St. Xavier's College (Autonomous), Ahmedabad

Syllabus of Semester – I of the following departments under Faculty of Computer Science based on Undergraduate Curriculum Framework to be implemented from the Academic Year 2025-26

DEPARTMENT OF COMPUTER SCIENCE

BSC-IT (Hons.) Category – IV

Major Course – 1: BSC-IT Programming Concepts with C

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course Title & Code	Credit Distribution of The Course			Eligibility Criteria	Pre-requisite(s) of the Course (if any)
	Lecture	Tutorial	Practical / Practice		
Programming Concepts with C (ITMC111C)	2	0	2		

Learning Objective:

The objective of this course is to:

- Introduce students to fundamental concepts of C programming and develop their problem-solving skills.
- Enable students to understand and apply data types, control structures, arrays, pointers, and functions in C.
- Provide hands-on experience with user-defined functions (UDFs), recursion, and memory management techniques using dynamic memory allocation.
- Equip students with the knowledge of file handling operations, including reading, writing, and manipulating data files.
- Develop an understanding of structures, unions, and advanced programming concepts in C.

Learning Outcomes :

By the end of the theory sessions, students will be able to:

- Understand the fundamental concepts of C programming, including syntax, keywords, and data types.
- Explain the working of control structures such as loops, conditional statements.
- Analyse memory management using pointers, pointer arithmetic, and pointer types.
- Differentiate between static and dynamic memory allocation.
- Describe functions, function calls, recursion, and parameter passing techniques
- Compare structures and unions, understanding their differences and use cases.
- Discuss the importance of file handling and its operations.
- Examine the concept of command-line arguments and their applications.
- Evaluate different storage classes (auto, extern, static, register) and their impact on memory and scope.

Unit	Unit Details	Hours
I	**1. Computer Programming Concepts** What is Programming ? - Types of programming languages (Machine, Assembly, High-Level) - Assembler, Compiler, Interpreter - History & Features of C. **2. Problem Solving Techniques** Problem-solving approach - Flowchart symbols & examples - Execution process of a C program. **3. Operators** Arithmetic – Relational – Logical – Bitwise - Assignment, etc. - Precedence & Associativity. **4. Strings in C** Declaration, Initialization, Input & Output - String Functions - gets() , puts(), getchar(), putchar(). **5. String Functions in C** strlen() , strcpy() , strcat() , strlwr(), strupr() , strrev() **6. Decision Making in C** if, if-else, if-else-if - switch-case - Nested decision statements **7. Looping in C** for, while, do-while - Nested loops - Control statements - break, continue, goto **8. Arrays** 1D array – Declaration – Initialization - Accessing elements - Array Operation - Searching and Sorting - Integer array - Character array - 2D array - Matrix representation - Matrix operations - Addition – Multiplication. **9. Pointers** Introduction – Definition – Declaration - Void Pointer - Pointer arithmetic - Chain of Pointers - Array of Pointers - Pointer to an Array **10. Functions in C** User-defined functions (UDF) – Types - Function prototype - Call by Value - Call by Reference - Recursion	15
II	**11. Structures in C** Definition – Declaration – Initialization - Structure within Structure - Array of Structures - Array within Structure - Passing structures to functions - Differences between Structure & Union **12. Storage Classes** - auto, extern, static, register **13. File Handling** What is File ? - Use of File - File Operations - File pointers - Working with multiple files **14. Introduction of Command line argument** **15. Dynamic Memory Allocation (DMA)** malloc(), calloc(), realloc(), free()	15

Text Books

- Programming in ANSI C. (6^{th} Ed.) – Balagurusami - Tata McGraw Hill Publication

References Books

- Programming In C (2nd Ed.) - Ashok N. Kamthane - Pearson Education
- The C Programming Language - DENNIS M. RITCHIE- AT&T Bell Laboratories Murray Hill, New Jersey
- Let us C – (15th Ed.) - Yashwant Kanetkar - BPB Publications
- Programming in C - Reema Thareja - Oxford University Press

SEMESTER – I		
BSC IT	**Programming Concepts with C - Practical**	**Credit 2**

Course Objectives:

The objective of this course is to:

- Introduce students to fundamental concepts of C programming and develop their problem-solving skills.
- Enable students to understand and apply data types, control structures, arrays, pointers, and functions in C.
- Provide hands-on experience with user-defined functions (UDFs), recursion, and memory management techniques using dynamic memory allocation.
- Equip students with the knowledge of file handling operations, including reading, writing, and manipulating data files.
- Develop an understanding of structures, unions, and advanced programming concepts in C.

Learning Objectives :

By the end of the practical sessions, students will be able to:

- Write and execute basic C programs using variables, operators, and control statements.
- Develop programs using arrays (1D & 2D), character arrays, and string handling functions.
- Implement pointer-based operations, including pointer to variables, pointer arithmetic, and array of pointers.
- Demonstrate the use of user-defined functions (UDFs), recursive functions, and function pointers.
- Apply dynamic memory allocation functions to manage runtime memory efficiently.
- Use structures, nested structures, and arrays of structures to store and manipulate data.
- Perform file handling operations like reading, writing, appending, and modifying files.
- Solve real-world problems using C programming concepts in a structured and optimized manner.

Unit	Unit Details	Hours
I	1. Introduction to Editor - Structure of 'C' Program - Compile , Debug and Execute the Program - Escape sequences – Comments - main(), getch(), clrscr() , printf() 2. Character set – Delimiters – Tokens - Key words – Identifiers – Constants 3. Data Types – Variables - Rules for defining variables - Type casting 4. Declaration and initialization of variables - Inbuilt functions – scanf() - sizeof operator 5. Implementation of operators - Evaluation of expression 6. Implementation of string and its functions 7. Implementation of if, if-else, if-else-if 8. Implementation of switch case and nested decision statements 9. Implementation of for, while, do-while 10. Implementation of nested loop and control statements 11. Implementation of 1D array 12. Implementation of 2D array 13. Implementation of 2D matrix and operations 14. Patterns	30
II	15. Use of pointer - Implementation of pointer - Implementation of array of pointer - Implementation of pointer to an array - Implementation of functions 16. Implementation of structure 17. Use of storage classes in programming 18. Implementation of File functions - fopen(), fclose(),fread(), fwrite(),fprintf(), fscanf() , fgetc(), fputc()	30

	19. Working with Multiple Files 20. Use of command line argument DMA Implementation through 1D array	

Text Books

- Programming in ANSI C. (6th Ed.) – Balagurusami - Tata McGraw Hill Publication

References Books

- Programming In C (2nd Ed.) - Ashok N. Kamthane - Pearson Education
- The C Programming Language - DENNIS M. RITCHIE- AT&T Bell Laboratories Murray Hill, New Jersey
- Let us C – (15th Ed.) - Yashwant Kanetkar - BPB Publications
- Programming in C - Reema Thareja - Oxford University Press

Practical List

Theory Session	Practical Session	Programs
1	1	1. Program to demonstrate the use of escape sequences in C. 2. Program to add and display single-line and multi-line comments in C. 3. Program to print a formatted output using printf() and escape sequences. 4. Program to use getch() and clrscr() to control screen output and user input. 5. Write a C program to input a string and print each character on a new line. 6. Write a C program to input a string and print it in pyramid form.
	2	7. Program to display structured information using printf(), escape sequences, and comments. 8. Program to declare and print different types of constants (integer, float, character, and string).
2	3	9. Program to demonstrate different data types in C by declaring and printing variables of each type. 10. Program to declare variables of various data types and display their sizes using the sizeof operator. 11. Program to show valid and invalid variable names based on C variable naming rules. 12. Program to demonstrate type conversion (implicit and explicit) in C.
	4	13. Program to accept values of different data types from the user and display them. 14. Program to declare variables of various data types and display their sizes using the sizeof operator.
3	5	15. Write a program to compare two user-input numbers using relational operators and display whether the first number is greater than, less than, or equal to the second number. 16. Write a program to evaluate the mathematical expression (a + b) * c / d using user-input values for a, b, c, and d, and display the result. 17. Write a program to demonstrate the difference between pre-increment (++a) and post-increment (a++) by displaying values before and after the operations.
	6	18. Write a C program to evaluate the following expression by taking user input for all variables and applying the BODMAS rule: [result=(a+b)∗c/d−e%f+g∗(h−i)/j]

		19. Write a C program to take two integer inputs from the user and perform the following bitwise operations: • Bitwise AND (&) to find common set bits. • Bitwise OR (|) to combine bits. • Bitwise XOR (^) to find differing bits. • Left Shift (<<) to multiply the first number by 2. • Right Shift (>>) to divide the second number by 2.
4	**7**	20. Write a C program to declare and initialize a string using character arrays and print it. 21. Write a C program to input a string from the user and display it. 22. Write a C program to input a string and print it in reverse order without using function. 23. Write a C program to input a string using scanf() and gets(), then display both.
	8	24. Write a C program to input a string using gets() and display it using puts(). 25. Write a C program to read a single character using getchar() and print it using putchar(). 26. Write a C program to input a sentence using gets() and display each character using putchar().
5	**9**	27. Write a C program to find the length of a string without using the strlen() function. 28. Write a C program to copy one string into another without using the strcpy() function.
	10	29. Write a C program to concatenate two strings using strcat(). 30. Take one string from the user and apply string fucntions - strlwr() , strupr() ,strrev(). 31. Write a C program that takes a string, converts it to lowercase, reverses it, and then concatenates it with the original string.
6	**11**	32. Write a C program to check if a number is positive using an if statement. 33. Write a C program to check whether a number is even or odd using if-else. 34. Write a C program to find the largest of two numbers using if-else. 35. Write a C program to categorize a person's age group using if-else-if.
	12	36. Write a C program to calculate the total marks, percentage, and grade of a student based on five subject marks using if-else-if conditions. 37. Write a C program to implement an ATM system where a user can check balance, deposit money, or withdraw money using if-else conditions. 38. Write a C program to determine whether a person is eligible to donate blood. The conditions are: • The person must be at least 18 years old. • If the person is above 18, check if their weight is 50 kg or more. • If both conditions are met, print "You are eligible to donate blood." • If age is less than 18, print "You are not eligible due to age." • If weight is less than 50 kg, print "You are not eligible due to weight."
7	**13**	39. Write a C program that takes a single character as input and checks whether it is a vowel (a, e, i, o, u) or a consonant using a switch-case statement. 40. Write a C program that takes a number (1-12) as input and prints the corresponding month name using a switch-case statement.
	14	41. Write a C program to create a basic calculator that performs addition, subtraction, multiplication, or division based on user input. • Use switch-case to select the operation. • Use a nested if inside the division case to prevent division by zero. 42. Write a C program to determine ticket price based on age group and seat type. • Use switch-case for seat type (1 for Regular, 2 for VIP).

		• Inside each case, use a nested if to check age (< 12 for child discount, >= 60 for senior discount).
8	**15**	43. Write a C program to print the first 10 natural numbers (1 to 10) using for loop, while loop and do-while loop. 44. Print numbers from 10 to 1 using for, while, and do-while loops. 45. Print addition of first 10 numbers. 46. Input a string and calculate its length without using built-in functions. 47. Check entered number is prime or not.
	16	48. Calculate and print the sum of the first N natural numbers using for, while, and do-while loops. 49. Compute the factorial of a given number using different loop structures. 50. Print all even and odd numbers up to N using a loop. 51. Write a program to concatenate two strings manually using loops. 52. Check the entered year is a leap year or not. 53. Take 3 digit number from user and print sum of digits.
9	**17**	54. Take a number as input and print its multiplication table using loops. 55. Write a C program to input a string and count the number of vowels and consonants. 56. Input a string and print its reverse manually using a loop. 57. Input a sentence and count the number of words (consider words separated by spaces).
	18	58. Print binary of entered integer number. 59. Write a program that converts all uppercase letters in a string to lowercase and vice versa without using built-in functions. 60. Input a string and check if it reads the same forward and backward (e.g., "madam" is a palindrome). 61. Input a string and a character, then count how many times the character appears in the string.
10	**19**	62. Write a C program to take numbers as input from the user and print them. If the user enters -1, the loop should terminate using break. 63. Write a C program to print numbers from 1 to 10, but skip even numbers using continue. 64. Write a C program that asks the user to enter a number. If the number is negative, use goto to prompt the user to enter the number again. 65. Print table of 1 to 5.
	20	66. Print table of 1 to 10 67. Print patterns : * ** *** **** --- * ** *** **** --- A BC DEF GHIJ --- A BB CCC DDDD --- H HE

		HEL HELL HELLO --- A bC dEf GhIj
11	**21**	68. Write a C program to declare an array of size 5, initialize it with some values, and print all elements. 69. Write a C program to input 10 numbers in an array and display them. 70. Write a C program to store 5 numbers in an array and print the first and last elements. 71. Write a C program to input 5 numbers in an array and print their sum.
	22	72. Write a C program to input 10 numbers in an array and find the largest and smallest number. 73. Write a C program to input 5 numbers in an array, then insert a new number at a specified position and display the updated array. 74. Write a C program to input 5 numbers in an array, remove an element at a specified position, and print the updated array.
12	**23**	75. Write a C program to input two arrays of size N and merge them into a single array. 76. Write a C program to copy the elements of one array into another array.
	24	77. Write a C program to sort an array. Print both ascending and descending order. 78. Search for the specific element in the array and print how many times element occurs and at which location.
13	**25**	79. Take one sentence from the user and also take one word from the user. Now check that word is present in the sentence or not. 80. Print the frequency of the word in the given sentence.
	26	81. Write a program to accept a string (character array) from the user and count the occurrences of each digit (0-9) present in the string using an integer array. 82. Take a roll number as an input and extract numbers, characters and special characters. Store them in separate arrays.
14	**27**	83. Take values of 3x3 matrix from the user and print them in matrix style. 84. Take row and column values from user and then take values for the same and finally print them on screen. 85. Initialize rows, columns and values for two matrices and print both the matrices separately.
	28	86. Write a program to find the transpose of a given matrix (swap rows and columns). 87. Write a program to find and display the sum of each row and each column of a matrix.
15	**29**	88. Write a program to accept two 3×3 matrices from the user and perform matrix addition. 89. Write a program to accept and display n names using a 2D character array.
	30	90. Write a program to multiply two matrices (check compatibility before multiplication). 91. Store numbers in a 2D array to create an X shape. 1 1 2 2 3 4 4 5 5

16	**31**	92. Declare an integer variable and a pointer to it. 93. Write a program to demonstrate basic pointer operations (declare, assign, and print address & value of a variable using a pointer). 94. Write a program to declare and use pointers for different data types (int, float, char).
	32	95. Write a program to demonstrate the use of a void pointer by assigning different data types to it. 96. Declare 3 pointers for three different variables. Initialize variables with integer values and then print those values through pointers.
17	**33**	97. Write a program to illustrate pointer arithmetic (increment and decrement of pointers). 98. Write a program to demonstrate pointer arithmetic (addition, and subtraction). 99. Write a program to swap two numbers using pointers.
	34	100. Write a program to demonstrate a chain of pointers (pointer to pointer) in C. 101. Write a program to reverse an array using pointers.
18	**35**	102. Write a program to store and display multiple strings using an array of pointers. 103. Write a program to access array elements using a pointer to an array.
	36	104. Write a program to count even and odd numbers in an array using pointers. 105. Write a program to calculate the length of a string using pointers (without using strlen()).
19	**37**	106. Program to show how to declare , define and call function. 107. Addition of two numbers using all four UDF categories.
	38	108. Addition of two numbers using all four UDF categories.
20	**39**	109. Write a function sum(int *a, int *b, int *result) that calculates the sum of two numbers using pointers. 110. Print Fibonacci series through function.
	40	111. Write a function int gcd(int *a, int *b) to find the greatest common divisor (GCD) of two numbers. 112. Write a function int lcm(int *a, int *b) that calculates the least common multiple (LCM) of two numbers.
21	**41**	113. Print the square of a number [void squareByValue(int num)] (Call by Value). 114. Modify the original number to store its square [void squareByReference(int *num)] (Call by Reference).
	42	115. Write a C program to check whether three given numbers form a Pythagorean triplet. A Pythagorean triplet consists of three positive integers (a, b, c) that satisfy the equation: a2+b2=c2 The program should take three integers as input and display whether they form a Pythagorean triplet or not.
22	**43**	116. Write a C program to convert temperature between Celsius and Fahrenheit. If temperature is in Celsius then convert it into Fahrenheit and vice-versa. Create two functions celsius() and fahrenheit(). The program should: • Take a temperature value as input. • Ask the user to choose between: • Converting Celsius to Fahrenheit using the formula: $F=(C\times9/5)+32$ • Converting Fahrenheit to Celsius using the formula: $C=(F-32)\times5/9$ • Display the converted temperature value.

		117. Write a C program to print the Fibonacci series using recursion.
	44	118. Write a C program to calculate the factorial of a number using recursion.
23	**45**	119. Define, declare and initialize the structure. 120. Write a C program to define a structure for a student with the following details: [Roll Number (integer) Name (string) Marks (float)]. The program should: • Declare a structure named Student. • Take user input for one student's details. • Display the entered details. 121. Write a C program to declare and initialize an Employee structure with: [Employee ID (integer) Name (string) Salary (float)]. The program should: • Declare a structure named Employee. • Initialize it with sample data directly in the program. • Print the stored employee details.
	46	122. Take details of a student (roll no , full name , gender , percentage). Initlaize all the values and print in tabular format.
24	**47**	123. Write a C program to define a structure "Date" (containing day, month, and year) and use it inside another structure "Person" (containing name and birthdate).
	48	124. Write a C program to define a structure "Address" (containing house number, city, and pin code) and use it inside another structure "Employee" (containing name, ID, and address). 125. Write a C program to define a structure "Author" (containing author name and nationality) and use it inside another structure "Book" (containing book title, price, and author details).
25	**49**	126. Write a C program to use an array of structures to store and display information of 5 students (Roll No, Name, Marks).
	50	127. Write a C program to define a structure "Student" that stores: • Name (string) • Roll Number (integer) • Marks in 5 subjects (array of 5 integers)
26	**51**	128. Write a C program to define a structure "Student" (containing name, roll number, and marks). Create a function that: • Takes the structure as an argument (pass by value). • Displays the student details inside the function. 129. Write a C program to define a structure "Employee" (containing employee ID, name, and salary). Create a function that: • Takes the structure as a pointer (pass by reference). • Increases the salary by 10%. • Displays the updated salary.
	52	130. Write a C program to define a structure "Rectangle" (containing length and width). Create: • A function that takes user input and returns a structure of type "Rectangle". • Another function that calculates and displays the area using the structure. 131. Write a C program to define a structure "Cricketer" (containing name, matches played, and total runs). Create a function that: • Takes an array of structures as an argument. • Calculates and displays the average runs of all players.
27	**53**	132. Write a C program to define a structure and a union with the following fields: [id (integer) name (character array of size 20) salary (float)] • Declare and initialize a structure and a union with sample data. • Display the size of both structure and union using sizeof(). • Show how memory sharing works in a union.

	54	133. Write a C program to demonstrate the difference between auto, extern, static, and register storage classes. • auto: Declare a local variable inside a function and print its value. • extern: Declare a global variable and access it in multiple functions. • static: Declare a static variable inside a function and show how its value persists across function calls. • register: Declare a register variable and print its value.
28	**55**	134. Write a C program that asks the user to enter their name, age, and city, writes this data to a file "info.txt", and then reads it back and displays it. 135. Write a program to copy content from one file (source.txt) to another (destination.txt) using fgetc() and fputc().
	56	136. Write a program that opens "data.txt" in append mode ("a") and allows the user to add more text to it. 137. Write a program that reads a file and counts the number of words, lines, and characters.
29	**57**	138. Write a program to read a file and write its content in reverse order to another file. 139. Write a program to merge the contents of "file1.txt" and "file2.txt" into "merged.txt".
	58	140. Write a C program that accepts two integers as command-line arguments and prints their sum. 141. Write a C program that accepts a string as a command-line argument and prints its length.
30	**59**	142. Write a C program that: Asks the user for the size of an integer array. Dynamically allocates memory using malloc(). Takes input and displays the array elements. Frees the allocated memory using free(). 143. Write a C program that: Takes the size of an integer array from the user. Allocates memory using calloc(). Prints the default initialized values (should be 0). Takes input and displays the array elements. Frees the memory using free().
	60	144. Write a C program that: Dynamically allocates an integer array using malloc(). Fills the array with user input. Asks the user whether they want to increase the size. If yes, uses realloc() to expand the array and takes additional inputs. Displays the final array and frees memory. 145. Write a C program that: Allocates memory for a 1D integer array using malloc(). Takes n elements as input from the user. Computes and prints the sum of all elements. Frees the allocated memory.

*****

St. Xavier's College (Autonomous), Ahmedabad

Syllabus of Semester – I of the following Department under Faculty of Computer Science based on Undergraduate Curriculum Framework to be implemented from the Academic Year 2025-26.

DEPARTMENT OF COMPUTER SCIENCE

BSc IT (Hons.) Category – IV

Major Course – 2: Computing Principles and Emerging Trends

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course Title & Code	Credit Distribution of The Course			Eligibility Criteria	Pre-requisite(s) of the Course (if any)
	Lecture	Tutorial	Practical / Practice		
Computing Principles and Emerging Trends (ITMC112C)	4	0	0	Nil	Nil

Learning Objectives:

The objective of this course is to provide students with a fundamental understanding of computer systems, including their architecture, memory, storage, and input/output devices. Students will learn number systems, logic gates, and Boolean functions, enabling them to analyse digital circuits. Additionally, the course introduces emerging technologies such as AI, IoT, Blockchain, Cloud Computing, and Quantum Computing, emphasizing their real-world applications and ethical considerations. By the end of the course, students will be equipped with both foundational computing knowledge and insights into the latest technological advancements.

Learning Outcomes:

Upon successful completion, students will be able to implement:

- Describe how computers process, store, and retrieve information.
- Demonstrate number system conversions and encoding techniques.
- Analyze logic circuits using Boolean functions and logic gates.
- Explain the purpose and functionality of different input/output devices.
- Evaluate the impact of AI, IoT, and Blockchain in real-world scenarios.
- Compare cloud computing models (IaaS, PaaS, SaaS) and their applications.
- Understand the fundamental working principles of Quantum Computing.
- Critically assess ethical considerations in AI and emerging technologies.

UNIT	Particulars	Hours
I	**Understanding the Computer and its Organisation and Architecture** • Classification of Computers • Computing Concepts • The Computer System and its applications **Central Processing** • Internal Communications, Machine Cycle	15

	• The Bus • Instruction Set **Memory and Storage Systems** • Memory Representation • Random Access Memory • Read Only Memory **Storage Systems** • Magnetic Storage Systems • Optical Storage Systems • Magneto Optical Systems • Solid-state Storage Devices **Input Devices** • Keyboard • Pointing Devices • Scanning Devices • Optical Recognition Devices • Digital Camera • Voice Recognition System • Data Acquisition Sensors • Media Input Devices **Output Devices** • Display Monitors • Printers • Impact Printers • Non-impact Printers • Plotters • Voice Output Systems • Projectors • Terminals	
2	**Computer Codes** Introduction to Computer Codes: Decimal System-Binary System Hexadecimal System-Octal System-4-bit BCD System-8-bit BCD System ASCII code-16-bit Unicode Conversion of Numbers (includes fixed and fractional number): Non decimal to decimal - Binary to decimal - Decimal to Binary - Octal to Binary – Octal to Decimal - Decimal to Octal - Binary to Hexadecimal - Hexadecimal to Binary - Hexadecimal to Decimal - Decimal to Hexadecimal - Hexadecimal to Octal - Octal to Hexadecimal	15
3	**Logic Gates and Digital Circuits** • Introduction • Basic Logic Gates • Derived Logic Gates • Conversion of Boolean Functions • Adder Circuits • Flip-flop Circuits • Application of Flip-flops **Venn Diagrams & Logical Relationships** • Basic Set Operations using Venn Diagrams • Application in Computer Science	15
4	**Overview of Emerging Technologies** **Artificial Intelligence (AI) & Machine Learning (ML)**	15

	• What is AI? Types of AI (Narrow, General, Super AI) • Machine Learning vs. Deep Learning • Applications of AI & ML (Healthcare, Finance, Automation) **Internet of Things (IoT)** • Concept of IoT & Smart Devices • IoT Architecture & Communication (Sensors, Networks, Cloud) • Real-World Applications (Smart Homes, Smart Cities, Healthcare) **Cloud Computing** • Introduction to Cloud Computing • Types of Cloud Services: IaaS, PaaS, SaaS • Benefits and Challenges of Cloud Computing **Blockchain Technology** • Basics of Blockchain • How Blockchain Works: Hashing, Mining, Consensus Mechanisms • Applications: Cryptocurrency, Smart Contracts, Digital Identity **Quantum Computing** • Basics of Quantum Mechanics & Computing Principles • How Quantum Computers Differ from Classical Computers • Potential Applications in Cryptography, AI, and Drug Discovery **Augmented Reality (AR) & Virtual Reality (VR)** • Differences Between AR & VR • Applications in Gaming, Healthcare, Education, and Training **Ethical AI & Responsible Tech Development** • Bias in AI and Ethical Concerns • The Role of Regulations in Emerging Tech • Sustainable & Green Computing	

Textbook:

- Fundamentals of Computer - Publisher – Balaguruswamy- McGraw-Hill

Reference Books:

- **Emerging Technologies for Business Professionals"** *– Nir Kshetri*
- **Handbook of Emerging Technologies for Learning"** *– George Siemens,*
- Technology Convergence and Emerging Industries - Tugrul Daim