JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJADA

VIZIANAGARAM

VIZIANAGARAM-535 003, A.P

(Established by Andhra Pradesh Act No.22 of 2021)

I-B. Tech INFORMATION TECHNOLOGY COURSE STRUCTURE

Common to

CSE (AI&DS), AI&DS, CSE(AI&ML), AI&ML, CSE (AI), AI, CSE (DS), DS, CSE (CS), CS, CSE (IoT& CSBT), CSE(IoT)

I B. Tech I Semester

S.No	Category	Course Code	Course Title	L	T	P	C
1.	HSMC		Communicative English	3	0	0	3
2.	BSC		Linear Algebra And Calculus	3	0	0	3
3.	BSC		Advanced Physics	3	0	0	3
4.	ESC		Computational Thinking & Problem Solving with Python	3	0	0	3
5.	HSMC Lab		Communicative English Lab	0	0	3	1.5
6.	BSC Lab		Advanced Physics Lab	0	0	3	1.5
7.	ESC Lab		Computational Thinking & Problem Solving with Python Lab	0	0	3	1.5
8.	ESC		Design Thinking &Innovations	0	1	1	1.5
9.	ESC		AI Tools & Applications	0	1	2	2
10.	MC		NSS / NCC / Scouts and Guides / Community Service	0	0	2	0
			Total	**12**	**2**	**15**	**20**

G. Jayakara

I B. Tech II Semester

S.No	Category	Course Code	Course Title	L	T	P	C
1.	BSC		Differential Equations And Vector Calculus	3	0	0	3
2.	BSC		Advanced Chemistry	3	0	0	3
3.	ESC		Data structures through C	3	0	0	3
4.	PCC		Digital logic Design	3	0	0	3
5.	HSMC		Universal human values and Professional Ethics	2	0	0	2
6.	BSC		Fundamentals of Quantum Computing	3	0	0	3
7.	ESC Lab		C& Data Structures lab	0	0	2	1
8.	ESC		Makers Lab -I	0	1	2	2
9.	MC		Environmental Science	2	0	0	0
			Total	**19**	**1**	**4**	**20**

G. Jayasuma

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

Computational Thinking and Problem Solving with Python

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Computational Thinking and Problem Solving with Python	ES C	3-0-0	3

Pre-requisite: Intermediate (10+2) passed. No prior programming experience is assumed.

Course Outline

This course positions computational thinking — decomposition, pattern recognition, abstraction and algorithmic reasoning — as the discipline running through every unit, with Python as the language in which that reasoning is implemented and tested. Unit I builds the foundation through algorithm design and flowcharting; each subsequent unit connects new Python constructs back to a specific computational-thinking skill: recognising repetition as pattern recognition, choosing a data structure as abstraction, breaking a program into functions as decomposition. A parallel good-practice thread introduces debugging discipline, documentation conventions and basic version control alongside the core content.

Course Objectives

1. To establish computational thinking — decomposition, pattern recognition, abstraction and algorithmic reasoning — as the foundation for problem solving, grounded in algorithm design and flowchart construction.
2. To develop proficiency in Python control structures, explicitly linking conditionals to abstraction and loops to pattern recognition.
3. To build competence in selecting Python data structures as an exercise in abstraction and pattern-based organisation of information.
4. To train students to decompose problems into functions and recursive sub-problems and to use AI coding assistants responsibly.
5. To equip students with file handling, exception handling and basic data analysis and visualisation skills.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Explain the principles of computational thinking and interpret algorithms and flowcharts that represent problem-solving logic.	K2
CO2	Apply pattern recognition and abstraction, through Python control structures, to implement step-by-step algorithmic solutions.	K3
CO3	Analyse problems to select and manipulate appropriate data structures — strings, lists, tuples and dictionaries — as an abstraction exercise.	K4
CO4	Evaluate the correctness and efficiency of modular Python programs through tracing, testing, debugging and exception handling.	K5
CO5	Create modular, well-documented Python programs using functions, recursion, file handling and data analysis to solve engineering problems.	K6

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	2	1	1	2	-	-	-	1	1	2	1	1

CO2	3	3	2	2	3	-	-	-	1	1	3	2	1
CO3	3	3	2	2	3	-	-	-	1	1	3	2	1
CO4	3	3	2	3	3	-	-	1	2	1	3	2	2
CO5	3	3	3	3	3	1	1	1	2	2	3	3	2

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

Syllabus

UNIT	CONTENTS	Contact Hours
UNIT – I	**Computational Thinking, Algorithm Design and Flowcharting:** Computational thinking pillars — decomposition, pattern recognition, abstraction and algorithmic thinking; the problem-solving process. Algorithm design — characteristics of a good algorithm, stepwise refinement, pseudocode. Flowcharts — standard symbols, construction rules and manual tracing for verification. Translating algorithms and flowcharts into Python — the Python interpreter, variables, data types, operators, expressions and input/output. Good coding practice — docstrings, inline comments, the assert statement and print-based debugging.	9
UNIT – II	**Control Structures:** Conditional statements — if, if-else and if-elif-else as abstraction over multiple cases; nested conditionals. Loops — for and while, as the recognition of repetitive patterns; range and iteration protocols; nested loops; loop control statements — break, continue and pass; the else clause on loops. Common algorithmic patterns — accumulator, counter, search and sentinel-controlled loops; tracing loop execution and identifying off-by-one errors.	10
UNIT – III	**Data Structures — Strings, Lists, Tuples, Sets and Dictionaries:** Strings as sequences — indexing, slicing, immutability, string methods and formatting; pattern matching in text. Lists — creation, indexing, slicing, mutation, list methods, list comprehensions, nested lists and matrix representation. Tuples — immutability, packing and unpacking. Sets — operations and use in de-duplication. Dictionaries — key-value abstraction, methods, iteration, nested dictionaries; choosing the appropriate data structure as an abstraction decision.	10
UNIT – IV	**Functions and Modular Programming:** Functions as decomposition — definition, arguments and return values; default, keyword and variable-length arguments; scope and lifetime of variables; lambda functions; map, filter and reduce. Recursion — algorithmic thinking applied to self-similar sub-problems; the call stack; recursive versus iterative solutions and informal comparison of their time and memory costs. Modules and packages; standard library modules — math, random, datetime, os; creating user-defined modules; responsible use of AI coding assistants for code review and explanation.	9
UNIT – V	**File Handling, Exception Handling and Data Analysis:** File handling — text and binary files, opening modes, reading and writing, the with statement, working with CSV files; decomposing a data workflow into read, process and write stages. Exception handling — errors versus exceptions, try-except-else-finally, built-in exceptions, raising and defining custom exceptions. Introduction to data analysis and visualisation — NumPy arrays and operations, Pandas Series and DataFrames, reading and cleaning data, descriptive statistics, and plotting with Matplotlib; interpreting patterns in real datasets.	10
	Total	**48**

Learning Resources

G. Jayasuma

Text Books

1. Downey, A. B. (2015). Think Python: How to think like a computer scientist (2nd ed.). O'Reilly Media.

2. Guttag, J. V. (2021). Introduction to computation and programming using Python: With application to computational modeling and understanding data (3rd ed.). MIT Press.

3. Reema Thareja. (2019). Python programming using problem solving approach. Oxford University Press.

4. Anita Goel, Python Programming, First Edition: An Object-Oriented Approach, Universities Press (India) Pvt. Ltd.

Reference Books

1. Sweigart, A. (2019). Automate the boring stuff with Python (2nd ed.). No Starch Press.

2. McKinney, W. (2022). Python for data analysis: Data wrangling with pandas, NumPy and Jupyter (3rd ed.). O'Reilly Media.

3. Zelle, J. M. (2016). Python programming: An introduction to computer science (3rd ed.). Franklin, Beedle & Associates.

4. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

Web Resources

1. Python Software Foundation — The Python Tutorial (official documentation). https://docs.python.org/3/tutorial/
2. NPTEL — Programming, Data Structures and Algorithms using Python. https://nptel.ac.in/
3. W3Schools — Python tutorial. https://www.w3schools.com/python/
4. Python Tutor — step-by-step visualisation of program execution. https://pythontutor.com/
5. NumPy, Pandas and Matplotlib official user guides. https://numpy.org/doc/stable/user/
6. GeeksforGeeks — Python programming language. https://www.geeksforgeeks.org/python-programming-language/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

Computational Thinking and Problem Solving with Python Lab

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Computational Thinking and Problem Solving with Python Lab	ESC	0 - 0 - 3	1.5

Pre-requisite: Co-requisite: Computational Thinking and Problem Solving with Python (theory), studied concurrently.

Course Outline

The laboratory follows the theory course week by week. For every exercise the student must write the algorithm, draw the flowchart, and then implement, execute and present the Python program with its output — so that the design step is never skipped in favour of typing code. Each three-hour session therefore carries two or three problems. The final session is an integrative capstone mini-project that exercises all four computational-thinking pillars on a realistic, imperfect dataset.

Course Objectives

1. To provide practice in translating a problem into an algorithm and a flowchart before implementation.
2. To develop fluency in Python control structures, data structures and functions through graded programming exercises.
3. To train students in systematic debugging, testing and exception handling.
4. To provide experience of an end-to-end data workflow through a capstone mini-project.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Construct algorithms and flowcharts for a given problem and translate them into working Python programs.	K3
CO2	Implement solutions using conditionals, loops and nested control structures for pattern-based problems.	K3
CO3	Select and manipulate appropriate Python data structures to organise and process real data.	K4
CO4	Debug, test and handle exceptions in Python programs, and verify correctness through systematic tracing.	K4
CO5	Develop a modular, documented Python application that reads, cleans, analyses and visualises a real dataset.	K6

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	2	2	1	3	-	-	1	2	1	3	1	1
CO2	3	3	3	2	3	-	-	1	2	1	3	2	1
CO3	3	3	3	3	3	-	-	1	2	1	3	2	2
CO4	3	3	3	3	3	-	-	2	2	2	3	3	2
CO5	2	2	2	2	3	1	1	3	3	3	2	2	2

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

Laboratory Schedule — 12 Sessions of 3 Hours Each

Common instruction for every session: the student must write the algorithm, draw the flowchart, and then implement, run and present the Python program with its output. Programs must carry a one-line docstring and comments on any non-obvious step.

S.No.	Experiment / Activity	CO Mapped
1	Find the largest of three numbers; check whether a given year is a leap year.	CO1
2	Compute simple and compound interest; compute the area and perimeter of a rectangle from user input; swap two numbers without a third variable.	CO1
3	Determine whether a number is positive, negative or zero; determine the BMI category using if-elif; classify a triangle as equilateral, isosceles or scalene.	CO2
4	Print all Armstrong numbers between 1 and 1000; simulate an ATM PIN-entry system allowing a maximum of three attempts; find the first n prime numbers using nested loops with break.	CO2
5	Encrypt and decrypt a message using a Caesar cipher; check whether two given strings are anagrams; count vowels, consonants, digits and spaces in a string.	CO3
6	Find the second-largest element in a list of scores; merge two sorted lists of roll numbers; generate squares of all even numbers from 1 to 50 using a list comprehension.	CO3
7	Store coordinates as tuples in a list and compute distances between consecutive points; maintain an inventory using a dictionary that updates stock after each transaction; count word frequency in a paragraph.	CO3
8	Using a function, check whether a number is perfect and list all perfect numbers up to 10,000; using *args, compute the total resistance of resistors in series or parallel.	CO1, CO4
9	Compute the GCD of two numbers recursively using the Euclidean algorithm; solve the Tower of Hanoi for n disks and count the moves; compute projectile range and maximum height using the math module.	CO4
10	Count the words, lines and characters in a text file; read a log file of sensor readings and append a new timestamped set of readings; read a CSV of marks and write the names of students scoring above 75% to a new file.	CO3, CO4
11	Build a calculator that gracefully handles ZeroDivisionError, ValueError and invalid operators; validate input for a unit-conversion tool and raise a custom exception for out-of-range values; using Pandas, read a CSV of daily electricity consumption and plot a line graph over a month.	CO4
12	Capstone mini-project — "Dataset Health-Check and Insights Tool": decompose the workflow into read → validate → clean → analyse → visualise → report stages, each as a separate function, using exception handling and NumPy, Pandas and Matplotlib; submit with a one-page reflection mapping each stage to the computational-thinking pillar it exercises.	CO5

Software Requirements: Python 3.10 or later with NumPy, Pandas and Matplotlib; an IDE such as VS Code, PyCharm Community Edition, IDLE or Jupyter Notebook; Git for the version-control primer; Python Tutor for execution visualisation. All are free and cross-platform. One computer per student is required.

Learning Resources

Text Books

1. Downey, A. B. (2015). Think Python: How to think like a computer scientist (2nd ed.). O'Reilly Media.

G. Jayasuma

2. Reema Thareja. (2019). Python programming using problem solving approach. Oxford University Press.

3. Guttag, J. V. (2021). Introduction to computation and programming using Python (3rd ed.). MIT Press.

Reference Books

1. Sweigart, A. (2019). Automate the boring stuff with Python (2nd ed.). No Starch Press.

2. McKinney, W. (2022). Python for data analysis (3rd ed.). O'Reilly Media.

3. Zelle, J. M. (2016). Python programming: An introduction to computer science (3rd ed.). Franklin, Beedle & Associates.

4. Matthes, E. (2023). Python crash course (3rd ed.). No Starch Press.

Web Resources

1. Python Software Foundation — The Python Tutorial. https://docs.python.org/3/tutorial/
2. Python Tutor — visualise code execution step by step. https://pythontutor.com/
3. NPTEL — Programming, Data Structures and Algorithms using Python. https://nptel.ac.in/
4. Pandas — official user guide and 10-minute introduction. https://pandas.pydata.org/docs/user_guide/
5. Matplotlib — official tutorials and gallery. https://matplotlib.org/stable/tutorials/
6. HackerRank — Python practice track for graded problem solving.
https://www.hackerrank.com/domains/python

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

I B.Tech. II Semester — Makers Lab

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Makers Lab	ESC	0 - 1 - 2	2

Pre-requisite: None. Engineering Physics and Digital Logic Design are useful but not assumed.

Course Outline

The Makers Lab equips students with hands-on skills across mechanical fabrication, electrical systems, electronics and embedded programming through project-based learning. Rather than a sequence of unrelated experiments, the course is organised so that each module contributes a subsystem to one functional prototype that the team designs, builds and tests by the end of the semester. Students learn to read a drawing, measure precisely, fabricate a part, wire a circuit, program a microcontroller and integrate the pieces into a working system — and, just as importantly, to diagnose why the assembled system does not work the first time.

Course Objectives

1. To develop practical skills in engineering drawing, precision measurement and the fabrication of components using conventional and digital methods.
2. To provide hands-on experience of electrical prototyping, power supplies, motor control and instrumentation.
3. To build competence in electronic component identification, sensor interfacing, microcontroller programming and PCB design.
4. To enable students to integrate multidisciplinary subsystems into a tested functional prototype through iterative design and troubleshooting.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Interpret and prepare engineering drawings, perform precision measurements, and identify electronic components using datasheets.	K3
CO2	Design, fabricate and assemble mechanical components and standard mechanisms using conventional workshop tools, 3D printing and laser cutting.	K3
CO3	Build, program and test electrical and embedded subsystems including power supplies, motor drives, sensors and microcontroller firmware.	K3
CO4	Analyse, troubleshoot and optimise an integrated engineering system through systematic testing and iterative prototyping.	K4

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	2	2	1	2	1	1	2	2	2	2	1	1
CO2	3	3	3	2	3	2	2	2	3	2	3	2	2
CO3	3	3	3	3	3	1	1	2	2	2	3	2	2
CO4	3	3	3	2	3	2	2	3	3	3	3	2	3

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

UNIT	CONTENTS	Hours (T + P)
UNIT – I	**Engineering Drawing, CAD and Precision Measurement:** Reading and preparing engineering drawings — orthographic projection, sectional views, dimensioning, tolerancing and scaling; introduction to parametric CAD modelling and assembly; export for manufacture. Precision measurement using steel rule, vernier caliper, micrometer, height gauge and depth micrometer; least count, zero error and measurement uncertainty; go/no-go gauges. Workshop safety, personal protective equipment and standard operating procedures.	3 + 8
UNIT – II	**Digital Fabrication — 3D Printing and Laser Cutting:** Additive manufacturing processes with emphasis on fused deposition modelling; slicing parameters — layer height, infill, supports and orientation; material selection (PLA, PETG, ABS); common print defects and their causes. Laser cutting and engraving — vector preparation, kerf compensation, material and power settings, safety. Design for manufacturing and design for assembly principles; fabrication of enclosures, brackets and a robot chassis; conventional workshop operations — marking, cutting, drilling, filing and fastening.	3 + 9
UNIT – III	**Electrical Systems — Prototyping, Power and Motor Control:** Generation and analysis of test signals using a function generator and digital storage oscilloscope; circuit prototyping on breadboard with switches and connectors. Transformer-based rectifier power supplies, filtering and linear voltage regulators; transistor switching circuits and relay drive. Astable and monostable multivibrators using the 555 timer. DC, BLDC, servo and stepper motors — characteristics, driver selection and interfacing with Arduino Nano; speed and direction control by PWM; introduction to PID-based speed control and evaluation of system response; electrical measurement, fault isolation and troubleshooting.	3 + 9
UNIT – IV	**Electronics and Embedded Systems:** Measurement of voltage, current, resistance, continuity and diode characteristics using a digital multimeter and bench supply; identification and testing of components using datasheets. Construction and testing of circuits using diodes, LEDs, Zener diodes, BJTs, rectifiers, voltage dividers and operational amplifiers. Microcontroller fundamentals — digital I/O, analog input, PWM and timers; ESP32 and Arduino programming environments; sensor interfacing over digital I/O, I²C and SPI; data acquisition and serial communication; introduction to wireless connectivity. PCB schematic capture, layout and routing using EDA software; soldering, assembly and board bring-up.	3 + 10
UNIT – V	**System Integration and Functional Prototype Project:** Requirement definition and subsystem partitioning; mechanical, electrical and electronic integration — wheel coupling, shaft alignment, wiring harness and enclosure fit; power budgeting and grounding practice. Systematic testing — unit test, integration test and system test; fault diagnosis, reliability assessment and validation; iterative refinement based on test results. Documentation — bill of materials, wiring diagram, firmware listing and test report; demonstration and peer review of the completed functional prototype.	2 + 6
	Total	14 + 42

Prototype project: Teams of three to four students select one functional prototype at the start of the semester — for example a line-following or obstacle-avoiding robot, an automated dispensing mechanism, a weather-monitoring station or a smart-home control node — and build one subsystem in each module so that integration in Unit V is the assembly of work already done rather than a fresh start.

Assessment: Practical Sessional 50%, Practical Mid-term 20% and Practical End-term 30%, in line with the assessment pattern adopted for project-based makers courses; the end-term assessment shall be a demonstration and viva on the functional prototype.

G. Jayakumar

Learning Resources

Text Books

1. Hajra Choudhury, S. K., Hajra Choudhury, A. K., & Roy, N. (2010). Elements of workshop technology, Volumes I and II. Media Promoters and Publishers.
2. Boylestad, R. L., & Nashelsky, L. (2015). Electronic devices and circuit theory (11th ed.). Pearson Education.
3. Bahga, A., & Madisetti, V. K. (2014). Internet of things: A hands-on approach. Universities Press.

Reference Books

1. Millman, J., & Halkias, C. C. (2010). Integrated electronics: Analog and digital circuits and systems (2nd ed.). McGraw Hill Education India.
2. Awari, G. K., Thorat, C. S., Ambade, V., & Kothari, D. P. (2021). Additive manufacturing and 3D printing technology: Principles and applications. CRC Press.
3. Nise, N. S. (2019). Control systems engineering (8th ed.). Wiley.
4. Monk, S. (2022). Programming Arduino: Getting started with sketches (3rd ed.). McGraw Hill.

Web Resources

1. Arduino — official documentation, language reference and tutorials. https://docs.arduino.cc/
2. Espressif — ESP32 programming guide and datasheets. https://docs.espressif.com/
3. KiCad — free and open-source PCB design suite with documentation. https://www.kicad.org/
4. Ultimaker Cura — slicing software and 3D printing settings guidance. https://ultimaker.com/software/ultimaker-cura/
5. Autodesk Tinkercad — browser-based 3D design and circuit simulation. https://www.tinkercad.com/
6. NPTEL — courses in Manufacturing Processes, Embedded Systems and IoT. https://nptel.ac.in/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

I B.Tech. II Semester — C and Data Structures Lab

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	C and Data Structures Lab	ESC	0 - 0 - 2	1

Pre-requisite: Co-requisite: Data Structures through C (theory), studied concurrently.

Course Outline

The laboratory turns the theory course into working code. Every data structure is implemented from first principles in C — no library containers — so that students confront pointer manipulation, dynamic allocation and memory release directly. Each exercise requires the student to state the algorithm and its expected complexity before coding, and to verify the implementation against boundary cases (empty structure, single element, full capacity, duplicate keys). The closing experiments introduce empirical timing so that students see asymptotic analysis confirmed by measurement.

Course Objectives

1. To provide practice in implementing abstract data types in C using pointers and dynamic memory allocation.
2. To develop the ability to implement and test linear and non-linear data structures.
3. To build competence in implementing searching, sorting and hashing techniques and comparing their performance.
4. To train students in systematic testing, boundary-case analysis and memory-leak avoidance.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Implement C programs using pointers, structures and dynamic memory allocation without memory leaks.	K3
CO2	Implement linear data structures — arrays, linked lists, stacks and queues — and their standard operations.	K3
CO3	Implement non-linear data structures — binary search trees, AVL trees, heaps and graphs — and their traversal algorithms.	K3
CO4	Implement searching, sorting and hashing techniques and compare their measured performance against the theoretical complexity.	K4
CO5	Test implementations systematically against boundary cases and document the work with algorithm, complexity and results.	K4

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	3	3	2	3	-	-	1	2	1	3	2	1
CO2	3	3	3	2	3	-	-	1	2	1	3	2	1
CO3	3	3	3	3	3	-	-	1	2	1	3	3	2
CO4	3	3	3	3	3	-	-	1	2	2	3	3	2
CO5	2	3	3	3	3	-	-	2	3	2	3	3	2

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

List of Experiments — Minimum 12 to be Completed

S.No.	Experiment / Activity	CO Mapped
1	C refresher — programs using pointers and pointer arithmetic, structures and unions, and functions demonstrating call by value versus call by reference.	CO1
2	Dynamic memory allocation using malloc, calloc, realloc and free; dynamic one- and two-dimensional arrays; sparse matrix representation using the triplet form and its transpose.	CO1
3	Array-based operations — insertion, deletion and merging; implementation of linear and binary search with a comparison of the number of key comparisons.	CO2, CO4
4	Singly linked list — creation, insertion at any position, deletion, traversal, searching, counting nodes and reversal.	CO2
5	Doubly linked list and circular linked list — all standard operations; representation and addition of two polynomials using linked lists.	CO2
6	Stack implementation using both arrays and linked lists; application — checking for balanced parentheses in an expression.	CO2
7	Conversion of an infix expression to postfix and prefix form, and evaluation of a postfix expression, using a stack.	CO2
8	Linear queue, circular queue and double-ended queue using arrays; priority queue implementation and a comparison of the representations.	CO2
9	Binary search tree — insertion, deletion and searching; inorder, preorder and postorder traversals implemented both recursively and iteratively; level-order traversal.	CO3
10	AVL tree — insertion with single and double rotations; construction of a max heap, heap insertion and deletion, and heap sort.	CO3
11	Graph representation using an adjacency matrix and an adjacency list; breadth-first and depth-first traversal; determination of connectivity and path existence.	CO3
12	Hashing with linear probing and separate chaining, including rehashing; implementation of quick sort, merge sort and radix sort, with measured execution time on the same dataset compared against the theoretical complexity.	CO4, CO5

Software Requirements: GCC or Clang compiler on Linux or Windows; a code editor or IDE such as VS Code, Code::Blocks or Dev-C++; Valgrind or a comparable tool for detecting memory leaks in the later experiments; VisuAlgo or the USFCA algorithm visualiser for pre-lab preparation. One computer per student.

Learning Resources

Text Books

1. Horowitz, E., Sahni, S., & Anderson-Freed, S. (2008). Fundamentals of data structures in C (2nd ed.). Universities Press.
2. Thareja, R. (2014). Data structures using C (2nd ed.). Oxford University Press.
3. Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language (2nd ed.). Prentice Hall.

Reference Books

1. Weiss, M. A. (2012). Data structures and algorithm analysis in C (2nd ed.). Pearson Education.
2. Tenenbaum, A. M., Langsam, Y., & Augenstein, M. J. (2019). Data structures using C. Pearson Education.
3. Kruse, R. L., & Ryba, A. J. (2007). Data structures and program design in C (2nd ed.). Pearson Education.
4. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms (4th ed.). MIT Press.

Web Resources

1. VisuAlgo — visualisation of data structures and algorithms. https://visualgo.net/en
2. Data Structure Visualizations, University of San Francisco. https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
3. GeeksforGeeks — data structures tutorial with C implementations. https://www.geeksforgeeks.org/data-structures/

4. NPTEL — Programming and Data Structures laboratory courses. https://nptel.ac.in/
5. Virtual Labs — Data Structures laboratory (NMEICT, Ministry of Education). https://www.vlab.co.in/
6. Compiler Explorer — inspect generated code and behaviour online. https://godbolt.org/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

I B.Tech. II Semester — Fundamentals of Quantum Computing

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Fundamentals of Quantum Computing	BSC	3 - 0 - 0	3

Pre-requisite: Mathematics-I (matrices, eigenvalues and complex numbers); Engineering Physics (Unit on quantum mechanics).

Course Outline

This course gives first-year students a rigorous but accessible entry into quantum computation. It builds the necessary linear algebra in Dirac notation, introduces the qubit and the quantum gates that act on it, and then develops the phenomena — superposition, entanglement and interference — that give quantum algorithms their advantage. Students study the canonical algorithms from Deutsch–Jozsa through Grover and Shor, examine why noise makes present-day devices hard to use, and gain hands-on experience by building and running circuits on a simulator and, where access permits, on real quantum hardware.

Course Objectives

1. To develop the mathematical foundation — complex vector spaces, Dirac notation and linear operators — required for quantum computation.
2. To introduce the qubit, quantum gates and quantum circuits, and the constraints imposed by unitarity and the no-cloning theorem.
3. To develop an understanding of entanglement and its use in communication and cryptographic protocols.
4. To impart a working knowledge of the principal quantum algorithms and their complexity advantages.
5. To familiarise students with noise, error correction and current hardware platforms, and to provide hands-on practice with a quantum programming framework.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Apply the linear algebra of complex vector spaces in Dirac notation to represent quantum states, operators and measurements.	K3
CO2	Analyse the action of single- and multi-qubit gates and construct quantum circuits for a given transformation.	K4
CO3	Explain entanglement and apply it to protocols such as superdense coding, teleportation and quantum key distribution.	K3
CO4	Analyse the principal quantum algorithms and compare their complexity with the best classical alternatives.	K4
CO5	Explain decoherence, error correction and hardware platforms, and implement and simulate quantum circuits using a programming framework such as Qiskit.	K3

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	3	2	1	2	-	-	-	-	-	3	2	1
CO2	3	3	2	2	3	-	-	-	-	-	3	2	2
CO3	3	3	2	2	3	1	-	1	-	-	3	2	2
CO4	3	3	3	3	3	1	-	1	-	1	3	3	3
CO5	3	3	3	2	3	2	1	1	1	1	3	3	3

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

Syllabus

UNIT	CONTENTS	Contact Hours
UNIT – I	**Mathematical and Physical Foundations:** Review of complex numbers; complex vector spaces and inner product spaces; Dirac bra–ket notation; orthonormal bases; Hilbert space. Linear operators and their matrix representations; Hermitian, unitary and projection operators; eigenvalues, eigenvectors and spectral decomposition; the Pauli matrices; tensor products of states and operators. Postulates of quantum mechanics — state space, evolution, measurement and composite systems; superposition and the effect of measurement; classical bits versus quantum bits; a comparison of classical and quantum models of computation.	10
UNIT – II	**Qubits, Quantum Gates and Circuits:** The qubit and its geometric representation on the Bloch sphere; global and relative phase. Single-qubit gates — Pauli X, Y and Z, Hadamard, phase (S) and $\pi/8$ (T) gates, and general rotation gates; their matrix forms and Bloch-sphere action. Multi-qubit systems and product states; controlled gates — CNOT, controlled-Z, SWAP and Toffoli; universality of gate sets. Quantum circuit notation, circuit identities and circuit depth; measurement in the computational and other bases; the no-cloning theorem and its consequences; reversible computation and the relation between classical and quantum circuits.	10
UNIT – III	**Entanglement and Quantum Protocols:** Separable and entangled states; the Bell states and their preparation; the EPR paradox; Bell inequality and its experimental violation (conceptual treatment); measures of entanglement (introduction). Superdense coding; quantum teleportation — protocol, circuit and the role of classical communication. Quantum cryptography — the BB84 protocol and the E91 entanglement-based protocol; detection of eavesdropping; quantum random number generation; implications of quantum computing for classical public-key cryptography and the motivation for post-quantum cryptography.	9
UNIT – IV	**Quantum Algorithms:** Quantum parallelism and the role of interference; the oracle model and query complexity. Deutsch's algorithm and the Deutsch–Jozsa algorithm; the Bernstein–Vazirani algorithm; Simon's algorithm. Grover's search algorithm — the diffusion operator, geometric interpretation, number of iterations and the quadratic speed-up; amplitude amplification. The quantum Fourier transform and its circuit; quantum phase estimation; order finding and Shor's factoring algorithm (structure and outline) with its implications for RSA; a comparative summary of the complexity advantages obtained.	10
UNIT – V	**Noise, Hardware and Quantum Programming:** Decoherence and noise; density matrices and mixed states (introduction); common noise channels — bit flip, phase flip and depolarising. Quantum error correction — the three-qubit bit-flip and phase-flip codes, the nine-qubit Shor code, and the threshold theorem (qualitative); fault tolerance. NISQ-era devices and their limitations; hardware platforms — superconducting qubits, trapped ions, photonic and neutral-atom systems, and their comparison; variational algorithms — VQE and QAOA (overview); introduction to quantum machine learning. Programming with Qiskit — building, transpiling, simulating and executing circuits; reading histograms and interpreting measurement statistics.	9
	Total	**48**

Suggested hands-on component: Although this is a theory course, students should be given at least six guided sessions on a simulator. Recommended free platforms are IBM Quantum Learning with Qiskit, the Microsoft Quantum Katas, and PennyLane; all run in a browser or on a standard laboratory computer and require no quantum hardware.

Learning Resources

Text Books

1. Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information (10th anniversary ed.). Cambridge University Press.
2. Hidary, J. D. (2021). Quantum computing: An applied approach (2nd ed.). Springer.
3. Kaye, P., Laflamme, R., & Mosca, M. (2007). An introduction to quantum computing. Oxford University Press.

G. Jayalone

Reference Books

1. Mermin, N. D. (2007). Quantum computer science: An introduction. Cambridge University Press.
2. Yanofsky, N. S., & Mannucci, M. A. (2008). Quantum computing for computer scientists. Cambridge University Press.
3. Rieffel, E. G., & Polak, W. H. (2011). Quantum computing: A gentle introduction. MIT Press.
4. Bernhardt, C. (2019). Quantum computing for everyone. MIT Press.

Web Resources

1. IBM Quantum Learning — courses, tutorials and the Qiskit textbook material. https://learning.quantum.ibm.com/
2. Qiskit — open-source SDK for quantum computing. https://www.ibm.com/quantum/qiskit
3. PennyLane — quantum machine learning demonstrations and tutorials. https://pennylane.ai/qml/
4. Microsoft Quantum Katas — self-paced programming exercises. https://github.com/microsoft/QuantumKatas
5. Quantum Country — Quantum Computing for the Very Curious. https://quantum.country/
6. NPTEL — courses in Quantum Computing and Quantum Information. https://nptel.ac.in/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

DIGITAL LOGIC DESIGN

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	DIGITAL LOGIC DESIGN	PCC	3 - 0 - 0	3

Course Objectives

1. To introduce the fundamentals of digital systems and number representations.
2. To develop the ability to analyze and simplify Boolean functions using algebraic and graphical methods.
3. To design and implement combinational logic circuits using logic gates and MSI components.
4. To understand the operation and design of sequential logic circuits using flip-flops, registers, and counters.
5. To provide a foundation for advanced courses in computer organization, embedded systems, and VLSI design.

Course Outcomes (COs)

After successful completion of the course, the student will be able to:

1. Apply number systems, codes, and Boolean algebra in digital circuit analysis.
2. Simplify Boolean expressions using Boolean theorems and Karnaugh maps.
3. Design and analyze combinational logic circuits using standard building blocks.
4. Design and implement sequential circuits using latches, flip-flops, registers, and counters.
5. Analyze and design digital systems for practical engineering applications.

UNIT-I: Number Systems and Digital Fundamentals

Digital Systems and Binary Information, Number Systems: Binary, Octal, Decimal, Hexadecimal ,Number Base Conversions ,Signed Numbers and Complements ,Binary Arithmetic ,Binary Codes: BCD, Gray Code, Excess-3 Code ,Error Detection Codes ,Introduction to Logic Gates

UNIT-II: Boolean Algebra and Logic Simplification:

Boolean Algebra and Postulates ,Basic Theorems and Properties ,Boolean Functions, Canonical Forms (SOP and POS) ,Logic Gates and Universal Gates ,Minimization using Boolean Algebra ,Karnaugh Maps (up to 4 Variables) ,Don't Care Conditions ,NAND and NOR Realizations

UNIT-III: Combinational Logic Circuits: Analysis and Design Procedure, Half Adder and Full Adder ,Half Subtractor and Full Subtractor ,Parallel Adders and BCD Adder, Magnitude Comparator ,Decoders and Encoders ,Priority Encoders ,Multiplexers and De-multiplexers ,Code Converters

UNIT-IV: Sequential Logic Circuits: Introduction to Sequential Logic ,Latches and Flip-Flops ,SR, JK, D, and T Flip-Flops ,Characteristic and Excitation Tables ,Flip-Flop Conversions ,Analysis of Clocked Sequential Circuits ,State Tables and State Diagrams ,Introduction to Finite State Machines (FSMs)

UNIT-V: Registers, Counters and Memory Elements: Registers and Register Transfer Operations, Shift Registers (SISO, SIPO, PISO, PIPO) ,Universal Shift Registers ,Asynchronous Counters ,Synchronous Counters ,Ring Counter and Johnson Counter ,Introduction to Semiconductor Memories ,ROM, RAM, PLA and PAL.

Text Books

1. M. Morris Mano and Michael D. Ciletti, Digital Design, 6th Edition, Pearson Education, 2018.
2. Charles H. Roth Jr. and Larry L. Kinney, Fundamentals of Logic Design, 7th Edition, Cengage Learning.
3. Ronald J. Tocci, Neal Widmer and Gregory Moss, Digital Systems: Principles and Applications, 12th Edition, Pearson Education.

Reference Books

1. M. Morris Mano and Charles R. Kime, Logic and Computer Design Fundamentals, 5th Edition, Pearson Education.
2. Thomas L. Floyd, Digital Fundamentals, 11th Edition, Pearson Education.
3. R. P. Jain, Modern Digital Electronics, 5th Edition, McGraw-Hill Education.

4. Donald P. Leach, Albert Paul Malvino and Goutam Saha, Digital Principles and Applications, Tata McGraw-Hill.
5. John F. Wakerly, Digital Design: Principles and Practices, 5th Edition, Pearson Education

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

I B.Tech. II Semester — Data Structures through C

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Data Structures through C	ESC	3 - 0 - 0	3

Pre-requisite: Computational Thinking and Problem Solving with Python; familiarity with basic programming constructs.

Course Outline

This course bridges the student's first programming experience into the systematic study of data organisation. It opens with a focused C refresher on pointers, structures and dynamic memory — the mechanisms every data structure in C is built from — together with asymptotic analysis as the language for comparing solutions. It then develops linear structures (arrays, linked lists, stacks and queues), non-linear structures (trees and graphs), and the searching, sorting and hashing techniques that make retrieval efficient. Throughout, each structure is introduced with the class of problem it solves rather than as an isolated implementation exercise.

Course Objectives

1. To bridge prior programming knowledge into C constructs — pointers, structures, functions and dynamic memory allocation — and to introduce algorithm-analysis techniques.
2. To develop the ability to implement linear data structures using the C programming language.
3. To enable the design and implementation of non-linear data structures such as trees and graphs.
4. To develop proficiency in searching, sorting and hashing techniques for efficient data organisation and retrieval.
5. To build problem-solving skills through the implementation of data structures for real-world applications.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Apply core C constructs and abstract data types, and analyse algorithm efficiency using asymptotic notation.	K3
CO2	Implement and manipulate linked-list based data structures using dynamic memory allocation in C.	K3
CO3	Analyse and implement stack- and queue-based solutions for problems such as expression evaluation and scheduling.	K4
CO4	Evaluate and implement tree-based structures — binary search trees, AVL trees and heaps — for given data-organisation requirements.	K5
CO5	Design and implement graph algorithms, hashing techniques and sorting solutions for a given application.	K6

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	3	3	2	2	2	-	-	-	1	1	3	2	1
CO2	3	3	3	2	2	-	-	-	1	1	3	2	1
CO3	3	3	3	2	2	-	-	-	1	1	3	2	2
CO4	3	3	3	3	2	-	-	-	1	1	3	3	2
CO5	3	3	3	3	3	-	-	1	2	1	3	3	2

Syllabus

UNIT	CONTENTS	Contact Hours
UNIT – I	**C Programming Bridge and Introduction to Data Structures:** C programming refresher — pointers and pointer arithmetic, pointers to pointers, arrays of pointers; structures and unions; self-referential structures; functions, call by value versus call by reference; dynamic memory allocation using malloc, calloc, realloc and free, and common memory errors. Abstract data types; classification of data structures. Algorithm analysis — time and space complexity, frequency count, asymptotic notation (Big-O, Omega and Theta), best, average and worst case. Arrays — one-dimensional and multi-dimensional, row-major and column-major representation, address calculation, array-based operations; sparse matrix representation and transpose.	10
UNIT – II	**Linked Lists:** Limitations of arrays and the need for dynamic structures; singly linked lists — representation, creation, insertion, deletion, traversal, searching, reversal and concatenation; doubly linked lists and their operations; circular singly and doubly linked lists; header-node representation; comparison of array and linked representations. Applications — polynomial representation, addition and multiplication using linked lists; sparse matrix representation using linked lists; memory-efficient linked lists.	9
UNIT – III	**Stacks and Queues:** Stack — abstract data type, representation using arrays and linked lists, push, pop, peek and overflow/underflow conditions. Applications of stacks — conversion of infix to postfix and prefix, evaluation of postfix and prefix expressions, balanced-parentheses checking, and the role of the stack in recursion and function calls. Queue — representation using arrays and linked lists; linear queue and its drawback; circular queue; double-ended queue; priority queue and its representations. Applications — CPU and disk scheduling, buffering and breadth-first traversal.	10
UNIT – IV	**Trees:** Tree terminology; binary trees — properties, array and linked representations; binary tree traversals — inorder, preorder and postorder (recursive and iterative), and level-order traversal; construction of a binary tree from traversals; expression trees. Binary search trees — searching, insertion and deletion, and their complexity. Threaded binary trees. AVL trees — balance factor, single and double rotations, insertion and deletion. Introduction to B-trees and their use in indexing. Heaps — max and min heaps, construction, insertion, deletion and heap sort; priority queue using a heap.	10
UNIT – V	**Graphs, Hashing and Advanced Sorting and Searching:** Graph terminology and representations — adjacency matrix, adjacency list and adjacency multi-list; graph traversals — breadth-first and depth-first search and their applications; spanning trees, minimum spanning trees by Prim's and Kruskal's algorithms; single-source shortest path by Dijkstra's algorithm. Hashing — hash functions, collision and collision-resolution techniques (linear probing, quadratic probing, double hashing and separate chaining), load factor and rehashing. Sorting — bubble, selection, insertion, quick, merge, radix and shell sorts with complexity comparison; searching — linear, binary and interpolation search.	9
	Total	**48**

Learning Resources

Text Books

1. Horowitz, E., Sahni, S., & Anderson-Freed, S. (2008). Fundamentals of data structures in C (2nd ed.). Universities Press.
2. Thareja, R. (2014). Data structures using C (2nd ed.). Oxford University Press.
3. Weiss, M. A. (2012). Data structures and algorithm analysis in C (2nd ed.). Pearson Education.

Reference Books

1. Kruse, R. L., & Ryba, A. J. (2007). Data structures and program design in C (2nd ed.). Pearson Education.

G. Jayakumar

2. Tenenbaum, A. M., Langsam, Y., & Augenstein, M. J. (2019). Data structures using C. Pearson Education.
3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms (4th ed.). MIT Press.
4. Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language (2nd ed.). Prentice Hall.

Web Resources

1. NPTEL — Programming, Data Structures and Algorithms. https://nptel.ac.in/
2. GeeksforGeeks — data structures tutorial. https://www.geeksforgeeks.org/data-structures/
3. VisuAlgo — visualisation of data structures and algorithms. https://visualgo.net/en
4. Virtual Labs — Data Structures laboratory (NMEICT, Ministry of Education). https://www.vlab.co.in/
5. Data Structure Visualizations, University of San Francisco.
https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
6. MIT OpenCourseWare — Introduction to Algorithms. https://ocw.mit.edu/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

AI Tools and Applications

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	AI Tools and Applications	ESC	0-1-2	2

Pre-requisite: Intermediate (10+2) passed. Basic computer literacy.

Course Outline

This course introduces first-year engineering students to the practical use of artificial intelligence tools across problem solving, data analysis, content creation and coding. Students gain hands-on experience with generative AI, prompt engineering, AI-assisted data visualisation and productivity and coding copilots through weekly laboratory practice. The course gives equal weight to responsible and ethical use — verification, citation, privacy and copyright — alongside technical competence, and serves as the foundation for AI-assisted work in higher-semester courses.

Course Objectives

1. To familiarise students with the landscape of AI tools and their underlying working principles.
2. To develop competence in prompt engineering for effective interaction with generative AI systems.
3. To enable students to apply AI tools for data analysis, visualisation and interpretation.
4. To train students in using AI-assisted tools for coding, writing and productivity tasks.
5. To instil awareness of the ethical, legal and societal considerations in AI tool usage.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Identify the categories, capabilities and limitations of major AI tools and select an appropriate tool for a given task.	K2
CO2	Explain the principles of prompt engineering and construct zero-shot, few-shot and chain-of-thought prompts that produce reliable output.	K3
CO3	Apply AI tools to first-year engineering problems in analysis, writing and coding.	K3
CO4	Analyse AI-generated output for accuracy, bias, hallucination and reliability before use.	K4
CO5	Evaluate the ethical, academic-integrity, privacy and copyright implications of AI tool use and justify appropriate usage choices.	K5

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	2	1	1	1	3	1	-	1	1	1	2	2	1
CO2	2	2	2	1	3	-	-	1	1	2	2	2	2
CO3	2	3	2	2	3	1	-	1	2	2	3	3	2
CO4	2	3	1	3	3	1	-	2	1	2	2	3	2
CO5	1	2	1	2	2	3	2	3	2	2	1	2	3

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

Syllabus

UNIT	CONTENTS	Hours (T + P)
UNIT – I	**Introduction to AI and the AI Tools Landscape:** What artificial intelligence, machine learning and generative AI are, and how they relate to one another; how large language models process and generate responses (conceptual, non-mathematical); core categories of AI tools — conversational and general-purpose assistants, coding assistants, data-analysis tools, image generation and writing and productivity tools; criteria for selecting the right tool for a task — accuracy needs, cost, data sensitivity and output format; setting up accounts and workspaces; free-tier versus paid-tier capabilities and limitations. Practicals: compare the same engineering query across two AI platforms; prepare a comparison chart of five tools by category and use case.	3 + 8
UNIT – II	**Prompt Engineering and Generative AI Tools:** Anatomy of a good prompt — context, clear instruction, desired format and constraints; zero-shot, few-shot and chain-of-thought prompting and when each is appropriate; role-based and context-setting prompts; iterative prompt refinement — diagnosing a poor response and rewriting the prompt; system prompts and conversation context; using generative AI for brainstorming, summarising and drafting technical content. Practicals: write basic, detailed and role-based versions of the same prompt and compare outputs; summarise a technical article and verify against the original; construct a chain-of-thought prompt for a multi-step numerical problem and check the reasoning.	3 + 8
UNIT – III	**AI Tools for Data Analysis and Visualisation:** AI-assisted spreadsheet and dataset analysis using natural-language commands; natural-language querying to extract summary statistics and insights; generating charts, graphs and visual summaries from tabular data; interpreting AI-generated statistical summaries correctly and recognising their limitations; identifying and correcting errors and misinterpretations in AI-generated analysis. Practicals: clean and summarise a small dataset using an AI tool; generate three chart types from the same data and justify which is most appropriate; cross-check one AI-generated statistical claim manually and report any discrepancy.	3 + 8
UNIT – IV	**AI Tools for Coding, Writing and Productivity:** AI coding assistants — code generation, explanation and debugging support for simple programs; the limits of generated code and the need to read before running; AI writing assistants for reports, e-mails and technical documentation; AI tools for scheduling, note-taking and meeting or lecture summarisation; AI-assisted presentation and slide-outline generation; integrating multiple AI tools into a coherent day-to-day academic workflow. Practicals: use an AI coding tool to write and debug a program for a given problem statement; draft a formal report with an AI writing assistant and then manually edit it for accuracy and tone; generate and refine a slide outline for a five-minute seminar.	3 + 8
UNIT – V	**Ethics, Reliability and Emerging Trends:** Bias and hallucination in AI outputs and techniques for detecting and fact-checking them; academic integrity and the appropriate citation of AI assistance in coursework; data privacy and security, including obligations under India's Digital Personal Data Protection Act, 2023 when uploading personal or institutional data; copyright and intellectual property in AI-generated text and images; emerging trends — multimodal AI and AI agents — and their implications for engineering practice. Practicals: document a hallucination or bias instance from a factual query; draft an AI-usage declaration for an assignment; review the privacy policy of one tool used in the course and identify a DPDP Act compliance consideration.	2 + 8
	Total	**14 + 40**

G. Jayasoma

Note on tool selection: Tool names change rapidly. The syllabus specifies categories of tools rather than mandating particular products; the department shall publish, before the start of each semester, the specific free-tier tools to be used in that offering, so that no student is required to purchase a subscription.

Learning Resources

Text Books

1. Mollick, E. (2024). Co-intelligence: Living and working with AI. Portfolio/Penguin.
2. Marr, B. (2024). Generative AI in practice: 100+ amazing ways generative artificial intelligence is changing business and society. Wiley.
3. Phoenix, J., & Taylor, M. (2024). Prompt engineering for generative AI. O'Reilly Media.

Reference Books

1. Russell, S., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed.). Pearson Education.
2. Selwyn, N. (2022). Education and technology: Key issues and debates (3rd ed.). Bloomsbury Academic.
3. Susskind, R., & Susskind, D. (2022). The future of the professions (Updated ed.). Oxford University Press.
4. Crawford, K. (2021). Atlas of AI: Power, politics, and the planetary costs of artificial intelligence. Yale University Press.

Web Resources

1. Google — Google AI Essentials, Grow with Google. https://grow.google/ai-essentials/
2. Anthropic — Prompt engineering overview, Claude documentation. https://docs.claude.com/en/docs/build-with-claude/prompt-engineering/overview
3. OpenAI — documentation and guides. https://platform.openai.com/docs
4. Ministry of Electronics and Information Technology — Digital Personal Data Protection Act, 2023. https://www.meity.gov.in/
5. UNESCO — Guidance for generative AI in education and research. https://www.unesco.org/
6. NPTEL — introductory courses on Artificial Intelligence. https://nptel.ac.in/

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY – GURAJADA, VIZIANAGARAM

B.Tech. R26 Regulations

I B.Tech. I Semester — Design Thinking and Innovation

COURSE CODE	COURSE NAME	CATEGORY	L - T - P	CREDITS
	Design Thinking and Innovation	ESC	0-1-1	1.5

Pre-requisite: Intermediate (10+2) passed. Teams of three to four students are formed at the start of Unit II.

Course Outline

This course introduces design thinking as applied specifically to software product design and UX/UI development. Through weekly tutorials and hands-on practicals, students learn to research digital-product users, map journeys and information architecture, wireframe and prototype interfaces, and validate designs through usability testing. The course uses industry-standard, free-to-use tools (Figma and Penpot) alongside AI-assisted ideation and UI-generation tools to mirror current practice. From Unit II onward students work in teams on a shared application problem, culminating in a tested interactive prototype and a UX case-study pitch that connects explicitly to development handoff.

Course Objectives

1. To introduce design thinking as applied to software products and UX/UI development for digital solutions.
2. To develop the ability to conduct user research and translate findings into personas and journey maps.
3. To build skills in ideation, information architecture, wireframing and prototyping using industry-standard and AI-assisted tools.
4. To familiarise students with usability heuristics, accessibility principles and structured usability testing.
5. To cultivate collaborative team-based workflows and the communication of design decisions through case studies and pitches.

Course Outcomes

CO	Course Outcome — At the end of the course, the student will be able to	Knowledge Level (K)
CO1	Explain the design thinking process as applied to UX/UI and software product development.	K2
CO2	Apply user research methods — interviews, journey mapping and competitive analysis — to identify digital product opportunities.	K3
CO3	Analyse user flows and information architecture to define scoped, testable feature requirements.	K4
CO4	Create wireframes and interactive prototypes using digital design tools, applying visual hierarchy and accessibility principles.	K6
CO5	Evaluate prototypes through usability testing and heuristic evaluation, and justify iterative design refinements.	K5

Mapping of Course Outcomes with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PSO1	PSO2	PSO3
CO1	2	2	2	1	1	2	1	1	2	2	2	1	2
CO2	1	3	2	2	2	2	1	2	2	3	2	1	2
CO3	2	3	3	2	2	1	1	1	2	2	3	2	2
CO4	1	2	3	2	3	2	2	2	2	2	3	2	3
CO5	1	3	3	3	3	2	2	3	3	3	3	2	3

Correlation levels: 3 = High, 2 = Medium, 1 = Low, – = Not applicable.

Syllabus

UNIT	CONTENTS	Hours (T + P)
UNIT – I	**Introduction to Design Thinking for Digital Products and User Empathy:** Design thinking overview for software and UX; human-centred design in digital products; the five-stage process — empathise, define, ideate, prototype and test; user research methods — interviews, surveys and contextual inquiry; empathy mapping for digital users; personas for software users. Practicals: conduct interviews on the college LMS or exam portal and build an empathy map; heuristic observation of an existing mobile application listing at least five usability frustrations; construct a persona with goals, frustrations and technology comfort level.	3 + 8
UNIT – II	**Define — Problem Framing for Digital Solutions:** Synthesising UX research; affinity mapping; user journey mapping with emotional highs and lows; point-of-view (POV) statements and How-Might-We (HMW) questions; scoping a minimum viable product; competitive and comparative application analysis. Teams of three to four are formed here and select one shared application problem to carry through the course. Practicals: build an affinity diagram from pooled research and identify at least three pain-point themes; create a journey map for one task; analyse three competing applications; write three POV statements and three HMW questions.	3 + 8
UNIT – III	**Ideation and Information Architecture:** Divergent and convergent ideation for software features; brainstorming rules; Crazy 8s sketching; user flows; site maps and information architecture; card sorting; the role of AI tools in ideation and content generation and their limitations. Practicals: run a Crazy 8s session for a chosen screen; create a user-flow diagram for the primary task mapping every screen and decision point; use an AI UI tool to generate ten alternate feature ideas or microcopy variations, compare them against the team's own for originality and feasibility, and justify the final selection in writing.	3 + 8
UNIT – IV	**Prototyping — Wireframes, Clickable Prototypes and AI-Assisted UI Generation:** Low-fidelity wireframes on paper and in digital tools; wireframing and prototyping using Figma or Penpot (both free); UI design principles — visual hierarchy, typography, colour and accessibility; building clickable prototypes; AI-assisted UI generation tools; design-system basics — components, tokens and reusable styles. The unit opens with a guided tool-bootcamp session before graded work begins. Practicals: sketch low-fidelity wireframes for at least four key screens; convert them into a mid-fidelity clickable prototype; generate and evaluate three alternate visual styles using an AI UI tool with written reasoning.	3 + 10
UNIT – V	**Usability Testing, Iteration, Ethics and Pitching:** Usability testing methods — moderated and unmoderated, task-based testing; the think-aloud protocol; Nielsen's ten usability heuristics; accessibility testing basics — colour contrast, tap-target size and screen-reader labels; dark patterns and inclusive design; ethics of persuasive design; iteration based on feedback; UX case-study	2 + 8

G. Jayasowmya

	presentation and development handoff. Practicals: conduct task-based usability testing with at least three users recording completion time and errors; perform a heuristic evaluation of the team prototype and revise it; deliver a five-minute UX case-study pitch with before-and-after screens and one slide on development handoff.	
	Total	**14 + 42**

Team Working Agreement: Unit I is individual work so that every student builds foundational research skills. From Unit II, all deliverables are submitted jointly by the team, but each student additionally submits a short individual contribution log for each unit noting their specific role and input; the instructor may reference this log when an individual student's grade needs adjustment.

Tools Required: Figma or Penpot (free) for wireframing and prototyping; a general-purpose AI assistant for research synthesis; an AI UI-generation tool such as Figma AI, Google Stitch or Uizard; WebAIM Contrast Checker for accessibility; a presentation tool for the final pitch. One computer per two students with internet access.

Learning Resources

Text Books

1. Norman, D. A. (2013). The design of everyday things (Rev. and expanded ed.). Basic Books.
2. Garrett, J. J. (2011). The elements of user experience: User-centered design for the web and beyond (2nd ed.). New Riders.
3. Brown, T. (2019). Change by design: How design thinking transforms organizations and inspires innovation (Rev. ed.). Harper Business.
4. Design Thinking: A Hands-on Approach, Chandramouli Subramanian, Thiyagarajan , Paramasivan, Sankaran Venkataramani, Universities Press

Reference Books

1. Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About face: The essentials of interaction design (4th ed.). Wiley.
2. Krug, S. (2014). Don't make me think, revisited: A common sense approach to web usability (3rd ed.). New Riders.
3. Kelley, T., & Kelley, D. (2013). Creative confidence: Unleashing the creative potential within us all. Crown Business.
4. Lewrick, M., Link, P., & Leifer, L. (2020). The design thinking toolbox. Wiley.

Web Resources

1. Nielsen Norman Group — 10 usability heuristics for user interface design. https://www.nngroup.com/articles/ten-usability-heuristics/
2. Interaction Design Foundation — What is design thinking? https://www.interaction-design.org/literature/topics/design-thinking
3. Figma — resources for design and prototyping. https://www.figma.com/resources/
4. Penpot — open-source design and prototyping platform. https://penpot.app/
5. WebAIM — colour contrast checker and accessibility guidance. https://webaim.org/resources/contrastchecker/
6. NPTEL — Design Thinking and Innovation courses. https://nptel.ac.in/