:

<table>
<tr><td>Ad</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td></tr>
<tr><td>Sales</td><td>14</td><td>25</td><td>35</td><td>45</td><td>56</td></tr>
</table>

3. Given the dataset below with two predictors x and y and response z, fit the model z ~ x + y. Report coefficients and Adjusted R-squared.

<table>
<tr><td>X</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>Y</td><td>2</td><td>1</td><td>4</td><td>3</td><td>5</td></tr>
<tr><td>Z</td><td>6.0</td><td>7.5</td><td>10.0</td><td>11.5</td><td>14.5</td></tr>
</table>
4. For given dataset predict z for x = 6 and y = 4.

<table>
<tr><td>X</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>Y</td><td>2</td><td>1</td><td>4</td><td>3</td><td>5</td></tr>
<tr><td>Z</td><td>6.0</td><td>7.5</td><td>10.0</td><td>11.5</td><td>14.5</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Working with Distributions:
1. Normal distribution (density & probability):</td></tr>
</table>

<table>
<tr><td></td><td>A machine produces items whose weights are approximately Normal with mean = 50 grams and sd = 10 grams.
a) Compute the probability density at weight = 60 grams.
b) Compute the probability that a randomly chosen item weighs less than or equal to 60 grams
2. Binomial distribution (point & cumulative probability):
A factory inspector inspects 10 items and each item independently has a 0.40 probability of being defective.
a) Compute the probability that exactly 3 items are defective.
b) Compute the probability that at most 3 items are defective (P(X ≤ 3)).
3. Poisson distribution (probability & cumulative):
A call center receives on average 2 calls per minute (assume Poisson).
a) Compute the probability that exactly 0 calls arrive in 1 minute.
b) Compute the probability that at most 2 calls arrive in 1 minute.
4. Simulation and comparison — Normal vs Uniform:
Generate two samples of size 500 each: one from N(0,1) and one from Uniform(-2,2). For reproducibility use 41 | P a g e set.seed(123). Compute and report sample mean and sd for both samples and display two histograms side-by-side for visual comparison.</td></tr>
<tr><td>9</td><td>Time Series Analysis and Data Visualization:
Create a monthly time series of length 24 (Jan 2023 → Dec 2024) using the values below, convert to a ts object with frequency=12, print start(), end(), frequency() and plot using plot.ts().
Data (24 values):
100, 105, 110, 108, 115, 120, 125, 130, 128, 135, 140, 145, 150, 155, 160, 158, 165, 170, 175, 180, 185, 190, 195, 200.</td></tr>
<tr><td>10</td><td>Graphical Models and Data Visualization:
1. Use the numeric vector x <- c(12,15,18,22,25,28,30,35,40,45) to draw a histogram and overlay a kernel density curve on top.
2. Use the data below to create side-by-side boxplots of Score by Class (Class1, Class2).
<table><tr><td>Class</td><td>Class1</td><td>Class1</td><td>Class1</td><td>Class2</td><td>Class2</td><td>Class2</td></tr><tr><td>Score</td><td>65</td><td>70</td><td>72</td><td>60</td><td>75</td><td>80</td></tr></table></td></tr>
</table>

Course Code	SKILL ENHANCEMENT COURSE SEM – I - Course Title	Credits	Lectures/Week
26CSSE151	**Linux Operating System**	**2**	**2**

Course Objectives (CO):

CO1: Recall fundamental concepts, terminology, Linux commands, utilities, and system components related to the Linux operating system.
CO2: Explain Linux architecture, file systems, command structures, permissions, security mechanisms, and the working of system tools and utilities.
CO3: Apply Linux commands, shell scripting, system administration techniques, and programming environment setup to perform file operations, process management, and basic automation tasks.
CO4: Analyze system behavior, file permissions, security features, process management, and networking operations to effectively manage and troubleshoot a Linux environment.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Identify Linux concepts, terminology, commands, utilities, and system components.
OC2: Explain Linux architecture, file systems, command structure, permissions, security features, and system utilities.
OC3: Apply Linux commands, file handling operations, filters, editors, and basic system administration techniques to perform tasks in a Linux environment.
OC4: Apply shell scripting concepts, including variables, control structures, and I/O redirection, to automate simple tasks.

Unit	Topics	No of Lectures
I	**Introduction to Linux Operating System and Basics:** History of Linux, GNU Info and Utilities, Various Linux Distributions, The Unix/Linux architecture, Features of Unix/Linux **Installation of Ubuntu Linux Operating System:** Booting and Installing from USB/DVD, Using Ubuntu Software Center / Using Synaptic, Exploring useful software packages **Becoming an Ubuntu Power User:** Administering system and user settings, Learning Unity keyboard shortcuts, Using the Terminal **Linux Basics:** Starting the shell, Shell prompt, Command	**15**

	structure, File Systems and Directory Structure, man pages, more documentation pages **File System Commands:** touch, help, man, more, less, pwd, cd, mkdir, rmdir, ls, find, etc. **File Handling Commands:** cat, cp, rm, mv, more, file, wc, od, cmp, diff, comm, gzip, gunzip, zip, unzip, tar, ln, umask, etc. **General Purpose Utility Commands:** cal, date, echo, man, printf, passwd, script, who, uname, tty, stty, etc. **Linux File Permissions:** Understanding Linux file permissions, Using Linux groups. Decoding file permissions, Changing security settings, chmod, chown, chgrp	
II	**Linux Security:** Understanding Linux Security, Uses of root, sudo command, Working with passwords, Understanding ssh **Networking Commands:** who, whoami, ping, telnet, ftp, ssh, etc. **Editors:** vi, sed, awk **Simple Filters and I/O Redirection**: head, tail, cut, paste, sort, grep family, tee, uniq, tr, etc. **Shell Scripting:** Defining variables, reading user input, exit and exit status commands, expr, test, [], if conditional, logical operators, Conditions (for loop, until loop, and while loop), arithmetic operations, Redirecting input/output in scripts, creating your own redirection. **Working and Managing Processes:** sh, ps, kill, nice, at, batch, etc. **Job scheduling commands:** ps, nice, renice, at, batch, cron table Installation of C/C++/Java/Python Compiler and Environment Setup and Basic programming using C and Python languages.	**15**

Textbooks:

- Linux Command line and Shell Scripting Bible, Richard Blum, Wiley India.
- Unix: Concepts and Applications, Sumitabha Das, 4th Edition, McGraw Hill.
- Official Ubuntu Book, Matthew Helmke& Elizabeth K. Joseph with Jose Antonio Rey and Philips Ballew, 8th Ed

Additional References:

- Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill, 2008.
- Linux: Complete Reference, Richard Petersen, 6th Edition, Tata McGraw-Hill
- Beginning Linux Programming, Neil Mathew, 4th Edition, Wiley Publishing, 2008.

Course Code	SEM I –Computer Science (Practical)	Credits	Practical/Week
26CSSE151	**Linux Operating System – Practical**	**2**	**4**

Course Objectives:

CO1: To familiarize students with basic Linux commands, file system structure, and essential system utilities through hands-on practice.
CO2: To develop understanding of Linux command syntax, file permissions, and basic security mechanisms using practical exercises.
CO3: To enable students to perform file management, process control, and job scheduling using Linux commands and tools.
CO4: To introduce shell scripting and basic system administration tasks for simple automation and problem-solving in a Linux environment.

Course Outcomes:

After successful completion of this course, students would be able to
OC1: Recall and identify fundamental Linux commands, utilities, file structures, and system components through hands-on practice.
OC2: Explain the working of Linux architecture, file systems, command syntax, file permissions, and basic security features using practical demonstrations.
OC3: Apply Linux commands, shell scripting, and system administration techniques to perform file operations, process management, job scheduling, and basic automation tasks.
OC4: Analyze and troubleshoot issues related to file permissions, process control, system performance, and basic networking using appropriate Linux tools.

1	**Basic Linux Commands and Navigation** Demonstrate the use of basic Linux commands such as pwd, ls, cd, mkdir, and rmdir. Create a directory structure, navigate between directories, and display contents using appropriate commands.
2	**File Handling Operations** Perform file operations using commands like touch, cat, cp, mv, and rm. Create files, copy them, rename them, and delete them while observing the effects of each command.

3	**File Permissions and Security** Create files and directories and analyze their default permissions. Modify permissions using chmod, change ownership using chown, and group using chgrp. Explain the role of permissions in system security.
4	**Working with Filters and I/O Redirection** Use filters such as head, tail, sort, uniq, cut, and grep to process text files. Demonstrate input/output redirection using >, >>, and | operators.
5	**Editors and Text Processing Tools** Edit a text file using vi editor. Perform basic operations like insert, delete, search, and save. Use sed or awk to perform pattern-based text processing.
6	**Shell Scripting Basics** Write a shell script to: Accept user input Perform arithmetic operations Use conditional statements (if) and loops (for or while) Execute and analyze the script output.
7	**Process Management** Use commands such as ps, top, kill, and nice to monitor and control processes. Start a process, change its priority, and terminate it.
8	**Job Scheduling** Schedule tasks using at, batch, and cron. Create a cron job to execute a script at a specified time and verify its execution.
9	**Networking Commands and Remote Access** Use networking commands such as ping, who, whoami, and uname. Demonstrate secure remote login using ssh and explain its importance.
10	**Programming Environment Setup and Execution** Install and configure compilers/interpreters for C and Python in Linux. Write and execute a simple C program and a Python program. Compare the execution process in both languages.

SEMESTER-II

Course Code	MAJOR SEM – II	Credits	Lectures/ Week
26CSMJ211	Paper I- Design and Analysis of Algorithm	2	2

Course Objectives (CO):

CO1: Remember fundamental concepts of algorithms, data structures, and computational problem-solving.
CO2: Understand algorithm analysis techniques, including time and space complexity and asymptotic notations, Understand the structure, operations, and applications of basic data structures such as arrays, stacks, and lists.
CO3: Apply recursion and iterative approaches to solve computational problems and compare their efficiency, basic sorting and searching techniques to organize and retrieve data efficiently, algorithm design techniques.
CO4: Analyze the performance of algorithms using different complexity measures and comparative methods, algorithm design techniques such as greedy, divide-and-conquer, dynamic programming, and backtracking,problem-solving approaches using appropriate algorithms for real-world computational tasks.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Identify fundamental concepts, terminology, data structures, algorithmic techniques, and their applications in solving computational problems.
OC2: Explain fundamental concepts of algorithms and data structures, including their design techniques, performance characteristics, and applications in solving computational problems.
OC3: Apply appropriate data structures and algorithmic techniques to solve computational problems, including the use of linear data structures, recursion and iteration, and fundamental sorting and searching methods.
OC4: Analyze algorithmic approaches and performance characteristics to select appropriate techniques for solving computational problems.

Unit	Topics	No of Lectures
I	**I Introduction to algorithms -** What is algorithm, analysis of algorithm, Types of complexity, Running time analysis, How to Compare Algorithms, Rate of Growth, Types of Analysis, Asymptotic Notation, Big-O Notation, Omega-Ω	15

	Notation,Theta-Θ Notation, Asymptotic Analysis, Performance characteristics of algorithms,Estimating running time / number of steps of executions on paper, Idea of Computability. **Introduction to Data Structures -** What is data structure, types, Introduction to Array(1-d & 2-d), Stack and List data structures, operations on these data structures, advantages disadvantages and applications of these data structures like solving linear equations, Polynomial Representation, Infix-to-Postfix conversion. **Recursion -** What is recursion, Recursion vs Iteration, recursion applications like Factorial of a number, Fibonacci series & their comparative analysis with respect to iterative version, Tower of Hanoi problem. **Basic Sorting Techniques** - Bubble, Selection and Insertion Sort & their comparative analysis	
II	**Searching Techniques** - Linear Search and its types, Binary Search and their comparative analysis, Selection Techniques - Selection by Sorting, Partition-based Selection Algorithm, Finding the Kth Smallest Elements in Sorted Order & their comparative analysis, String Algorithms - Pattern matching in strings, Brute Force Method & their comparative analysis **Algorithm Design Techniques** - Introduction to various types of classifications/design criteria and design techniques, Greedy Technique - Concept, Advantages & Disadvantages, Applications, Implementation using problems like - file merging problem. Divide-n-Conquer - Concept, Advantages & Disadvantages, Applications, Implementation using problems like - merge sort, Strassen's Matrix Multiplication. **Dynamic Programming -** Concept, Advantages & Disadvantages, Applications, Implementation using problems like - Fibonacci series, Factorial of a number, Longest Common subsequence. **Backtracking Programming -** Concept, Advantages & Disadvantages, Applications, Implementation using problems like N-Queen Problem	**15**
Textbooks: ● Data Structure and Algorithm Using Python, Rance D. Necaise, Wiley India Edition, 2016.		

- Data Structures and Algorithms Made Easy, Narasimha Karumanchi, CareerMonk Publications, 2016.
- Introduction to Algorithms, Thomas H. Cormen, 3rd Edition, PHI.

Additional References:

- Introduction to the Design and Analysis of Algorithms, Anany Levitin, Pearson, 3rd Edition, 2011.
- Design and Analysis of Algorithms, S. Sridhar, Oxford University Press, 2014.Achyut S. Godbole, Atul Kahate, Operating Systems, Tata McGraw Hill, 2017
- Naresh Chauhan, Principles of Operating Systems, Oxford Press, 2014
- Andrew S Tanenbaum, Herbert Bos, Modern Operating Systems, 4e Fourth Edition, Pearson Education, 2016

Course Code	MAJOR SEM – II	Credits	Lectures/ Week
26CSMJ212	Paper II - Object Oriented Programming	2	2

Course Objectives (CO):

CO1: To introduce the fundamentals of programming and object-oriented concepts in C++
CO2: To develop understanding of control structures, arrays, strings and functions for solving computational problems
CO3: To enable students to apply object-oriented principles for designing classes and objects
CO4: To build the ability to analyze advanced OOP concepts for efficient program design

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Define fundamental concepts of OOPs including tokens, data types, I/O and operators in C++ with examples
OC2: Explain control structures, arrays, strings and functions to solve computational problems
OC3: Apply OOPs principles to formulate class and object design
OC4: Analyze polymorphism, inheritance, file handling and pointer-based memory management in program design

Unit	Topics	No of Lectures
I	**Introduction to Programming Concepts:** Object oriented programming paradigm, basic concepts of object oriented programming, benefits of object oriented programming, object oriented languages, applications of object oriented programming.Tokens-keywords, identifiers, constants-integer, real, character and string constants, backslash constants, features of C++ and its basic structure, simple C++ program without class, compiling and running C++ program. **Data Types, Data Input Output and Operators:** Basic data types,variables, rules for naming variables, programming constants, the type cast operator, implicit and explicit type casting, cout and cin statements,operators, precedence of operators. **Decision Making, Loops, Arrays and Strings:** Conditional	15

	statements-if,if...else, switch loops- while, do...while, for, types of arrays and string and string manipulations **Unified Modeling Language (UML):** Introduction to UML & class diagrams. **Classes, Abstraction & Encapsulation:** Classes and objects, Dot Operator, data members, member functions, passing data to functions, scope and visibility of variables in function. **Constructors and Destructors:** Default constructor, parameterized constructor, copy constructor, private constructor, destructors. **Working with objects:** Accessor - mutator methods, static data and static function, access specifiers, array of objects.	
II	**Polymorphism -** Binding-static binding & overloading, constructor overloading function overloading, operator overloading, overloading unary and binary operators. **Modelling Relationships in Class Diagrams:** Association, Aggregation-Composition and examples covering these principles. **Inheritance:** Defining base class and its derived class, access specifiers, types of inheritance-single, multiple, hierarchical, multilevel, hybrid inheritance, friend function and friend class, constructors in derived classes. **Modelling Relationships:** Generalization-Specialization and examples covering these principles **Run time Polymorphism -** Dynamic Binding, Function overriding, virtual function, pure virtual function, virtual base class, abstract class. **Pointers:** Introduction to pointers, * and & operators, assigning addresses to pointer variables, accessing values using pointers, pointers to objects & this pointer, pointers to derived classes **File Handling:** File Stream classes, opening and closing file-file opening modes, text file handling, binary file handling. **Applying OOP to solve real life applications:** To cover case studies like library management, order management etc. to design classes covering all relationships	**15**

Textbooks:

- Object Oriented Programming with C++, Balagurusamy E., 8th Edition, McGraw Hill Education India.
- UML & C++: A Practical Guide to Object Oriented Development, Lee/Tepfenhart, Pearson Education, 2nd Edition 2015

Additional References:

- Mastering C++ by Venugopal, Publisher: McGraw-Hill Education, 2017
- Let Us C++ by KanetkarYashwant, Publisher: BPB Publications, 2020
- Object Oriented Analysis and Design by Timothy Budd TMH, 2001

Course Code	MAJOR SEM-II Practical	Credits	Lectures/Week
26CSMJP21	CS Practical (A+B)	2	4

Part A - Design and Analysis of Algorithm

Course Objectives:

CO1: To develop hands-on skills in designing and analyzing basic algorithms and understanding their performance using simple programs.
CO2: To enable students to implement fundamental data structures such as arrays, stacks, and lists for solving computational problems.
CO3: To provide practical experience in applying techniques like recursion, sorting, and searching to real programming problems.
CO4: To introduce students to algorithm design techniques (Greedy, Divide & Conquer, Dynamic Programming, Backtracking) through implementation and experimentation.

Course Outcomes:

After successful completion of this course, students would be able to
CO1: Recall and identify basic concepts of algorithms, complexity, and data structures through program-based exercises.
CO2: Understand and explain the working of data structures, recursion, and algorithm behavior by observing program execution and outputs.
CO3: Apply algorithms and data structures to solve problems such as sorting, searching, recursion-based problems, and stack applications using programs.
CO4: Analyze and compare the performance of different algorithms (sorting, searching, design techniques) based on time complexity and practical execution.

1	Array Operations: Implement programs for 1-d arrays, Implement programs for 2-d arrays.

2	List-Based Stack Operations: Create a list-based stack and perform stack operations.
3	Linear and Binary Search: Implement linear and binary search algorithms on a list.
4	Sorting Algorithms: Implement sorting algorithms (e.g., bubble, selection, insertion).
5	Nth Max/Min Element: Implement algorithms to find Nth Max/Min element in a list.
6	String Pattern Matching: Implement algorithms to find a pattern in a given string.
7	Recursion: Implement recursive algorithms (e.g., factorial, Fibonacci, Tower of Hanoi).
8	Greedy Algorithm: Solve problems like file merging and coin change using the Greedy Algorithm.
9	Divide and Conquer: Implement algorithms like merge sort and Strassen's Matrix Multiplication.
10	Dynamic Programming: Implement algorithms for Fibonacci series and Longest Common Subsequence using dynamic programming.

Part B : **Object Oriented Programming – Practical**
Course Objectives (CO): CO1:To build a strong foundation in C++ programming and problem-solving concepts CO2:To develop understanding of program execution CO3:To promote understanding of structured program design using object-oriented approaches CO4:To develop analytical and problem-solving skills for improving program efficiency and reliability
Course Outcomes (OC): After successful completion of this course, students would be able to OC1:Identify and classify errors and basic C++ constructs OC2:Trace execution of C++ programs using constructors, access specifier and special functions OC3:Develop C++ programs using classes, objects, control structures and OOP concepts to solve given problems OC4:Debug and optimize C++ programs using polymorphism, pointers and file handling

1	Programs based on fundamental concepts of OOPS 1. Write a program to display the message HELLO WORLD. 2. Write a program to declare constant and variable. 3. Write a program to declare variables of different data types. 4. Write a program to declare multiple variables of the same data types. 5. Write a program to find the addition, subtraction, multiplication and division of two numbers. 6. Write a program to swap two numbers without using a third variable. 7. Write a program to input two numbers using cin and display addition using cout. 8. Write a program to input student details (name, roll no, marks) and display them properly.
2	Programs based on Conditional statements 1. Write a program to enter a number from the user and display the month name. If number >13 then display invalid input using switch case. 2. Write a program to check whether the number is even or odd. 3. Write a program to check whether the number is positive, negative or zero. 4. Write a program to find the factorial of a number. 5. Write a program to check whether the entered number is prime or not. 6. Write a program to find the largest of three numbers.
3	Programs based on branching and loops 1. Write a program to find the sum of numbers from 1 to 100. 2. Write a program to print the Fibonacci series. 3. Write a program to find the reverse of a number. 4. Write a program to find whether a given number is palindrome or not. 5. Write a program to count the digit in a number.
4	Program to demonstrate one and two dimensional arrays using classes.
5	Programs to demonstrate various types of constructors and destructors.
6	Programs to demonstrate use of public, protected & private scope specifiers.
7	Programs to demonstrate single and multilevel inheritance, function overloading and overriding.
8	Programs to implement functions, friend functions and inline functions.
9	Programs to demonstrate use of pointers. 1. Write a C++ program to display the value and address of a variable using pointers. 2. Write a program to perform addition and subtraction of two pointer variables. 3. Write a C++ program to swap two numbers using pointers (call by address).
10	Programs to demonstrate text and binary file handling.

Course Code	MINOR SEM – II	Credits	Lectures/ Week
26CSMR221	**Paper I -IT tools and Digital Productivity**	**2**	**2**

Course Objectives (CO):

CO1:Recall the basic concepts of IT tools, software types, and digital productivity.
CO2: Explain the features and functionalities of document creation, data management, and presentation tools.
CO3:Apply IT tools to create professional documents, spreadsheets, and presentations for academic and business purposes..
CO4:Analyze the use of digital collaboration tools and version control systems to improve productivity and teamwork..

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1:Describe various IT tools and their role in enhancing digital productivity.
OC2:Explain the working of document processing, spreadsheet analysis, and presentation tools.
OC3: Apply appropriate tools to create, format, and manage digital content effectively.
OC4: Analyze collaborative workflows using cloud-based tools and version control systems.

Unit	Topics	No of Lectures
I	**Introduction to IT Tools, Digital Environment, Document Processing tools** ● Overview of IT tools and their applications ● Types of software: System software and Application software ● Introduction to digital productivity ● Cloud computing basics and online tools (Google Workspace / Microsoft 365 overview) ● Creating and formatting documents ● Styles, templates, tables, and images ● Page layout, headers, footers, references ● Mail merge and document collaboration ● Exporting and sharing documents.	**15**
II	**Spreadsheet basics**: worksheets, cells, formatting	**15**

	• Formulas and functions (SUM, AVERAGE, IF, COUNT) • Charts and data visualization • Sorting, filtering, and data validation • Introduction to pivot tables • Creating presentations: themes, layouts, multimedia • Slide transitions and animations • Effective presentation design principles • Collaboration tools: Google Drive, Microsoft Teams, etc. • Introduction to version control (basic Git concepts)	

References:

- Kenneth H.Rosen. Discrete Mathematics and its applications. (7th edition) McGraw-Hill Higher Education, 2017.
- Bernard Kolman, Robert C.Busby , and Sharon Cutler Ross. Discrete Mathematical Structures (6th edition). Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2015.

Additional References:

- John Clark and Derek Holton, a first look at Graph Theory, 2013

ASSIGNMENTS:

Practical 1: Basic Document Formatting

- Create a document on “Role of IT in Education” with proper formatting.
- Apply bullets, numbering, alignment, and spacing.
- Apply heading styles in a document.
- Add headers, footers, page numbers, and margins

Practical 2: Tables and Images

- Insert and format a table (student details).
- Insert an image, apply wrapping, and add a caption.
- Insert Smart chart
- Insert a Table of Contents (TOC) using styles.
- Add citations/references and update the TOC.

Practical 3: Mail Merge

- Create a data source (names & addresses).
- Generate personalized letters/labels using mail merge

Practical 4: Spreadsheet Formatting and applying functions

- Create a sheet with student marks and apply formatting.

- Calculate total and average.
- Use SUM, AVERAGE, COUNT functions.
- Apply IF function to assign grades.

Practical 5: Charts and Pivot Tables

- Create bar and pie charts from data.
- Add titles and labels and interpret results
- Create a dataset and generate a pivot table.
- Summarize data using totals/averages.

Practical 6: Data Handling

- Perform sorting and filtering.
- Apply data validation (dropdown list).

Practical 7: Basic Presentation and Transitions

- Create a 6–8 slide presentation on “Digital India”.
- Use themes, layouts, text, and images
- Apply slide transitions.
- Add custom animations to objects

Practical 8: Multimedia

- Insert images, audio, or video in slides.
- Maintain proper alignment and consistency.

Practical 9: File Management using Google Drive

- Create folders and upload, rename, organize files.
- Perform move, copy, delete operations.
- Share files with Viewer, Commenter, Editor access.
- Compare permission levels.

Practical 10: Collaboration and Integration

- Collaborate on a Google Doc with comments and edits.
- Check and restore version history.
- Create files using Docs/Sheets/Slides from Drive.
- Upload and convert files (Word/PDF to Docs) and share.

Course Code	OPEN ELECTIVE SEM – I	Credits	Lectures/ Week
26CSOE132	Paper II– Cyber Law and Digital Security	2	2

Course Objectives (CO):

CO1: To understand concepts of e-business, e-commerce, transaction types, and e-business models.
CO2: To explain the role of e-payment systems, internet marketing, e-tailing, and AI tools in digital commerce.
CO3: To apply AI applications such as chatbots, recommendation systems, and customer analytics in e-commerce.
CO4: To analyze security, ethical issues, and future opportunities of AI in e-commerce.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Define the concepts of e-business, e-commerce, types of e-commerce transactions, and e-business models.
OC2: Explain the role of e-payment systems, internet marketing, e-tailing, and AI tools used in digital commerce.
OC3: Apply AI applications such as chatbots, recommendation systems, and customer analytics in e-commerce business situations.
OC4: Examine security, ethical issues, and future opportunities of artificial intelligence in e-commerce.

Unit	Topics	No of Lectures
I	**Fundamentals of Cyber World and Cyber Crimes** • Introduction to Cyber World • Cyber Law in India • Cyber Crime vs Conventional Crime • Cyber Fraud and Financial Crimes • Cyber Stalking, Identity Theft, Online Fraud	15
II	**Cyber Legal Framework and Digital Security in India** • Overview of General Laws in India • IT Act 2000 • Penalties and Offences • Digital Signature	15

	• Electronic Records and Legal Validity	

Textbook:
• Textbook on Cyber Law, Pavan Duggal, 2nd Edition Reprint, LexisNexis, 2023**.**
Additional References:
• Cyber Crimes & Laws, Sushma Arora & Raman Arora, 3rd Edition, Taxmann Publications, 2023.
• Introduction to Cyber Crime, Sunit Belapure & Nina Godbole, Latest Edition, Wiley India.
• Cyber Laws and IT Protection, Harish Chander, Latest Edition, PHI Learning.
• Cyber Law and Information Technology, Nandan Kamath, Latest Edition, Universal Law Publishing.

Course Code	SKILL ENHANCEMENT COURSE SEM – II	Credits	Lectures/Week
26CSVC251	**Advance Python Programming - II**	**2**	**2**

Course Objectives (CO):

CO1: Define fundamental concepts of Object-Oriented Programming, Python programming constructs, GUI components, database operations, and data science tools.
CO2: Explain the principles of OOP including classes, objects, inheritance, polymorphism, abstraction, exception handling, and their implementation in Python.
CO3: Apply Python programming skills to develop applications involving GUI design, database connectivity, and data analysis using relevant libraries.
CO4: Analyze real-world problems to design structured, efficient, and modular solutions using OOP concepts, exception handling, and data-driven approaches.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Describe the fundamental concepts of Object-Oriented Programming, Python constructs, GUI components, database systems, and data science tools.
OC2: Explain the implementation of OOP principles such as classes, objects, inheritance, polymorphism, abstraction, and exception handling in Python.
OC3: Apply Python programming skills to develop applications using GUI frameworks, perform database operations, and utilize libraries like NumPy, pandas, and Matplotlib for data handling.
OC4: Analyze problems to design efficient, modular, and scalable solutions using OOP concepts, database integration, exception handling, and data analysis techniques.

Unit	Topics	No of Lectures

I	**OOPs In Python:** Introduction to OOPs, Problems in Procedure Oriented Approach, Features of Object Oriented Programming System (OOPS), Constructors and Destructors. **Classes and Objects-** Creating a Class, Self-Variable, Types of Variables, Types of Methods, Passing Members of One Class to Another Class **Inheritance and Polymorphism:** Types of Inheritance, Constructors in Inheritance, Overriding Super Class Constructors and Methods, super() method, Polymorphism, Duck Typing , Operator Overloading, Method Overloading, Method Overriding **Abstract Classes and Interfaces**: Abstract Class, Abstract Method, Interfaces in Python	**15**
II	**Graphical User Interface:** Creating a GUI in Python, Widget classes, Layout Manager, Event Handling **Working with Databases:** DBMS, working with MySQL Database-retrieving, inserting, deleting, updating rows from table, Creating Database Tables through Python **Exceptions:** Errors in a Python Program, Exceptions and Exceptions handling, **Data Science Tools:** Introduction to NumPy, Matplotlib, pandas, Scipy,	**15**

Textbooks:

- Practical Programming: An Introduction to Computer Science Using Python 3, Paul Gries , Jennifer Campbell, Jason Montojo, Pragmatic Bookshelf, 2nd Edition, 2014
- Programming through Python, M. T Savaliya, R. K. Maurya& G M Magar, Sybgen Learning India, 2020

Additional References:

- Python: The Complete Reference, Martin C. Brown, McGraw Hill, 2018
- Beginning Python: From Novice to Professional, Magnus Lie Hetland, Apress, 2017
- Programming in Python 3, Mark Summerfield, Pearson Education, 2nd Ed, 2018

Advance Python Programming - II Practical

Course Objectives (CO):

CO1: Define fundamental concepts of Object-Oriented Programming, Python programming constructs, GUI components, database operations, and data science tools.

CO2: Explain the principles of OOP including classes, objects, inheritance, polymorphism, abstraction, exception handling, and their implementation in Python. CO3: Apply Python programming skills to develop applications involving GUI design, database connectivity, and data analysis using relevant libraries. CO4: Analyze real-world problems to design structured, efficient, and modular solutions using OOP concepts, exception handling, and data-driven approaches.	
Course Outcomes (OC): After successful completion of this course, students would be able to OC1: Describe the fundamental concepts of Object-Oriented Programming, Python constructs, GUI components, database systems, and data science tools. OC2: Explain the implementation of OOP principles such as classes, objects, inheritance, polymorphism, abstraction, and exception handling in Python. OC3: Apply Python programming skills to develop applications using GUI frameworks, perform database operations, and utilize libraries like NumPy, pandas, and Matplotlib for data handling. OC4: Analyze problems to design efficient, modular, and scalable solutions using OOP concepts, database integration, exception handling, and data analysis techniques.	
1	**Constructors and Destructors:** 1. Write a program to create a class Employee with a constructor to initialize name and salary, and display the details. 2. Create a class Student with both default and parameterized constructors. 3. Demonstrate the use of a destructor by printing a message when an object is destroyed. 4. Write a program to count the number of objects created using a constructor. 5. Create a class Book and initialize its attributes using constructor chaining. 6. Write a program to show the order of execution of constructor and destructor.
2	**Classes and Objects:** 1. Create a class Car with attributes and methods to display details. 2. Write a program to demonstrate the use of self variable. 3. Create a class to demonstrate instance variables and class variables. 4. Write a program to demonstrate instance methods, class methods, and static methods. 5. Create two classes Engine and Car, and pass object of one class to another. 6. Design a class BankAccount with deposit and withdrawal methods. 7. Create a class Rectangle to calculate area and perimeter. 8. Design a class Student to store and display student information.
3	**Inheritance and Polymorphism:** 1. Implement single inheritance using classes Person and Student.

	2. Demonstrate multiple inheritance with classes Teacher, Researcher, and Professor. 3. Write a program to show multilevel inheritance using Animal → Mammal → Dog. 4. Implement hierarchical inheritance using a Shape base class.
4	**Write a program to demonstrate constructor execution in parent and child classes:** 1. Use super() to call parent class constructor and display values. 2. Override a method in child class and show difference in output. 3. Write a program where child class calls parent method using super().
5	**Design a class to represent a bank account. It includes the following members Data:** Members (Account Number, Name, Type of Account, Balance amount) Methods: 1. To give initial values for type of account and balance 2. To deposit an amount 3. To withdraw an amount 4. To display name and balance Initial value for Account Number and Name has to be taken from the command line. 2. Define a class 'Overload' which will overload the method 'add ( )' as int add (int x, int y) & int add (int a, int b, int c) which will return the sum x+y & a+b+c respectively. 3. Write a function Area( ) to find the area of a circle, triangle. If only one argument is specified it will calculate the area of the circle. If two argument is specified it will calculate the area of the triangle. Use the concept of function overloading.
6	**GUI Form with Event Handling:** Create a **Student Registration Form** using Tkinter with: 1. Widgets: Label, Entry, Button 2. Layout: grid() 3. Event Handling: On button click, display entered details in a label or popup

7	**GUI Calculator with Exception Handling:** Design a **Calculator GUI** that: 1. Takes two numbers as input 2. Performs operations (+, −, ×, ÷) 3. Handles errors like division by zero using try-except
8	**MySQL Database CRUD Operations using Python:** Write a **Python program** to: 1. Connect to MySQL database 2. Create a table (Student/Employee) 3. Insert, retrieve, update, and delete records 4. Display records in tabular format
9	**Database-Driven GUI Application:** Create a **GUI-based Student Management System** that: 1. Takes input from GUI form 2. Stores data in MySQL database 3. Displays records on button click 4. Handles database errors using exception handling
10	**Data Analysis and Visualization:** Using a sample dataset (CSV file): 1. Load data using pandas 2. Perform basic analysis (mean, filtering, grouping) 3. Use NumPy for numerical operations 4. Visualize data using Matplotlib (line/bar chart) 5. Apply one SciPy function (e.g., statistical calculation)

Course Code	VOCATIONAL SKILL COURSE SEM – II	Credits	Lectures/ Week
26CSSCE241	**Web Technologies**	**2**	**2**

Course Objectives (CO):

CO1: Frontend Proficiency: To equip students with the skills to structure web content using HTML5 and apply advanced styling techniques using CSS for responsive and aesthetic design.
CO2: Client-Side Dynamism: To teach the fundamentals of JavaScript and the Document Object Model (DOM) to create interactive user interfaces and perform client-side validations.
CO3: Data Interchange & Asynchrony: To introduce XML and AJAX technologies for structured data representation and seamless, non-blocking communication between the client and server.
CO4: Server-Side Logic: To develop a comprehensive understanding of PHP for backend processing, session management, and database connectivity to build functional web applications.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1: Build well-structured web pages using HTML5 elements, multimedia, and CSS properties to manage layout, typography, and positioning.
OC2: Compare and contrast XML and HTML structures, implementing DTDs and XSLT to ensure data integrity and transformation across web applications.
OC3: Implement asynchronous web features using AJAX and JavaScript Browser Objects to evaluate and improve user experience and interface responsiveness.
OC4: Design and develop a dynamic web application using PHP that manages form data, handles file systems, maintains user sessions, and integrates with a database.

Unit	Topics	No of Lectures
I	Fundamental Elements of HTML, Formatting Text in HTML, Organizing Text in HTML, List Tags, Links and URLs in HTML, Tables in HTML, Images on a Web Page, Image Formats, Image Maps, Colors, Navigation across multiple pages, Forms in HTML, Interactive Elements, Working with Multimedia - Audio and Video File Formats, HTML elements for inserting Audio / Video on a web page. **CSS:** Understanding the Syntax of CSS, CSS Selectors, Inserting CSS in an HTML Document, CSS properties to work with background of a Page, CSS properties to work with Fonts and Text Styles, CSS properties for positioning an element.	**15**

	JavaScript: Using JavaScript in an HTML Document, Programming, Fundamentals of JavaScript – Variables, Operators, Control Flow Statements, Popup Boxes, Functions – Defining and Invoking a Function, Defining Function arguments, defining a return Statement, Calling Functions with Timer, JavaScript Objects - String, RegExp, Math, Date, Browser Objects - Window, Navigator, History, Location, Document, Cookies, Document Object Model, Form Validation using JavaScript	
II	**XML:** Comparing XML with HTML, Advantages and Disadvantages of XML, Structure of an XML Document, XML Entity References, with Internal / External DTD, XSLT Elements and Attributes **AJAX:** AJAX Web Application Model, How AJAX Works, XMLHttpRequest Object – Properties and Methods, Handling asynchronous requests using AJAX e.g. Mouseover, button click, **PHP:** Variables and Operators, Retrieving data from HTML forms, Program Flow, Arrays, working with Files and Directories, working with Databases, Working with Cookies, Sessions, and Headers	**15**

Textbooks:

- HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX, PHP and jQuery, 2ed, Dreamtech Press, 2016
- Web Programming and Interactive Technologies, scriptDemics, StarEdu Solutions India, 2018
- PHP: A Beginners Guide, Vikram Vaswani, TMH

Additional References:

- HTML, XHTML, and CSS Bible Fifth Edition, Steven M. Schafer, WILEY, 2011
- Learning PHP, MySQL, JavaScript, CSS & HTML5, Robin Nixon, O'Reilly, 2018
- PHP, MySQL, JavaScript & HTML5 All-in-one for Dummies, Steve Suehring, Janet Valade Wiley, 2018

Course Code	VOCATIONAL SKILL COURSE :SEM I	Credits	Practical/Week
26CSSEC241	Web Technologies Practical	2	2

Course Objectives (CO):

CO1: Frontend Proficiency: To equip students with the skills to structure web content using HTML5 and apply advanced styling techniques using CSS for responsive and aesthetic design.
CO2: Client-Side Dynamism: To teach the fundamentals of JavaScript and the Document Object Model (DOM) to create interactive user interfaces and perform client-side validations.
CO3: Data Interchange & Asynchrony: To introduce XML and AJAX technologies for structured data representation and seamless, non-blocking communication between the client and server.
CO4: Server-Side Logic**:** To develop a comprehensive understanding of PHP for backend processing, session management, and database connectivity to build functional web applications.

Course Outcomes:

After successful completion of this course, students would be able to
CO1: Learn PHP and XML Format
CO2: Use server side scripting with PHP to generate the web pages dynamically using the database connectivity.
CO3: Apply the HTML and CSS features with different layouts as per need of applications Use the JavaScript to develop the dynamic web pages.
CO4: Analyze the concepts of the World Wide Web, and the requirements of effective web design

Practicals:

1	Design a web page using different text formatting tags. Design a web page with links to different pages and allow navigation between web pages.
2	Design a web page that automatically redirects the user to another page. Design a web page with different tables.
3	Design a web page embedding image, audio and video. Design a web page with Imagemaps.
4	Design a web page with a form that uses all types of controls. Design a web page demonstrating different semantics.
5	Design a web page demonstrating different stylesheet types. Design a web page demonstrating grouping selectors.
6	Using JavaScript, design a web page to accept a number from the user and print its Factorial.

7	Using JavaScript, a web page that prints Fibonacci series/any given series
8	Write a JavaScript program to display all the prime numbers between 1 and 100.
9	Write a JavaScript program to accept a number from the user and display the sum of its digits.
10	Write a JavaScript program to design a simple calculator.
11	Design a form and validate all the controls placed on the form using JavaScript.

Course Code	**SKILL ENHANCEMENT COURSE SEM – II**	**Credits**	**Lectures/Week**
26CSSE252	**Web development using WordPress - II**	**2**	**2**

Course Objectives (CO):

CO1: Define and describe the fundamentals of web development and the role of WordPress as a content management system.
CO2: Explain the process of installing, configuring, and managing a WordPress website in a local environment.
CO3: Apply themes, plugins, and content creation techniques to design and customize websites.
CO4: Analyze website performance, usability, and basic security practices for effective website management.

Course Outcomes (OC):

After successful completion of this course, students would be able to
OC1:Remember basic concepts of web development and WordPress components such as themes, plugins, and dashboards.
OC2: Understand how WordPress websites are created, configured, and managed.
OC3: Apply WordPress tools to build and customize a functional website without using paid services.
OC4: Analyze website structure, design choices, and basic SEO and security aspects for improvement.

Unit	**Topics**	**No of Lectures**
I	**Introduction and Setup of WordPress** ● Basics of websites, domains, hosting (concept only) and CMS	**15**

	● Introduction to WordPress and its features ● Local installation using XAMPP/WAMP (no subscription required) ● WordPress dashboard overview and settings ● Creating pages, posts, categories, and menus	
II	**Customization and Website Management** ● Themes: installation, customization, and layout design ● Plugins: installation and basic usage (forms, SEO, security) ● Media management (images, videos, galleries) ● Basic SEO concepts and website optimization ● Website security basics and backup	**15**

Textbooks:

- WordPress for Beginners (or WordPress All-in-One For Dummies), Lisa Sabin-Wilson, Latest Edition, Wiley Publishing.
- WordPress for Beginners 2023/2024, Dr. Andy Williams, 2023 Edition, Self-Published.
- WordPress Explained: Your Step-by-Step Guide to WordPress, Raelene Morey, Latest Edition, Independently Published.
- WordPress 24-Hour Trainer, George Plumley, 1st Edition, Wiley Publishing

Course Code	**SEM II –Computer Science (Practical)**	**Credits**	**Practical/Week**
26CSSE252	**Web Development using WordPress - II-Practical**	**2**	**4**

Course Objectives:

CO1: Identify and describe the basic tools, interface, and components of WordPress through hands-on exploration.
CO2: Explain and demonstrate the process of setting up and configuring a WordPress website in a local environment.
CO3: Apply themes, plugins, and content creation features to design and develop simple websites.
CO4: Analyze website structure, usability, and basic maintenance practices for effective website management.

Course Outcomes:

At the end of the practical course, students will be able to:

OC1: Remember and recognize WordPress tools, dashboard features, and website components during practical tasks. OC2: Understand the steps involved in installing, configuring, and managing a WordPress website. OC3: Apply practical skills to create, customize, and manage a functional website using free WordPress tools. OC4: Analyze and improve website design, content organization, and basic SEO/security practices through hands-on work.	
1	Install XAMPP/WAMP and set up a local WordPress website.
2	Log in to the WordPress dashboard and explore different menu options.
3	Create and publish a page and a blog post with proper formatting.
4	Create categories and organize posts under them.
5	Install and apply a free theme and customize its appearance.
6	Create a navigation menu and link it to pages.
7	Add images and media to posts and pages.
8	Install and activate a free plugin (e.g., contact form or SEO plugin).
9	Customize the website homepage using available theme options.
10	Perform basic website maintenance: update plugins/themes and take a backup (manual or plugin-based).

Evaluation Scheme for First Year 2026-2027 (UG)

under NEP 2020 (4 credits)

I. Internal Evaluation for Theory Courses – 40 Marks

1) Continuous Internal Assessment (CIA) Assignment - Tutorial/ Case Study/ Project / Presentations/ Group Discussion / Ind. Visit. – 20 marks

2) Continuous Internal Assessment (CIA) ONLINE / OFFLINE Unit Test – 20 marks - 30 Minutes

II. External Examination for Theory Courses – 60 Marks

Duration: 2 Hours

Theory question paper pattern:

Question	Based on	Marks
Q.1	Unit I	15
Q.2	Unit II	15
Q.3	Unit III	15
Q.4	Unit IV	15

- All questions shall be compulsory with internal choice within the questions.
- Each Question may be sub-divided into sub questions as a, b, c, d, etc. & the allocation of Marks depends on the weightage of the topic.

NOTE: To pass the examination, attendance is compulsory in both Internal & External Theory Examinations.

Evaluation Scheme for First Year 2026-2027 (UG)
under NEP 2020 (2 credits)

I. Internal Evaluation for Theory Courses – 20 Marks

1) Continuous Internal Assessment (CIA) Assignment - Tutorial/ Case Study/ Project / Presentations/ Group Discussion / Ind. Visit. – 10 marks

2) Continuous Internal Assessment (CIA) ONLINE / OFFLINE Unit Test – 10 marks - 15 Minutes

II. External Examination for Theory Courses – 30 Marks

Duration: 1 Hour

Theory question paper pattern:

Question	Based on	Marks
Q.1	Unit I	15
Q.2	Unit II	15

- All questions shall be compulsory with internal choice within the questions.
- Each Question may be sub-divided into sub questions as a, b, c, d, etc. & the allocation of Marks depends on the weightage of the topic.

NOTE: To pass the examination, attendance is compulsory in both Internal & External Theory Examinations.

FY 2026-2027 Evaluation pattern and Question paper pattern for Semester End Practical Examination of Major Course

Internal Continuous Assessment: 40% (20 Marks)	Semester End Examination: 60% (30 Marks)	Duration for End semester examination
Viva/ assignment/ objective question test /Small experiments (15 Marks), Overall performance (5 Marks) = 20 Marks	As per paper pattern	1 hr 30 mins

FY 2026-2027 Evaluation pattern and Question paper pattern for Semester End Practical Examination of Vocational Skill Courses and Skill Enhancement Courses

Internal Continuous Assessment: 40% (20 Marks)	Semester End Examination: 60% (30 Marks)	Duration for End semester examination
Viva/ assignment/ objective question test /Small experiments (15 Marks), Overall performance (5 Marks) = 20 Marks	As per paper pattern	1 hr 30 mins

Signatures of Team Members:

SR.No.	Name	Signature
1.	Dr. Manisha Divate	
2.	Prof. Neeta Kulkarni	
3.	Dr. Lakhichand Patil	
4.	Mr. Niraj R. Wani	
5.	Ms Akshaya Rane	
6	Dr. Akshata Nayak	
7	Dr. Prabha Kadam	
8	Dr. Sampada Margaj	
9	Prof. Shubhangi Page	
10	Dr. Siddhesh Kadam	
11	Prof. Komal Jadhav	
12	Prof. Manali Kurle	

Letter Grades and Grade Points:

Semester GPA/ Programme CGPA Semester/ Programme	% of Marks	Alpha-Sign/ Letter Grade Result	Grading Point
9.00 - 10.00	90.0 – 100	O (Outstanding)	10
8.00 - < 9.00	80.0 - < 90.0	A+ (Excellent)	9
7.00 - < 8.00	70.0 - < 80.0	A (Very Good)	8
6.00 - < 7.00	60.0 - < 70.0	B+ (Good)	7
5.50 - < 6.00	55.0 - < 60.0	B (Above Average)	6
5.00 - < 5.50	50.0 - < 55.0	C (Average)	5
4.00 - < 5.00	40.0 - < 50.0	P (Pass)	4
Below 4.00	Below 40.0	F (Fail)	0
Ab (Absent)	-	Ab (Absent)	0

Justification for Bachelors in Computer Science

1.	Necessity for starting the course:	Industry Requirement
2.	Whether the UGC has recommended the course:	**Yes**
3.	Whether all the courses have commenced from the academic year 2023-24.	The course has already commenced in the university and in the academic year 1999-2000 it is restructured under NEP 2020
4.	The courses started by the University are self-financed, whether adequate number of eligible permanent faculties are available?:	This course is self-financed based on the sanction given by University of Mumbai to affiliated colleges time to time.
5.	To give details regarding the duration of the Course and is it possible to compress the course?:	The duration of the program is three years (6 semesters). It is not possible to compress the course.
6.	The intake capacity of each course and no. of admissions given in the current academic year:	120 per division
7.	Opportunities of Employability / Employment available after undertaking these courses:	Software Developer, Business Analysts, etc..