# Course Structure

&

# Detailed Syllabus

## M.Tech

## Computer Science and Engineering

## Academic Regulations-R25

### Applicable for the Batches Admitted from 2025-2026


**AVANTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

(Approved by A.I.C.T.E., New Delh i& Affiliated to J.N.T.U.H, Hyderabad)

NAAC "A" Accredited Institute

Gunthapally(V),Abdullapurmet(M),R.R(Dist),Near Ramojifilm City, Hyderabad,Pin -501512.

www.aietg.ac.in,principalavih@avanthi.edu.in

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# AVANTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

## Vision and Mission of the Institute

### VISION

To develop highly skilled professionals with ethics & human values.

### MISSION

1. To provide high-quality education along with professional training and exposure to the workplace.
2. To encourage a professional mindset that goes beyond academic achievement.
3. To promote holistic education among Department students by means of integrated pedagogy and scholarly mentoring for excellence in both personal and professional domains.
4. To consistently enhance the teaching and learning procedures in order to prepare students for successful careers in business or overseas or in further education.
5. To carefully prepare students to be globally employable professionals who will meet societal demands and contribute to the nation's technological advancement through their research and innovative talents.

### QUALITY POLICY

AIET focuses a strong emphasis on the moral principles of delivering cutting edge skilling by establishing the best infrastructure through interactive & activity-based learning. It also strives for an ambitious & effective governance that is responsive in every aspect, and makes use of the latest developments in knowledge and communication technology to encourage students to adopt a global perspective

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (Data Science)

## AVANTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

## Program: M.TECH- COMPUTER SCIENCE AND ENGINEERING
## Regulation: R25

## Vision and Mission of the Department

### DEPARTMENT VISION

To become a center of excellence the computer science and information technology discipline with a strong research and teaching environment.

### DEPARTMENT MISSION

1. To provide qualitative education and generate new knowledge by engaging in cutting edge research and by offering state of the art undergraduate, post graduate, leading careers as computer professional in the widely diversified of industry, government and academia.

2. To promote a teaching and learning process that yields advancements instate of art in computer science and engineering in integration of research result and innovative into other scientific discipline leading to new products.

3. To harness human capital for sustainable competitive edge and social relevance by including the philosophy of continuous learning and innovation in computer science and engineering.

4. To provide inculcate team work, imbibe leadership qualities, professional ethics and social responsibilities.

# AVANTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

**(Autonomous)**

(Approved by A.I.C.T.E., New Delhi & Affiliated to J.N.T.U.H, Hyderabad)

NAAC "A" Accredited Institute

Gunthapally (V), Abdullapurmet (M), R.R (Dist), Near Ramoji film City, Hyderabad, Pin -501512.

www.aietg.ac.in, principalavih@avanthi.edu.in


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Course Structure**

**Program– M. Tech Computer Science and Engineering**

(Applicable from the academic year 2025-2026 to 2028-2029)

Program: **M.Tech Computer Science and Engineering** **Regulation: R25**

**I Year I Semester-Course Structure**

| S.No | Category | Course Code | Course Title | Hours per Week | | | |
|---|---|---|---|---|---|---|---|
| | | | | Lecture | Tutorial | Practical | Credits |
| 1 | PC | R25D58101 | **Professional Core-1**<br>Mathematical Foundations of Computer Science | 3 | 0 | 0 | 3 |
| 2 | PC | R25D58102 | **Professional Core-2**<br>Advanced Data Structures | 3 | 0 | 0 | 3 |
| 3 | PE | R25D58103<br>R25D58104<br>R25D58105<br>R25D58106 | **Professional Elective-1**<br>1. Database Programming with PL/SQL<br>2. Deep Learning<br>3. Natural Language Processing<br>4. Advanced UNIX Programming | 3 | 0 | 0 | 3 |
| 4 | PE | R25D58107<br>R25D58108<br>R25D58109<br>R25D58110 | **Professional Elective-2**<br>1. Applied Cryptography<br>2. Software Quality Engineering<br>3. Mining Massive Datasets<br>4. Agile Methodologies | 3 | 0 | 0 | 3 |
| 5 | PC | R25D58111 | **Laboratory-1**<br>Advanced Data Structures Lab | 0 | 0 | 4 | 2 |
| 6 | PE | R25D58112<br>R25D58113<br>R25D58114<br>R25D58115 | **Professional Elective Laboratory-1**<br>1. Database Programming with PL/SQL Lab<br>2. Deep Learning Lab<br>3. Natural Language Processing Lab<br>4. Advanced UNIX Programming Lab | 0 | 0 | 4 | 2 |
| 7 | CC | R25D58116 | Research Methodology & IPR | 2 | 0 | 0 | 2 |
| 8 | AC | R25D58117<br>R25D58118 | **Audit Course – I**<br>1. English for Research Paper Writing<br>2. Disaster Management | 2 | 0 | 0 | 0 |
| | | | **Total** | **16** | **0** | **08** | **18** |

| Category | Courses | Credits |
|---|---|---|
| Professional Core Course | 03 | 08 |
| Professional Elective Course | 03 | 08 |
| Compulsory Course | 01 | 02 |
| Audit Course | 1 | 0 |
| Total | 08 | 18 |

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Program: M. Tech Computer Science and Engineering** **Regulation: R25**

**I Year II Semester- Course Structure**

| S.No | Category | Course Code | Course Title | Hours per Week | | | |
|---|---|---|---|---|---|---|---|
| | | | | Lecture | Tutorial | Practical | Credits |
| 1 | PC | R25D58201 | **Professional Core-3**<br>Advanced Algorithms | 3 | 0 | 0 | 3 |
| 2 | PC | R25D58202 | **Professional Core-3**<br>Advanced Computer Architecture | 3 | 0 | 0 | 3 |
| 3 | PE | R25D58203<br>R25D58204<br>R25D58205<br>R25D58206 | **Professional Elective-3**<br>1. Enterprise Cloud Concepts<br>2. Cyber Security<br>3. Parallel computing<br>4. Large Language Models | 3 | 0 | 0 | 3 |
| 4 | PE | R25D58207<br>R25D58208<br>R25D58209<br>R25D58210 | **Professional Elective-4**<br>1. Bioinformatics<br>2. Adhoc Sensor Networks<br>3. Robotic Process Automation<br>4. Generative AI | 3 | 0 | 0 | 3 |
| 5 | PC | R25D58211 | **Laboratory-2**<br>Advanced Algorithms Lab | 0 | 0 | 4 | 2 |
| 6 | PE | R25D58212<br>R25D58213<br>R25D58214<br>R25D58215 | **Professional Elective Laboratory-2**<br>1. Enterprise Cloud Concepts<br>2. Cyber Security<br>3. Parallel computing<br>4. Large Language Models | 0 | 0 | 4 | 2 |
| 7 | PJ | R25D58216 | Mini Project with Seminar | 2 | 0 | 0 | 2 |
| 8 | AC | R25D58217<br>R25D58218 | **Audit Course – II**<br>1. Constitution of India<br>2. Pedagogy Studies | 2 | 0 | 0 | 0 |
| | | | **Total** | **16** | **0** | **08** | **18** |

| Category | Courses | Credits |
|---|---|---|
| Professional Core Courses | 3 | 08 |
| Professional Elective Courses | 3 | 08 |
| Project | 1 | 02 |
| Audit Course | 1 | 0 |
| Total | 08 | 18 |

**Chairperson**
**Board of Studies (CSE)**

**MTCS1102** **Advanced Data Structures** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to:

1. Introduce heap data structures such as leftist trees, binomial heaps, Fibonacci heaps, and min-max heaps.
2. Introduce advanced data structures such as disjoint sets, hash tables, and efficient search structures.
3. Enable students to understand digital search structures and their applications.
4. Develop the ability to analyze and implement efficient string pattern matching algorithms.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| **MTCS1102.1** | Understand and implement heap structures such as min-max heaps, leftist trees, binomial heaps, and Fibonacci heaps. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L1, L2 |
| **MTCS1102.2** | Apply hashing techniques and collision resolution strategies to design efficient hash tables. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L2, L3 |
| **MTCS1102.3** | Analyze and implement advanced search structures such as AVL trees, red-black trees, splay trees, and B-trees. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 3 | L3, L4 |
| **MTCS1102.4** | Develop digital search structures like tries, suffix trees, and Patricia trees for solving real-world problems. | 3 | 3 | 3 | 3 | 1 | 2 | 3 | 3 | L4, L5 |
| **MTCS1102.5** | Evaluate and implement efficient string matching algorithms such as KMP, Boyer–Moore, and Rabin–Karp. | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I: Heap Structures** **(15 Hours)**

- Introduction, Min-Max Heaps, Leftist Trees, Binomial Heaps, Fibonacci Heaps. CO's-CO1

**Self-Learning Topics**: Applications of heaps in priority queues, amortized analysis of Fibonacci heaps.

**UNIT II: Hashing and Collisions** **(10 Hours)**

- Introduction to Hash Tables, Hash Functions.
- Division Method, Multiplication Method, Mid-Square Method, Folding Method.
- Handling Collisions: Chaining, Open Addressing, Double Hashing. CO's-CO2

**Self-Learning Topics**: Perfect hashing, cuckoo hashing.

**UNIT III: Search Structures** **(15 Hours)**

- OBST (Optimal Binary Search Trees), AVL Trees, Red-Black Trees, Splay Trees.
- Multiway Search Trees: B-trees, 2-3 Trees. CO's-CO3

**Self-Learning Topics**: B+ trees, real-time applications in databases.

**UNIT IV: Digital Search Structures** **(10 Hours)**

- Digital Search Trees, Binary Tries, Patricia Tries.
- Multiway Tries, Standard Tries, Compressed Tries, Suffix Trees. CO's-CO4

**Self-Learning Topics**: Applications in text indexing, genome sequencing.

**UNIT V: Pattern Matching** **(10 Hours)**

- Introduction to String Matching.
- Brute Force, Naïve String Matching.
- Boyer–Moore Algorithm, Knuth-Morris-Pratt (KMP), Horspool's Algorithm, Rabin–Karp Algorithm. CO's - CO5

**Self-Learning Topics**: Real-world applications in search engines, bioinformatics, and intrusion detection.

Board of Studies : Computer Science and Engineering
Approved in BOS No: -- , August, 2025
Approved in ACM No: 01

**Reference Books**

1. Sahni, Horowitz, Mehta, *Fundamentals of Data Structures in C++*, Universities Press.
2. T.H. Cormen, *Introduction to Algorithms*, PHI.

**Additional References**

1. S.K. Basu, *Design Methods and Analysis of Algorithms*, PHI.
2. Mark Allen Weiss, *Data Structures & Algorithm Analysis in C++*, Pearson Education.
3. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, *Fundamentals of Computer Algorithms* (2nd Edition), Universities Press.

**Online Learning Resources / Virtual Labs**

1. MIT OpenCourseWare – Advanced Data Structures
2. NPTEL – Data Structures and Algorithms
3. Coursera – Algorithms Specialization

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define min-max heaps with an example.
2. What is a Fibonacci heap?
3. Define a hash table and list different hash functions.
4. What is an AVL tree?
5. Define a B-tree with an example.
6. What is a Patricia trie?
7. List the different string matching algorithms.
8. Define the Rabin-Karp algorithm.
9. What is a suffix tree?
10. What is the division method in hashing?

**L2: Understand**

1. Explain the working of binomial heaps with an example.
2. Discuss the role of collision resolution strategies in hashing.
3. Differentiate between AVL trees and red-black trees.
4. Explain splay tree operations.
5. Explain the structure of a 2-3 tree with an example.
6. Discuss the role of tries in digital search structures.
7. Explain how suffix trees are used in text indexing.
8. Compare Boyer–Moore and KMP algorithms.
9. Explain the significance of load factor in hash tables.
10. Explain compressed tries with an example.

**L3: Apply**

1. Construct a min-max heap for the following sequence of numbers: 10, 5, 20, 3, 15.
2. Apply the mid-square method for hashing the keys {12, 25, 32, 47}.
3. Write an algorithm to insert a node into an AVL tree.
4. Perform deletion in a red-black tree.
5. Write a program to build a trie for a given dictionary of words.
6. Apply Horspool's algorithm to match a pattern in a given text.
7. Demonstrate insertion in a B-tree of order 3.
8. Apply Rabin-Karp algorithm to match the pattern "ABA" in text "ABABABA".
9. Write a program to implement open addressing in hashing.
10. Show rotations in AVL trees during insertion.

**L4: Analyze**

1. Analyze the amortized complexity of Fibonacci heap operations.
2. Compare hashing methods: division vs multiplication method.
3. Analyze the differences between B-trees and 2-3 trees.
4. Discuss the advantages of red-black trees over AVL trees.
5. Compare Patricia tries with standard tries.
6. Analyze the working of KMP algorithm.
7. Compare chaining vs open addressing in collision handling.
8. Differentiate suffix trees and suffix arrays.
9. Analyze performance of Horspool vs Boyer–Moore.
10. Discuss trade-offs between space and time in digital search structures.

**L5: Evaluate**

1. Evaluate the efficiency of Fibonacci heaps vs binary heaps.
2. Evaluate the impact of collisions on hash table performance.
3. Evaluate the role of balancing in search trees.
4. Compare efficiency of AVL, red-black, and splay trees.
5. Evaluate the use of tries in IP routing.
6. Evaluate performance differences between Rabin-Karp and KMP algorithms.
7. Critically analyze the role of suffix trees in genome sequencing.
8. Evaluate the limitations of B-trees in large databases.
9. Compare the efficiency of Boyer–Moore vs brute force.
10. Evaluate hashing in distributed systems.

**L6: Create**

1. Design and implement a Fibonacci heap with insert and delete operations.
2. Create a hash table using chaining for a set of student records.
3. Construct an AVL tree for a given dataset and demonstrate rotations.
4. Develop a red-black tree and test with insert and delete operations.
5. Create a compressed trie for a dictionary.
6. Develop a suffix tree for the string “BANANA”.
7. Implement KMP algorithm for string pattern matching.
8. Create a simulation of IP routing using tries.
9. Implement Rabin-Karp algorithm for plagiarism detection.
10. Design a program to compare performance of different string matching algorithms.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11031 DATABASE PROGRAMMING WITH PL/SQL 3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Knowledge on significance of SQL fundamentals.
2. Evaluate functions and triggers of PL/SQL.
3. Knowledge on control structures, packages in PL/SQL and its applications.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11031.1 | Understand importance of PL/SQL basics | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11031.2 | Implement functions and procedures using PL/SQL | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11031.3 | Understand the importance of triggers in database | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L4, L5 |

**SYLLABUS**

**UNIT I: PL/SQL Basics (15 Hours)**

Block Structure, Behavior of Variables in Blocks, Basic Scalar and Composite Data Types, Control Structures, Exceptions, Bulk Operations, Functions, Procedures, and Packages, Transaction Scope. **CO's-CO1**

Self-Learning Topics: Advanced Exception Handling, Nested Blocks.

**UNIT II: Language Fundamentals & Control Structures (10 Hours)**

Lexical Units, Variables and Data Types, Conditional Statements, Iterative Statements, Cursor Structures, Bulk Statements, Introduction to Collections, Object Types: Varray and Table Collections, Associative Arrays, Oracle Collection API. **CO's-CO2**

Self-Learning Topics: Cursor FOR loops, Advanced Collection Techniques.

**UNIT III: Functions and Procedures (15 Hours)**

Function and Procedure Architecture, Transaction Scope, Calling Subroutines, Positional Notation, Named Notation, Mixed Notation, Exclusionary Notation, SQL Call Notation, Functions, Function Model Choices, Creation Options, Pass-by-Value Functions, Pass-by-Reference Functions, Procedures, Supporting Scripts. **CO's – CO3**

Self-Learning Topics: User-defined Functions for Business Applications.

**UNIT IV: Packages (10 Hours)**

Package Architecture, Package Specification, Prototype Features, Serially Reusable Precompiler Directive, Variables, Types, Components: Functions and Procedures, Package Body, Definer vs. Invoker Rights Mechanics, Managing Packages in the Database Catalog, Validating and Describing Packages, Checking Dependencies. **CO's-CO4**

Self-Learning Topics: Building Reusable Packages for Enterprise Applications.

**UNIT V: Triggers (10 Hours)**

Introduction to Triggers, Database Trigger Architecture, DDL Triggers, Event Attribute Functions, Building DDL Triggers, DML Triggers, Statement-Level Triggers, Row-Level Triggers, Compound Triggers, INSTEAD OF Triggers, System and Database Event Triggers, Restrictions, SQL Statements. **CO's – CO5**

Self-Learning Topics: Security and Auditing with Triggers.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Text Books**

1. Oracle Database 12c PL/SQL Programming – Michael McLaughlin, McGraw Hill Education.

**References Books**

1. Benjamin Rosenzweig, Elena Silvestrova Rakhimov, Oracle PL/SQL by Example, Fifth Edition.
2. Dr. P. S. Deshpande, SQL & PL / SQL for Oracle 11g Black Book.

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Explain the block structure of PL/SQL with an example.
2. Define scalar and composite data types in PL/SQL.
3. What are exceptions in PL/SQL? Give examples.
4. List and explain PL/SQL control structures.
5. Define a procedure in PL/SQL with an example.

**L2: Understand**

1. Explain lexical units in PL/SQL with examples.
2. Discuss cursor structures in PL/SQL with examples.
3. Differentiate between positional, named, and mixed notations in procedures.
4. Explain package specification and body with examples.
5. Discuss the role of triggers in database programming.

**L3: Apply**

1. Write a PL/SQL block to add two numbers and print the result.
2. Write a PL/SQL program to calculate factorial using a loop.
3. Write a PL/SQL procedure to display employee details from a table.
4. Write a PL/SQL function to calculate the area of a circle.
5. Write a PL/SQL block using cursors to fetch records from a table.

**L4: Analyzing**

1. Compare procedures and functions in PL/SQL.
2. Discuss differences between implicit and explicit cursors.
3. Analyze the role of triggers in maintaining database integrity.
4. Differentiate between package specification and package body.
5. Differentiate between row-level and statement-level triggers.

**L5: Evaluating**

1. Evaluate the importance of packages in PL/SQL applications.
2. Critically analyze the role of exception handling in PL/SQL.
3. Evaluate performance considerations in using cursors.
4. Evaluate the advantages of using bulk operations in PL/SQL.
5. Evaluate the role of triggers in database auditing.

**L6: Create**

1. Create a PL/SQL package for employee management (functions and procedures).
2. Write a PL/SQL trigger to prevent deletion of records from a table.
3. Develop a PL/SQL program to implement transaction control.
4. Create a PL/SQL block that demonstrates bulk collect operation.
5. Design a PL/SQL function to calculate salary increments for employees.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11032** **Deep Learning** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To understand complexity of Deep Learning algorithms and their limitations.
2. To be capable of performing experiments in Deep Learning using real-world data.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11032.1 | Implement deep learning algorithms, understand neural networks and traverse the layers of data | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L2, L3 |
| MTCS11032.2 | Learn topics such as CNNs, RNNs, training deep networks and high-level interfaces | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 2 | L3, L4 |
| MTCS11032.3 | Understand applications of Deep Learning to Computer Vision | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | L4, L5 |
| MTCS11032.4 | Understand and analyze applications of Deep Learning to NLP | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | L5, L6 |

**SYLLABUS**

**UNIT I: Introduction to Deep Learning** **(15 Hours)**

Feed forward Neural Networks, Gradient Descent and Backpropagation Algorithm, Unit Saturation, the Vanishing Gradient Problem and mitigation techniques, ReLU, Heuristics for avoiding bad local minima, Faster Training Heuristics, Nestorov's Accelerated Gradient Descent, Regularization, Dropout. **CO's-CO1**

Self-Learning Topics: Optimization pitfalls in deep learning advanced activation functions.

**UNIT II: CNNs, RNNs, and Deep Unsupervised Learning** **(15 Hours)**

Convolutional Neural Networks: Architectures, Convolution/Pooling Layers. Recurrent Neural Networks: LSTM, GRU, Encoder–Decoder Architectures. Deep Unsupervised Learning: Auto encoders, Variational Autoencoders, Adversarial Generative Networks, DBMs. Attention and Memory Models, Dynamic Memory Models. **CO's – CO2**

Self-Learning Topics: Recent advances in GANs and Transformers.

**UNIT III: Deep Learning for Computer Vision** **(10 Hours)**

Image Segmentation, Object Detection, Automatic Image Captioning, Image Generation with GANs, Video-to-Text with LSTMs, Attention Models for Vision Tasks. **CO's-CO3**

Self-Learning Topics: Transfer Learning and Pretrained CNN models for Vision.

**UNIT IV: Deep Learning for NLP** **(10 Hours)**

Introduction to NLP, Vector Space Model of Semantics. Word Vector Representations: Skip-Gram, CBOW, GloVe. Evaluation and Applications in Word Similarity. **CO's – CO4**
Self-Learning Topics: Transformer-based models (BERT, GPT).

**UNIT V: Advanced NLP Applications** **(10 Hours)**
Analogy Reasoning, Named Entity Recognition, Opinion Mining using RNNs. Parsing and Sentiment Analysis using Recursive Neural Networks. Sentence Classification with CNNs. Dialogue Generation with LSTMs. **CO's – CO5**
Self-Learning Topics: Dialogue systems with attention and reinforcement learning.

Board of Studies : Computer Science and Engineering
Approved in BOS No: -- , August, 2025
Approved in ACM No: 01

**Text Books**

1. Deep Learning – Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press.
2. The Elements of Statistical Learning – T. Hastie, R. Tibshirani, J. Friedman, Springer.
3. Probabilistic Graphical Models – D. Koller, N. Friedman, MIT Press.

**References Books**

1. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.
2. Yegnanarayana, B., *Artificial Neural Networks*, PHI Learning, 2009.
3. Golub, G. H., Van Loan, C. F., *Matrix Computations*, JHU Press, 2013.
4. Satish Kumar, *Neural Networks: A Classroom Approach*, Tata McGraw-Hill, 2004.

**Extensive Reading**

- http://www.deeplearning.net
- https://www.deeplearningbook.org/
- https://developers.google.com/machine-learning/crash-course/ml-intro
- www.cs.toronto.edu/~fritz/absps/imagenet.pdf
- http://neuralnetworksanddeeplearning.com/

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define a feed-forward neural network.

2. Explain the vanishing gradient problem.
3. What is the role of activation functions in deep learning?
4. List types of regularization techniques.
5. Define autoencoder with example.

**L2: Understand**

1. Explain the backpropagation algorithm with an example.
2. Discuss convolution and pooling operations in CNNs.
3. Explain the concept of recurrent connections in RNNs.
4. Describe dropout and its importance.
5. Discuss vector space models in NLP.

**L3: Apply**

1. Implement a simple neural network for digit classification (conceptual).
2. Train a CNN for image classification on a sample dataset.
3. Write steps to build a sentiment analysis system using RNNs.
4. Apply word embeddings for similarity detection.
5. Demonstrate the use of GANs for image generation.

**L4: Analyzing**

1. Compare CNNs and RNNs in terms of architecture and applications.
2. Analyze the limitations of autoencoders versus VAEs.
3. Differentiate between Skip-Gram and CBOW models.
4. Discuss differences between LSTM and GRU.
5. Analyze the challenges in dialogue generation using RNNs.

**L5: Evaluating**

1. Evaluate ReLU vs Sigmoid activations.
2. Critically analyze advantages of transfer learning in computer vision.
3. Evaluate the impact of attention in NLP tasks.
4. Evaluate the strengths and weaknesses of GANs.
5. Compare recursive neural networks with CNNs for text classification.

**L6: Create**

1. Design a CNN model for image segmentation.
2. Develop an RNN-based chatbot using LSTMs.
3. Create a deep learning pipeline for opinion mining.
4. Design a GAN for generating synthetic images.
5. Implement a Transformer-based model for machine translation.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11033** **Natural Language Processing** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Introduction to some of the problems and solutions of NLP and their relation to linguistics and statistics.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11033.1 | Show sensitivity to linguistic phenomena and ability to model them with formal grammars | 3 | 2 | 1 | 2 | | 2 | 3 | 2 | L1, L2 |
| MTCS11033.2 | Understand and carry out proper experimental methodology for training and evaluating empirical NLP systems | 3 | 3 | 2 | 2 | 1 | 3 | 3 | 2 | L2, L3 |
| MTCS11033.3 | Manipulate probabilities, construct statistical models over strings/trees, and estimate parameters using supervised/unsupervised training | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | L3, L4 |
| MTCS11033.4 | Design, implement, and analyze NLP algorithms | 3 | 3 | 2 | 2 | 2 | 3 | 3 | 3 | L4, L5 |
| MTCS11033.5 | Design different language modeling techniques | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I: Finding Structure in Text** **(15 Hours)**

Words and their internal components, morphology and linguistic foundations, issues and challenges in text analysis, morphological models and approaches to handling word formation and variations. Documents – introduction to document representation, methods for structuring and analyzing documents, complexity of various approaches, performance evaluation of methods, and applications of text structuring in information retrieval and natural language processing.**CO's-CO1**

Self-Learning Topics: Subword tokenization (BPE, WordPiece).

**UNIT II: Syntax Analysis** **(10 Hours)**

Parsing natural language and its computational challenges, treebanks as a data-driven approach to syntax analysis, representation of syntactic structures and formalisms, parsing algorithms and their efficiency, ambiguity resolution in parsing using rule-based and statistical methods, and multilingual issues in syntax analysis with reference to structural differences and language diversity. **CO's – CO2**

Self-Learning Topics: Dependency parsing with modern NLP libraries.

**UNIT III: Semantic Parsing** **(10 Hours)**

Introduction to semantic interpretation and the role of meaning representation in natural language processing, system paradigms for semantic parsing, approaches to mapping syntactic structures to semantic forms, word sense systems and methods for word sense disambiguation, semantic resources and lexical databases, software tools and frameworks supporting semantic parsing, and applications of semantic interpretation in information extraction, question answering and machine translation. **CO's – CO3**

Self-Learning Topics: Word embeddings and contextual embeddings (Word2Vec, BERT).

**UNIT IV: Predicate-Argument Structures (10 Hours)**

Meaning representation systems and their role in natural language understanding, predicate-argument structures as a framework for capturing semantic relations, methods for representing predicates and their associated arguments, approaches to identifying and labeling semantic roles, computational models for predicate-argument analysis, tools and software for predicate-argument structure extraction, and applications in information extraction, machine translation, and question answering systems. **CO's – CO4**

Self-Learning Topics: FrameNet and PropBank role labeling.

**UNIT V: Discourse Processing & Language Modeling (15 Hours)**

Cohesion, Reference Resolution, Discourse Cohesion and Structure. Language Modeling: Introduction, N-Gram Models, Model Evaluation, Parameter Estimation, Model Adaptation, Types of Language Models, Multilingual and Cross-Lingual Issues. **CO's – CO5**

Self-Learning Topics: Transformer-based language models (BERT, GPT, XLNet).

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Text Books**

1. Multilingual Natural Language Processing Applications: From Theory to Practice – Daniel M. Bikel and Imed Zitouni, Pearson.
2. Natural Language Processing and Information Retrieval – Tanvier Siddiqui, U.S. Tiwary.

**Reference Books**

1. Speech and Natural Language Processing – Daniel Jurafsky & James H. Martin, Pearson.

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define morphological models in NLP.
2. What are Treebanks? Give examples.
3. Explain parsing in natural language with an example.
4. Define semantic parsing.
5. What is an N-gram language model?

**L2: Understand**

1. Explain issues in morphological analysis.
2. Discuss syntactic ambiguity resolution methods.
3. Explain word sense disambiguation systems.
4. Describe Predicate-Argument structure with example.
5. Discuss discourse cohesion in NLP.

**L3: Apply**

1. Apply a parsing algorithm to a given sentence.
2. Build a simple unigram/bigram model for a dataset.
3. Implement a word sense disambiguation system using supervised learning.
4. Construct a semantic role labeling task using PropBank dataset.
5. Apply cohesion techniques for reference resolution.

**L4: Analyze**

1. Compare supervised vs unsupervised training methods in NLP.
2. Analyze multilingual issues in parsing.
3. Differentiate between Treebank parsing and dependency parsing.
4. Compare semantic interpretation methods.
5. Analyze types of language models and their challenges.

**L5: Evaluate**

1. Evaluate statistical models for NLP tasks.
2. Critically analyze parameter estimation techniques in N-gram models.
3. Evaluate supervised vs unsupervised training in semantic parsing.
4. Compare performance of probabilistic vs neural NLP models.
5. Evaluate cross-lingual NLP challenges.

**L6: Create**

1. Develop a parser for a small dataset using CFG.
2. Create a word embedding model for text similarity.

3. Build a simple sentiment analysis system using N-grams.
4. Design a semantic parser using supervised training.
5. Implement a transformer-based model for question answering.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11034** **Advanced UNIX Programming** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Understand and make effective use of Linux utilities and shell scripting language (bash) to solve problems.
2. Implement standard Linux utilities (ls, mv, cp, etc.) in C using system calls.
3. Develop systems programming skills including file system programming, process and signal management, and inter-process communication (IPC).
4. Gain fundamental skills to write network programs using sockets.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11034.1 | Work confidently in Linux environment using system utilities and commands. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11034.2 | Write shell scripts in bash to automate Linux administrative tasks. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11034.3 | Develop C programs using Unix system calls for file handling and process management. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L3, L4 |
| MTCS11034.4 | Apply inter-process communication techniques such as pipes, message queues, semaphores, and shared memory. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 3 | L4, L5 |
| MTCS11034.5 | Design and implement client-server programs using Unix sockets for network communication. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I:** **(15 Hours)**

Linux Utilities - File handling utilities, Security by file permissions, Process utilities, Disk utilities, Networking commands, Filters, Text processing utilities and Backup utilities.

Shell programming with Bourne again shell (bash) - Introduction, shell responsibilities, pipes and Redirection, here documents, running a shell script, the shell as a programming language, shell meta characters, file name substitution, shell variables, command substitution, shell commands, the environment, quoting, test command, control structures, arithmetic in shell, shell script examples, interrupt processing, functions, debugging shell scripts. **CO's – CO1**

**Self Learning Topics:** System Monitoring, Job Scheduling

**UNIT II:** **(10 Hours)**

Files and Directories - File Concept, File types, File System Structure, file metadata-Inodes, kernel support for files, system calls for file I/O operations- open, creat, read, write, close, lseek, dup2, file status information-stat family, file and record locking- fcntl function, file permissions - chmod,

fchmod, file ownership-chown, lchown, fchown, links-soft links and hard links – symlink, link, unlink. Directories - Creating, removing and changing Directories- mkdir, rmdir, chdir, obtaining current working directory-getcwd, Directory contents, Scanning Directories-opendir, readdir, closedir, rewinddir functions. **CO's – CO2**

**Self-Learning Topics**: File Compression , Disk Quotas

**UNIT III:** **(10 Hours)**

Process – Process concept, Layout of a C program image in main memory, Process environment-environment list, environment variables, getenv, setenv, Kernel support for process, process identification, process control - process creation, replacing a process image, waiting for a process, process termination, zombie process, orphan process, system call interface for process management-fork, vfork, exit, wait, waitpid, exec family, Process Groups, Sessions and Controlling Terminal, Differences between threads and processes.

Signals – Introduction to signals, Signal generation and handling, Kernel support for signals, Signal function, unreliable signals, reliable signals, kill, raise, alarm, pause, abort, sleep functions.

**CO's – CO3**

**Self-Learning Topics**: Thread Management, Signal Masking

**UNIT IV:** **(10 Hours)**

Interprocess Communication - Introduction to IPC, IPC between processes on a single computer system, IPC between processes on different systems, pipes-creation, IPC between related processes using unnamed pipes, FIFOs-creation, IPC between unrelated processes using FIFOs (Named pipes), differences between unnamed and named pipes, popen and pclose library functions. Message Queues - Kernel support for messages, APIs for message queues, client/server example. Semaphores - Kernel support for semaphores, APIs for semaphores, file locking with semaphores.

**CO's – CO4**

**Self-Learning Topics:** Shared Memory, Socket Programming

**UNIT V:** **(15 Hours)**

Shared Memory - Kernel support for shared memory, APIs for shared memory, shared memory example. Sockets - Introduction to Berkeley Sockets, IPC over a network, Client- Server model, Socket address structures (Unix domain and Internet domain),Socket system calls for connection oriented protocol and connectionless protocol, example-client/server programs-Single Server-Client connection, Multiple simultaneous clients, Socket options- setsockopt and fcntl system calls, Comparison of IPC mechanisms. **CO's – CO5**

**Self-Learning Topics:** Network Programming, Port Binding

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Text Books**

1. Unix System Programming using C++, T. Chan, PHI.
2. Advanced Programming in the Unix Environment, 2nd edition, W. R. Stevens and S. A. Rago, Pearson Education.
3. Unix Concepts and Applications, 4th Edition, Sumitabha Das, TMH.
4. Unix Network Programming, W. R. Stevens, PHI.

**Reference Books**


1. C Programming Language, Kernighan and Ritchie, PHI.
2. Beginning Linux Programming, 4th Edition, N. Matthew, R. Stones, Wrox, Wiley India Edition.
3. Unix for programmers and users, 3rd Edition, Graham Glass, King Ables, Pearson.
4. System Programming with C and Unix, A. Hoover, Pearson.
5. Unix System Programming, Communication, Concurrency and Threads, K. A. Robbins and S. Robbins, Pearson Education.
6. Unix shell Programming, S. G. Kochan and P. Wood, 3rd edition, Pearson Education.
7. Shell Scripting, S. Parker, Wiley India Pvt. Ltd.
8. Unix and Shell programming, B. A. Forouzan and R. F. Gilberg, Cengage Learning.
9. Linux System Programming, Robert Love, O'Reilly, SPD.


**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1 – Remembering**

1. Define inode in a Linux file system.
2. List the system calls used for file I/O operations.
3. What is a symbolic link?
4. Name the types of processes in Linux.
5. Define a zombie process.
6. List the different IPC mechanisms in Linux.
7. What is a FIFO in Linux?
8. Define Berkeley sockets.
9. Name the basic utilities used for text processing in Linux.
10. List shell meta-characters used in Bash scripting.

**L2 – Understanding**

1. Explain the difference between hard links and symbolic links.
2. Describe the difference between shared memory and message queues.
3. Explain how pipes work in Bash scripting.
4. Describe the difference between fork() and vfork().

5. Explain the use of environment variables in shell scripting.
6. Describe the role of semaphores in process synchronization.
7. Explain connection-oriented and connectionless sockets.
8. Describe the function of chmod and chown in Linux.
9. Explain the difference between threads and processes.
10. Describe how signals are handled in Linux.

**L3 – Applying**

1. Write a Bash script to display all running processes.
2. Write a C program to create a file, write data, and read it using system calls.
3. Demonstrate IPC using message queues between two processes.
4. Implement producer-consumer synchronization using semaphores.
5. Write a client-server program using TCP sockets.
6. Demonstrate the use of dup2() to redirect standard output to a file.
7. Write a script to monitor disk usage and display alerts.
8. Use opendir() and readdir() to list all files in a directory.
9. Demonstrate sending a signal from one process to another using kill().
10. Implement a shared memory segment and read/write data between two processes.

**L4 – Analyzing**

1. Compare message queues and shared memory for IPC efficiency.
2. Analyze the difference between popen() and pclose() usage.
3. Compare pipelines versus temporary files for command output.
4. Analyze the effect of file permissions on multi-user access.
5. Compare reliable and unreliable signal handling mechanisms.
6. Analyze the performance impact of semaphores in concurrent processes.
7. Compare TCP and UDP socket communication for reliability.
8. Analyze the differences between chmod and fchmod.
9. Compare threads and processes in terms of memory usage and context switching.
10. Analyze the trade-offs between named and unnamed pipes.

**L5 – Evaluating**

1. Evaluate the reliability of TCP over sockets in client-server communication.
2. Assess the pros and cons of using shared memory versus message queues.
3. Evaluate the efficiency of pipelines in Bash scripting.
4. Critically examine the use of semaphores for process synchronization.
5. Assess the use of file locking versus semaphores in concurrent file access.
6. Evaluate the effectiveness of chmod in enforcing file security.
7. Assess the advantages of using shell meta-characters in scripts.
8. Evaluate the performance of TCP sockets under multiple clients.
9. Critically analyze the design of a logging system using signals.
10. Evaluate the efficiency of using dup2() for redirection in scripts.

**L6 – Creating**

1. Design a shell script that automates system backup with logging.
2. Develop a client-server application supporting multiple clients using TCP sockets.
3. Create a program that synchronizes multiple processes using semaphores.
4. Design a Bash script that monitors disk usage and sends alerts.
5. Develop a program to implement shared memory communication between processes.

6. Create a C program that demonstrates message passing with multiple processes.
7. Design a script that parses log files and generates a summary report.
8. Develop a program that handles multiple signals and logs events.
9. Create a TCP-based chat application for multiple users.
10. Design a system to automate file organization into directories based on file type.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11041** **Applied Cryptography** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Knowledge on significance of cryptographic protocols and symmetric and public key algorithms

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11041.1 | Understand the various cryptographic protocols. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11041.2 | Analyze key length and algorithm types and modes. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11041.3 | Illustrate different public key algorithms in cryptosystems. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L3, L4 |
| MTCS11041.4 | Understand special algorithms for protocols and usage in real-world systems. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L4, L5 |
| MTCS11041.5 | Apply real-world cryptographic standards and tools (e.g., Kerberos, PGP, PKCS). | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**Unit I: Foundations:** **(15 Hours)**

Terminology, Steganography, Substitution Ciphers and Transposition Ciphers, Simple XOR, One Time Pads, Computer Algorithms, Large Numbers, Cryptographic Protocols: Protocol Building Blocks Introduction to Protocols, Communications Using Symmetric Cryptography, One-Way Functions, One-Way Hash Functions, Communications Using Public-Key Cryptography, Digital Signatures, Digital Signatures with Encryption, Random and Pseudo-Random-Sequence Generation **CO's – CO1**

**Self-Learning Topics:** Caesar cipher variations, Modern steganography techniques.

**Unit II Cryptographic Techniques** **(10 Hours)**

Key length: Symmetric Key length, Public key length, comparing symmetric and public key length. Algorithm types and modes: Electronic Codebook Mode, Block Replay, Cipher Block Chaining Mode, Stream Cipher, Self-Synchronizing Stream Ciphers, Cipher-Feedback Mode, Synchronous Stream Ciphers, Output-Feedback Mod, Counter Mode, Other Block-Cipher Modes. **CO's – CO2**

**Self-Learning Topics:** Attacks on modes of operation.

**Unit III Public-Key Algorithms** **(15 Hours)**

Background, Knapsack Algorithms, RSA, Pohlig-Hellman, Rabin, ElGamal, McEliece, Elliptic Curve Cryptosystems, LUC, Finite Automaton Public-Key Cryptosystems

Public-Key Digital Signature Algorithms: Digital Signature Algorithm (DSA), DSA Variants, Gost Digital Signature Algorithm, Discrete Logarithm Signature Schemes, Ong-Schnorr-Shamir, ESIGN **CO's-CO3**

**Self-Learning Topics:** Quantum-safe public key algorithms.

**Unit IV Special Algorithms for Protocols** **(10 Hours)**

Multiple-Key Public-Key Cryptography, Secret-Sharing Algorithms, Subliminal Channel, Undeniable Digital Signatures, Designated Confirmer Signatures, Computing with Encrypted Data, Fair Coin Flips, One-Way Accumulators, All-or-Nothing Disclosure of Secrets, Fair and Failsafe Cryptosystems, Zero-Knowledge Proofs of Knowledge, Blind Signatures, Oblivious Transfer, Secure Multiparty Computation, Probabilistic Encryption, Quantum Cryptography. **CO's–CO4**
**Self-Learning Topics:** Blockchain and cryptographic protocols.

**Unit V Real World Approaches** **(10 Hours)**

IBM Secret key management protocol, ISDN, Kerberos, KryptoKnight, Privacy enhanced mail (PEM), Message security protocol (MSP), PGP, Public-Key Cryptography Standards (PKCS), Universal Electronic Payment System (UEPS). **CO's – CO5**
**Self-Learning Topics:** TLS/SSL, Secure messaging protocols.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Bruce Schneier, Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth).

**Reference Books:**

1. Bruce Schneier, Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C.
2. William Stallings, Cryptography and Network Security: Principles and Practice, Pearson.
3. Menezes, Van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press.

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Cryptography and Network Security
2. Coursera – Applied Cryptography Specialization
3. edX – Computer Security and Applied Cryptography
4. MIT OCW – Computer Systems Security (Cryptography modules)

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define substitution and transposition ciphers with examples.
2. What is a one-way function? Give examples.
3. Define RSA algorithm.
4. What is a digital signature? Explain.
5. Define Kerberos protocol.
6. What is PGP?
7. Define symmetric and asymmetric encryption.
8. What is a zero-knowledge proof?
9. Explain one-time pad.
10. Define steganography with an example.

**L2: Understand**

1. Explain key length in symmetric and public key cryptography.
2. Discuss stream cipher and block cipher differences.
3. Explain cipher block chaining with example.
4. Explain RSA key generation process.
5. Discuss the purpose of one-way hash functions.
6. Explain the difference between digital signatures and message authentication codes.
7. Describe quantum cryptography basics.
8. Explain Kerberos authentication steps.
9. Explain how PGP provides email security.
10. Discuss role of PKCS in cryptographic standards.

**L3: Apply**

1. Write steps of RSA key generation and encryption.
2. Apply ElGamal encryption on a sample plaintext.
3. Write a simple substitution cipher program.
4. Demonstrate Caesar cipher with key 3.
5. Implement XOR encryption for a given string.
6. Show how to generate digital signatures using RSA.
7. Apply hash functions (SHA-256) to a message.
8. Demonstrate DES encryption process.
9. Show how to sign a document with DSA.
10. Implement a simple secret sharing scheme.

**L4: Analyze**

1. Differentiate between block cipher modes (ECB, CBC, OFB).
2. Analyze strengths and weaknesses of Rabin cryptosystem.
3. Compare RSA and ECC in terms of efficiency.
4. Differentiate between symmetric and public key cryptography.
5. Analyze security issues with one-time pad.
6. Compare PGP and PEM for email security.
7. Discuss Kerberos limitations in distributed systems.
8. Differentiate between hash function and MAC.
9. Analyze security of ElGamal encryption.
10. Compare blind signature and digital signature.

**L5: Evaluate**

1. Evaluate Kerberos authentication protocol.
2. Compare DSA and RSA digital signatures.
3. Evaluate security of ECC over RSA.
4. Assess DES vulnerabilities.
5. Evaluate TLS/SSL as a cryptographic standard.
6. Compare PKCS standards for real-world usage.
7. Evaluate PGP's role in secure communications.
8. Critically analyze blockchain cryptography.
9. Evaluate the need of quantum cryptography.
10. Compare symmetric vs. public key performance in practice.

**L6: Create**

1. Design a simple zero-knowledge proof protocol.
2. Create an RSA-based encryption/decryption program.
3. Build a Python program to generate a digital signature.
4. Create a simple implementation of Caesar cipher.
5. Design a Kerberos-like authentication system.
6. Create a blockchain simulation with digital signatures.
7. Demonstrate ElGamal encryption programmatically.
8. Build a secret sharing scheme in Python.
9. Design a secure multiparty computation example.
10. Create a protocol using hash-based message authentication.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11042** **Software Quality Engineering** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Knowledge on significance of Quality, quality assurance, quality engineering.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11042.1 | Understand software quality and its perspectives. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11042.2 | Analyze defect prevention and defect reduction in software quality assurance. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11042.3 | Illustrate software quality engineering activities and process. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L3, L4 |
| MTCS11042.4 | Apply test planning, execution, management, and automation techniques. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L4, L5 |
| MTCS11042.5 | Evaluate coverage and usage-based statistical testing for real-world applications. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

**SYLLABUS**

**UNIT I: Software Quality** **(15 Hours)**

Quality: perspectives and expectations, Quality frameworks and ISO-9126, correctness and defects: Definitions, properties and Measurements, A historical perspective of quality, software quality.

**CO's – CO1**

**Self-Learning Topics**: Evolution of quality models, Case studies on ISO standards.

**UNIT II: Quality Assurance** **(10 Hours)**

Classification: QA as dealing with defects, Defect prevention – Education and training, Formal method, Other defect prevention techniques.Defect Reduction – Inspection: Direct fault detection and removal, Testing: Failure observation and fault removal, Other techniques and risk identification. Defect Containment – Software fault tolerance, Safety assurance and Failure containment. **CO's – CO2**

**Self-Learning Topics**: Statistical defect prediction models.

**UNIT III: Quality Engineering** **(15 Hours)**

Quality Engineering: Activities and process, Quality planning: Goal setting and Strategy formation, Quality assessment and Improvement, Quality engineering in software process. **CO's – CO3**

**Self-Learning Topics:** Six Sigma and CMMI in software quality.

**UNIT IV: Test Activities, Management and Automation** **(10 Hours)**

Test planning and preparation, Test execution, Result checking and measurement, Analysis and follow-up.

Activities, People and Management, Test Automation.

**CO's – CO4**

**Self-Learning Topics:** Automated testing tools (Selenium, JUnit).

**UNIT V: Coverage and Usage Testing** **(10 Hours)**

Checklist based testing and its limitations, Testing for partition coverage.

Usage-based Statistical testing with Musa's operational profiles, Constructing operational profiles.

Case Study: Operational profile for the Cartridge Support Software.

**CO's – CO5**

**Self-Learning Topics:** AI in test case generation, Model-based testing.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Reference Books:**

1. Jeff Tian, *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*
2. Richard N. Taylor, *Software Architecture: Foundations, Theory, and Practice*

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Software Testing & Quality Assurance
2. Coursera – Software Testing and Automation Specialization
3. edX – Software Engineering Essentials: Quality & Testing
4. IEEE Xplore Digital Library – Software Quality Engineering case studies

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define software quality and explain its importance.
2. What is ISO-9126? Explain its dimensions.
3. Define software defect and correctness.
4. State different perspectives of quality.
5. What is quality assurance?
6. Define fault tolerance in software.
7. What are software metrics?
8. Define operational profile in testing.
9. List activities in software quality engineering.
10. Define defect prevention with example.

**L2: Understand**

1. Explain the perspectives of software quality with examples.
2. Discuss ISO-9126 framework in detail.
3. Explain defect reduction techniques.
4. Describe inspection and testing as defect detection methods.
5. Discuss software failure containment methods.
6. Explain role of training in defect prevention.
7. Discuss goal setting in quality planning.
8. Explain risk identification in quality assurance.
9. Discuss statistical testing using operational profiles.
10. Explain the role of automation in quality assurance.

**L3: Apply**

1. Apply defect prevention through training in a software project.
2. Show how to prepare a quality plan for a software product.
3. Write steps to perform inspection in defect reduction.
4. Demonstrate test planning and execution with a case.
5. Apply Musa's operational profile for usage-based testing.
6. Create a checklist for testing.
7. Apply quality assessment on a software project.
8. Demonstrate automation using a tool like Selenium.
9. Show usage-based statistical testing steps.
10. Apply improvement cycle for a software process.

**L4: Analyze**

1. Differentiate between defect prevention, reduction, and containment.
2. Compare ISO-9126 with other quality models.
3. Analyze limitations of checklist-based testing.
4. Differentiate testing and inspection methods.
5. Compare manual testing with test automation.
6. Analyze effectiveness of operational profiles.
7. Compare software metrics in quality measurement.
8. Analyze impact of defect reduction on reliability.
9. Differentiate quality planning vs. quality assurance.
10. Compare statistical vs. non-statistical testing.

**L5: Evaluate**

1. Evaluate the strengths of ISO-9126 quality framework.
2. Critically analyze defect containment approaches.
3. Evaluate automation's impact on test efficiency.
4. Assess Musa's operational profile approach.
5. Evaluate software quality planning strategies.
6. Critically evaluate Six Sigma for software projects.
7. Compare quality assurance and software process improvement.
8. Evaluate effectiveness of risk identification in QA.
9. Compare different defect prevention techniques.
10. Evaluate statistical testing for reliability improvement.

**L6: Create**

1. Design a quality plan for a medium-scale software project.
2. Create an operational profile for a web application.
3. Develop a checklist for partition-based testing.
4. Create a fault tolerance mechanism for a safety-critical system.
5. Design a defect prevention training module.
6. Create a test automation framework using Selenium.
7. Develop a strategy for defect containment.
8. Construct a case study on defect reduction using inspections.
9. Design metrics for software quality assessment.
10. Create a model for AI-driven test case generation.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11043** **Mining Massive Datasets** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- This course will cover practical algorithms for solving key problems in mining of massive datasets.
- Focus on parallel algorithmic techniques used for large datasets.
- Stream processing algorithms for continuous data, page ranking algorithms for web search, and online advertisement systems are studied in detail.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11043.1 | Handle massive data using MapReduce. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11043.2 | Develop and implement algorithms for massive datasets and methodologies in the context of data mining. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11043.3 | Understand the algorithms for extracting models and information from large datasets. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L3, L4 |
| MTCS11043.4 | Develop recommendation systems. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L4, L5 |
| MTCS11043.5 | Gain experience in matching various algorithms for particular classes of problems. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I: Introduction & MapReduce** **(15 Hours)**

- Data Mining – Introduction, Definition of Data Mining, Statistical Limits on Data Mining.
- MapReduce and the New Software Stack – Distributed File Systems, MapReduce framework, Algorithms using MapReduce. **CO's – CO1**

  **Self-Learning Topics:** Hadoop and Spark ecosystems, Case studies on MapReduce applications.

**UNIT II: Similarity Search & Streaming Data** **(10 Hours)**

- Similarity Search: Finding Similar Items, Applications of Near-Neighbor Search, Shingling of Documents, Similarity-Preserving Summaries of Sets, Distance Measures.
- Streaming Data: Mining Data Streams – Stream Data Model, Sampling Data in a Stream, Filtering Streams. **CO's – CO2**

**Self-Learning Topics**: Locality Sensitive Hashing (LSH), Real-world stream mining systems (e.g., Twitter analytics).

**UNIT III: Link Analysis, Frequent Itemsets & Clustering (15 Hours)**

- Link Analysis: PageRank, Efficient Computation of PageRank, Link Spam.
- Frequent Itemsets: Handling Larger Datasets in Main Memory, Limited-Pass Algorithms, Counting Frequent Items in a Stream.
- Clustering: The CURE Algorithm, Clustering in Non-Euclidean Spaces, Clustering for Streams and Parallelism. **CO's – CO3**

  **Self-Learning Topics:** Graph-based ranking algorithms (HITS), Advanced clustering algorithms for big data.

**UNIT IV: Web Advertising & Recommendation Systems (10 Hours)**

- Advertising on the Web: Issues in On-Line Advertising, On-Line Algorithms, The Matching Problem, The Adwords Problem, Adwords Implementation.
- Recommendation Systems: A Model for Recommendation Systems, Content-Based Recommendations, Collaborative Filtering, Dimensionality Reduction, The Netflix Challenge. **CO's – CO4**

  **Self-Learning Topics:** Hybrid recommendation systems, Real-world recommender case studies (Amazon, YouTube, Netflix).

**UNIT V: Mining Social-Network Graphs (10 Hours)**

- Social Networks as Graphs, Clustering of Social-Network Graphs, Partitioning of Graphs, Simrank, Counting Triangles. **CO's – CO5**

  **Self-Learning Topics**: Community detection algorithms, Influence maximization in social networks.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Text Books:**

1. Jure Leskovec, Anand Rajaraman, Jeff Ullman, *Mining of Massive Datasets*, 3rd Edition.

**Reference Books:**

1. Jiawei Han & Micheline Kamber, *Data Mining – Concepts and Techniques*, 3rd Edition, Elsevier.
2. Margaret H Dunham, *Data Mining: Introductory and Advanced Topics*, Pearson.
3. Ian H. Witten and Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Mining Massive Datasets
2. Coursera – Big Data Specialization

3. edX – Data Mining and Machine Learning Fundamentals
4. Stanford Online – Mining Massive Datasets (by Jure Leskovec)

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define data mining.
2. What is MapReduce?
3. Define similarity search with example.
4. What is PageRank?
5. Define frequent itemset.
6. What is clustering?
7. Define collaborative filtering.
8. What is a social network graph?
9. Define SimRank.
10. What is the Netflix challenge?

**L2: Understand**

1. Explain the statistical limits of data mining.
2. Describe the MapReduce programming model.
3. Explain shingling of documents.
4. Discuss the stream data model.
5. Explain efficient computation of PageRank.
6. Describe limited-pass algorithms for frequent itemsets.
7. Explain the CURE clustering algorithm.
8. Discuss the Adwords problem.
9. Explain collaborative filtering in recommender systems.
10. Discuss the concept of SimRank in graph mining.

**L3: Apply**

1. Apply MapReduce to compute word frequencies in a dataset.

2. Implement a similarity-preserving hashing technique.
3. Use sampling techniques on a data stream.
4. Apply PageRank algorithm on a small web graph.
5. Implement a limited-pass algorithm for frequent items.
6. Apply clustering in non-Euclidean spaces.
7. Demonstrate online matching algorithm in advertising.
8. Apply collaborative filtering to movie recommendation.
9. Construct an operational profile for a recommender system.
10. Apply triangle counting in a social network graph.

**L4: Analyze**

1. Compare distributed file systems with traditional file systems.
2. Analyze different distance measures for similarity search.
3. Differentiate stream mining vs. batch mining.
4. Compare PageRank with HITS algorithm.
5. Analyze performance of frequent itemset algorithms.
6. Differentiate CURE clustering and k-means clustering.
7. Analyze challenges in online advertising algorithms.
8. Compare content-based and collaborative recommender systems.
9. Analyze partitioning methods in graph mining.
10. Compare usage-based and structure-based clustering in social networks.

**L5: Evaluate**

1. Evaluate advantages of MapReduce in large datasets.
2. Assess the limitations of shingling methods.
3. Evaluate effectiveness of stream filtering techniques.
4. Critically analyze PageRank against link spam.
5. Evaluate frequent itemset mining approaches.
6. Compare clustering methods for data streams.
7. Evaluate Adwords implementation strategies.
8. Critically assess Netflix challenge outcomes.
9. Evaluate SimRank in large-scale graphs.
10. Compare statistical vs. machine learning-based social network mining.

**L6: Create**

1. Design a MapReduce algorithm for inverted index generation.
2. Create a similarity search system for text documents.
3. Build a stream mining system for Twitter hashtags.
4. Create a PageRank computation for an academic citation graph.
5. Develop a limited-pass frequent itemset miner.
6. Create a clustering algorithm for high-dimensional data.
7. Build an online advertisement matching system.
8. Design a hybrid recommendation system combining content and collaborative filtering.
9. Develop a graph mining algorithm for community detection.
10. Create a real-world case study on mining social network data.
11.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11044** **Agile Methodologies** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Knowledge on concepts of Agile development, releasing, planning and developing **Course**

**Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO | PO | PO | PO | PO | PO | PSO 1 | PSO 2 | Do |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS11044.1 | Understand basic concepts of agile methods and extreme programming. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS11044.2 | Analyze real customer involvement and ubiquitous language. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS11044.3 | Discuss risk management and iteration planning. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L3, L4 |
| MTCS11044.4 | Summarize incremental requirements, refactoring, incremental design and architecture. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L4, L5 |
| MTCS11044.5 | Apply Agile practices like TDD, continuous integration, and performance optimization in projects. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I: Introduction & Extreme Programming** **(15 Hours)**

Agile Development: Why Agile – Understanding Success, Beyond Deadlines, Organizational Success, Introduction to Agility.

Agile methods, Road to Mastery, XP Life Cycle, XP Team, XP Concepts.

**Adopting XP:** Suitability, Implementation, Assessing Agility.

**Practicing XP:** Pair Programming, Energized Work, Informative Workspace, Root Cause Analysis, Retrospectives. **CO's – CO1**

**Self-Learning Topics:** Agile manifesto, Case studies of Agile adoption.

**UNIT II: Collaborating** **(10 Hours)**

Collaboration practices in software development emphasizing trust among team members, co-located work practices such as sitting together, and active involvement of real customers throughout the development process. Use of ubiquitous language to ensure consistent understanding across business and technical stakeholders. Conducting effective meetings for planning, coordination, and progress tracking. Implementation and adherence to coding standards to improve code readability, maintainability, and team collaboration. Practices for iteration demos, showcasing incremental progress to stakeholders, and structured reporting of progress, issues, and outcomes to facilitate transparent and continuous communication within the team. **CO's – CO2**

**Self-Learning Topics:** Agile team dynamics, Distributed Agile teams.

UNIT III: Releasing (10 Hours)

Practices for achieving a bug-free release, including rigorous testing, defect tracking, and quality assurance measures. Version control techniques for managing source code changes, branching, merging, and maintaining a reliable codebase. Fast build practices to accelerate compilation, packaging, and deployment processes. Continuous integration strategies for automating builds, running tests, and detecting integration issues early. Collective code ownership to encourage shared responsibility for code quality and maintainability. Importance of documentation for releases, including user manuals, technical guides, change logs, and release notes, to ensure clarity and smooth handover to stakeholders and support teams. **CO's – CO3**

**Self-Learning Topics**: CI/CD pipelines in Agile.

**UNIT IV: Planning (10 Hours)**

Version planning and release planning strategies to manage software evolution, scheduling, and timely delivery of features. Risk management practices including identification, assessment, mitigation, and monitoring of technical, project, and business risks. Iteration planning for defining the scope, selecting stories, and setting iteration goals. Use of slack to handle uncertainties and provide flexibility in scheduling. Story creation, prioritization, and refinement to capture functional requirements effectively. Estimation techniques for effort, time, and resources, including relative sizing, story points, and velocity tracking to support realistic planning and progress measurement. **CO's – CO4**

**Self-Learning Topics**: Agile estimation techniques (Planning Poker, Story Points).

**UNIT V: Developing (15 Hours)**

Managing incremental requirements and their implementation throughout the development process, ensuring alignment with customer expectations and evolving needs. Use of customer tests to validate functionality and gather feedback continuously. Test Driven Development (TDD) practices to write tests before code, ensuring correctness, maintainability, and early defect detection. Refactoring techniques to improve code structure, readability, and performance without changing external behavior. Incremental design and architecture approaches to evolve system structure iteratively, supporting flexibility and adaptability. Implementation of spike solutions for exploring technical challenges and reducing uncertainty. Performance optimization strategies to enhance efficiency, responsiveness, and scalability of the system. Conducting exploratory testing to uncover unexpected issues and validate software behavior in real-world scenarios.

**CO's – CO4, CO5**

**Self-Learning Topics:** Agile DevOps practices, Automated acceptance testing.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Text Book:**

1. James Shore and Shane Warden, *The Art of Agile Development*, 11th Indian Reprint, O'Reilly, 2018.

**Reference Books:**

1. Andrew Stellman and Jennifer Greene, *Learning Agile*, O'Reilly, 4th Indian Reprint, 2018.

2. Venkat Subramaniam and Andy Hunt, *Practices of an Agile Developer*, SPD, 5th Indian Reprint, 2015.
3. Jim Highsmith, *Agile Project Management*, Pearson Low Price Edition, 2004.

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Agile Software Development
2. Coursera – Agile Development Specialization
3. edX – Agile Project Management
4. Scrum.org – Agile and Scrum resources

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels L1: Remember**

1. Define Agile development.
2. What is Extreme Programming (XP)?
3. Define continuous integration.
4. What are Agile values?
5. Define iteration planning.
6. What is pair programming?
7. Define user stories.
8. What is refactoring?
9. Define exploratory testing.
10. What is spike solution?

**L2: Understand**

1. Explain XP life cycle with its phases.
2. Discuss Agile methods and their differences.
3. Explain the role of real customer involvement.
4. Describe ubiquitous language in Agile.
5. Explain bug-free release.
6. Discuss collective ownership in Agile projects.
7. Explain risk management in Agile planning.
8. Discuss test-driven development with example.
9. Explain exploratory testing.
10. Discuss role of retrospectives in Agile.

**L3: Apply**

1. Apply Agile principles to a software project example.
2. Demonstrate pair programming in solving a problem.
3. Apply iteration demo with a case study.
4. Implement continuous integration for a sample project.
5. Apply Agile release planning with stories.
6. Demonstrate TDD with a small Python/Java program.
7. Apply refactoring to improve design.
8. Show performance optimization in Agile iteration.
9. Apply story points in estimating project tasks.
10. Demonstrate documentation in Agile environment.

**L4: Analyze**

1. Differentiate XP and Scrum methodologies.
2. Compare Agile vs. traditional project management.
3. Analyze challenges in customer involvement.
4. Differentiate incremental design vs. traditional design.
5. Analyze advantages of continuous integration.
6. Compare test-driven development and customer tests.
7. Analyze effectiveness of spike solutions.
8. Differentiate Agile estimation vs. traditional estimation.
9. Compare Agile retrospectives and post-mortems.
10. Analyze role of performance optimization in Agile projects.

**L5: Evaluate**

1. Evaluate benefits of Agile over traditional methods.
2. Assess importance of coding standards in Agile.
3. Evaluate release planning strategies.
4. Critically analyze exploratory testing in Agile.
5. Compare collaborative vs. individual coding approaches.
6. Evaluate risk management in Agile projects.
7. Assess continuous integration tools in Agile workflow.
8. Evaluate incremental architecture approaches.
9. Critically assess Agile DevOps integration.

**L6: Create**

1. Design an Agile release plan for an e-commerce project.
2. Create user stories for a banking application.
3. Build a test-driven development cycle for login functionality.
4. Create a retrospective plan for an Agile team.
5. Develop a spike solution for a new feature.
6. Design an incremental architecture for a microservices system.
7. Create a pair programming session example.
8. Develop a case study on Agile adoption.

**Chairperson**
**Board of Studies (CSE)**

**MTCS1106 ADVANCED DATA STRUCTURES LAB 0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Introduces the basic concepts of Abstract Data Types.
2. Reviews basic data structures such as stacks and queues.
3. Introduces a variety of data structures such as hash tables, search trees, tries, heaps, graphs,
4. and B-trees.
5. Introduces sorting and pattern matching algorithms.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| **MTCS1106.1** | Ability to select data structures that efficiently model the information in a problem. | 3 | 3 | 3 | L1,L2 |
| **MTCS1106.2** | Ability to assess efficiency trade-offs among different data structure implementations. | 3 | 3 | 3 | L2,L3 |
| **MTCS1106.3** | Implement and know the application of algorithms for sorting and pattern matching. | 3 | 3 | 3 | L2,L3 |
| **MTCS1106.4** | Design programs using advanced data structures such as hash tables, search trees, tries, heaps, graphs, and B-trees. | 3 | 3 | 3 | L2,L3 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**SYLLABUS**

**Developing the following programs:**

Experiment 1:

Write a program to perform the following operations on a Binary Search Tree:

- Insert an element
- Delete an element
- Search for a key element

Experiment 2:

Write a program to implement the following sorting methods:

- Merge Sort
- Heap Sort
- Quick Sort

Experiment 3:

Write a program to perform the following operations on a B-Tree:

- Insert an element
- Delete an element
- Search for a key element

Experiment 4:

Write a program to perform the following operations on a Min-Max Heap:

- Insert an element
- Delete an element
- Search for a key element

Experiment 5:Write a program to perform the following operations on a Leftist Tree:

- Insert an element
- Delete an element
- Search for a key element

Experiment 6:Write a program to perform the following operations on a Binomial Heap:

- Insert an element
- Delete an element
- Search for a key element

Experiment 7:Write a program to perform the following operations on an AVL Tree:

- Insert an element
- Delete an element
- Search for a key element

Experiment 8:Write a program to perform the following operations on a Red-Black Tree:

- Insert an element
- Delete an element
- Search for a key element

Experiment 9:Write a program to implement all the functions of a Dictionary using Hashing.

Experiment 10:Write a program to implement Knuth-Morris-Pratt (KMP) pattern matching algorithm.

Experiment 11:Write a program to implement Brute Force pattern matching algorithm.

Experiment 12:Write a program to implement Boyer-Moore pattern matching algorithm.

**Text Books:**

1. *Fundamentals of Data Structures in C,* E. Horowitz, S. Sahni, Susan Anderson Freed, 2nd Edition, Universities Press.
2. *Data Structures Using C,* A.S. Tanenbaum, Y. Langsam, M.J. Augenstein, PHI/Pearson Education.
3. *Introduction to Data Structures in C,* Ashok Kamthane, 1st Edition, Pearson.

**Reference Books:**

1. *The C Programming Language*, B.W. Kernighan, Dennis M. Ritchie, PHI/Pearson Education.
2. *C Programming with Problem Solving*, J.A. Jones & K. Harrow, Dreamtech Press.
3. *Data Structures: A Pseudocode Approach with C,* R.F. Gilberg, B.A. Forouzan, 2nd Edition, Cengage Learning.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11071 DATABASE PROGRAMMING WITH PL/SQL LAB 0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Knowledge on significance of SQL fundamentals.
2. Evaluate functions and triggers of PL/SQL
3. Knowledge on control structures, packages in PL/SQL and its applications

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS11071.1 | Understand importance of PL/SQL basics. | 3 | 3 | 3 | L1,L2 |
| MTCS11071.2 | Implement functions and procedures using PL/SQL. | 3 | 3 | 3 | L2,L3 |
| MTCS11071.3 | Understand the importance of triggers in database systems. | 3 | 3 | 3 | L2,L3 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**SYLLABUS**

**Developing the following programs:**

**Experiment 1:**

Write a PL/SQL program using FOR loop to insert ten rows into a database table.

**Experiment 2:**

Given the table **EMPLOYEE(EmpNo, Name, Salary, Designation, DeptID)**, write a cursor to select the five highest paid employees.

**Experiment 3:**

Illustrate embedding PL/SQL in a high-level host language (C/Java) to demonstrate a banking debit transaction.

**Experiment 4:**

Given an integer *i*, write a PL/SQL procedure to insert the tuple *(i, 'xxx')* into a relation.

**Experiment 5:**

Write a PL/SQL program to demonstrate **Exceptions**.

**Experiment 6:**

Write a PL/SQL program to demonstrate **Cursors**.

**Experiment 7:**

Write a PL/SQL program to demonstrate **Functions**.

**Experiment 8:**

Write a PL/SQL program to demonstrate **Packages**.

**Experiment 9:**

Write PL/SQL queries to create **Procedures**.

**Experiment 10:**

Write PL/SQL queries to create **Triggers**.

**Text Books:**

1. *Oracle PL/SQL Programming* – Steven Feuerstein, Bill Pribyl, O'Reilly Media.
2. *Database System Concepts* – Abraham Silberschatz, Henry Korth, S. Sudarshan, 6th Edition, McGraw Hill.

**Reference Books:**

1. *Learning SQL* – Alan Beaulieu, O'Reilly Media.
2. *Fundamentals of Database Systems* – Ramez Elmasri, Shamkant B. Navathe, 7th Edition, Pearson.
3. *Oracle Database 12c PL/SQL Programming* – Michael McLaughlin, Oracle Press.

**Chairperson**
**Board of Studies (CSE)**

**MTCS11072** **Deep Learning Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To Build the Foundation of Deep Learning.
2. To Understand How to Build the Neural Network.
3. To enable students to develop successful machine learning concepts.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS11072.1 | Learn the fundamental principles of Deep Learning. | 3 | 3 | 3 | L1,L2 |
| MTCS11072.2 | Identify appropriate Deep Learning algorithms for various learning tasks across domains. | 3 | 3 | 3 | L2,L3 |
| MTCS11072.3 | Implement Deep Learning algorithms and solve real-world problems. | 3 | 3 | 3 | L2,L3 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**SYLLABUS**

**Developing the following programs:**

Experiment 1:

Setting up the Spyder IDE environment and executing a Python program.

Experiment 2:

Installing Keras, TensorFlow, and PyTorch libraries and demonstrating their use.

Experiment 3:

Applying a Convolutional Neural Network (CNN) on computer vision problems.

Experiment 4:

Image classification on MNIST dataset using CNN model with Fully Connected Layer.

Experiment 5:

Applying Deep Learning models in the field of Natural Language Processing (NLP).

Experiment 6:

Training a sentiment analysis model on IMDB dataset using RNN layers with LSTM/GRU nodes.

Experiment 7:

Applying Autoencoder algorithms for encoding real-world data.

Experiment 8:

Applying Generative Adversarial Networks (GANs) for image generation and unsupervised tasks.

**Text Books**

1. *Deep Learning* – Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press.
2. *The Elements of Statistical Learning* – T. Hastie, R. Tibshirani, J. Friedman, Springer.
3. *Probabilistic Graphical Models* – D. Koller, N. Friedman, MIT Press.

**Reference Books**

1. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.
2. Yegnanarayana, B., *Artificial Neural Networks*, PHI Learning, 2009.
3. Golub, G. H., Van Loan, C. F., *Matrix Computations*, JHU Press, 2013.
4. Satish Kumar, *Neural Networks: A Classroom Approach*, Tata McGraw-Hill, 2004.

**Extensive Reading**

1. Deep Learning Net
2. Deep Learning Book
3. Google ML Crash Course
4. ImageNet Paper
5. Neural Networks and Deep Learning

**Chairperson**
**Board of Studies (CSE)**

**MTCS11073** **Natural Language Processing Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- To Develop and explore the problems and solutions of NLP.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS11073.1 | Show sensitivity to linguistic phenomena and model them with formal grammars. | 3 | 2 | - | L2,L3 |
| MTCS11073.2 | Manipulate probabilities, construct statistical models over strings and trees, and estimate parameters using supervised and unsupervised methods. | 3 | 3 | 2 | L2,L3 |
| MTCS11073.3 | Design, implement, and analyze NLP algorithms. | 3 | 3 | 3 | L2,L, L4 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**SYLLABUS**

**Developing the following programs using Python:**

Experiment 1: Tokenization.
Experiment 2: Stemming.
Experiment 3: Stop word removal (a, the, are, etc.).
Experiment 4: Word analysis (frequency distribution, concordance, collocations).
Experiment 5: Word generation using rule-based / probabilistic approaches.
Experiment 6: POS tagging using NLTK / Spacy.
Experiment 7: Morphological analysis of words.
Experiment 8: Chunking (shallow parsing).
Experiment 9: N-Grams model generation.
Experiment 10: N-Gram smoothing techniques.

**TEXT BOOKS:**

1. Multilingual natural Language Processing Applications: From Theory to Practice – Daniel M. Bikel and Imed Zitouni, Pearson Publication
2. Natural Language Processing and Information Retrieval: Tanvier Siddiqui, U.S. Tiwary

**REFERENCES:**

**1.** Speech and Natural Language Processing - Daniel Jurafsky & James H Martin, Pearson Publications

**Chairperson**
**Board of Studies (CSE)**

**MTCS11074** **Advanced UNIX Programming Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- To Develop and explore the problems and solutions of NLP.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS11074.1 | Develop shell scripts in order to perform shell programming. | 3 | 2 | - | L2,L3 |
| MTCS11074.2 | Demonstrate the UNIX file system and utilities. | 3 | 3 | 2 | L2,L3 |
| MTCS11074.3 | Apply UNIX system calls to implement process and file management. | 3 | 3 | 3 | L2,L3 ,L4 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**SYLLABUS**

**List of Experiments**

Experiment 1: Practice session on UNIX commands and vi editor.

Experiment 2: Write a shell script to print the factorial of first *n* natural numbers.

Experiment 3: Write a shell script to generate a multiplication table of the given number.

Experiment 4: Write a shell script to list the files in the current directory to which the user has read, write and execute permissions.

Experiment 5: Write a shell script to compare two strings by reading strings from the command line.

Experiment 6: Write a shell script to read a username & find whether the user is currently logged in or not.

Experiment 7: Write shell scripts to find the length of a given string and to extract a substring from a given string.

Experiment 8: Write a shell script that counts the number of lines and words present in a given file.

Experiment 9: Write a shell script that displays the list of all files in the given directory.

Experiment 10: Write a shell script that copies multiple files to a directory.

Experiment 11: Write a shell script (small calculator) that adds, subtracts, multiplies, and divides the given two integers. Provide options: add (-a), subtract (-s), multiply (-m), quotient (-c), remainder (-r).

Experiment 12: Write a C program that illustrates the use of opendir, readdir, and closedir APIs.

Experiment 13: Write a C program that takes one or more file/directory names as command line input and reports: file type, number of links, time of last access, and permissions.

Experiment 14: Write a C program that illustrates the creation of a child process using the fork() system call.

Experiment 15: Write a C program to demonstrate inter-process communication using pipes/shared memory.

**Text Books**

1. Sumitabha Das – *UNIX Concepts and Applications*, Tata McGraw-Hill.
2. Maurice J. Bach – *The Design of the UNIX Operating System*, Prentice-Hall.

**References**

1. W. Richard Stevens – *Advanced Programming in the UNIX Environment*, Pearson.
2. Brian W. Kernighan, Rob Pike – *The UNIX Programming Environment*, Pearson.
3. M. J. Rochkind – *Advanced UNIX Programming*, Addison Wesley.

**Chairperson**
**Board of Studies (CSE)**

**MTCS1201** **Advanced Algorithms** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. Introduce students to the advanced methods of designing and analyzing algorithms.
2. The student should be able to choose appropriate algorithms and use it for a specific problem.
3. To familiarize students with basic paradigms and data structures used to solve advanced algorithmic problems.
4. Students should be able to understand different classes of problems concerning their computation difficulties.
5. To introduce the students to recent developments in the area of algorithmic design.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS1201.1 | Analyze the complexity/performance of different algorithms. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS1201.2 | Determine the appropriate data structure for solving a particular set of problems. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS1201.3 | Categorize different problems in various classes according to their complexity. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L3, L4 |
| MTCS1201.4 | Apply paradigms like greedy, divide & conquer, dynamic programming, and network flows to algorithmic problems. | 3 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | L4, L5 |
| MTCS1201.5 | Evaluate and adapt recent trends in searching, sorting, and advanced data structures. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT I: Sorting & Graph Basics** **(15 Hours)**

Review of sorting algorithms, Topological sorting.Graph: Definitions and elementary algorithms: Shortest path by BFS, shortest path in edge-weighted graphs (Dijkstra's).Depth-first search, computation of strongly connected components.Correctness proof of algorithms and time/space analysis. Amortized analysis examples. **CO's – CO1**

**Self-Learning Topics:** Advanced sorting techniques (Radix sort, Counting sort).

**UNIT II: Matroids & Graph Matching** **(10 Hours)**

Introduction to greedy paradigm. Algorithm to compute maximum weight maximal independent set. Application to Minimum Spanning Tree (MST). Graph Matching:

Algorithm to compute maximum matching, characterization of maximum matching by augmenting paths. Edmond's Blossom algorithm. **CO's – CO2**

**Self-Learning Topics:** Applications of greedy paradigm in real-world problems.

**UNIT III: Flow Networks & Matrix Computations (15 Hours)**

Flow Networks: Maxflow-mincut theorem, Ford-Fulkerson method, Edmond-Karp algorithm. Matrix Computations: Strassen's algorithm, Divide and Conquer paradigm. Inverse of a triangular matrix, relations between complexities of basic matrix operations, LUP-decomposition.

**Self-Learning Topics**: Push-relabel algorithm, recent fast matrix multiplication techniques.

**CO's-CO3**

**UNIT IV: Dynamic Programming, Modular Arithmetic & FFT (15 Hours)**

Shortest Path in Graphs: Floyd-Warshall algorithm, examples of dynamic programming. Modular Representation: Chinese Remainder Theorem, base-representation and modulo-representation conversions. Extension to polynomials and interpolation problem. Discrete Fourier Transform (DFT): In complex field and modulo ring. Fast Fourier Transform (FFT) algorithm, Schönhage-Strassen integer multiplication. **CO's – CO4**

**Self-Learning Topics:** Applications of FFT in polynomial multiplication, cryptography.

**UNIT V: Linear Programming & NP-Completeness (10 Hours)**

Linear Programming: Feasibility region geometry, Simplex algorithm.NP-completeness: Examples, proof of NP-hardness and NP-completeness. Recent Trends: Problem-solving paradigms using new searching and sorting techniques with recently proposed data structures. **CO's – CO5**

**Self-Learning Topics:** Approximation algorithms, randomized algorithms, parameterized complexity.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**Reference Books:**

1. Cormen, Leiserson, Rivest, Stein, *Introduction to Algorithms*.
2. Aho, Hopcroft, Ullman, *The Design and Analysis of Computer Algorithms*.
3. Kleinberg and Tardos, *Algorithm Design.*

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Advanced Data Structures and Algorithms
2. Coursera – Algorithms Specialization (Stanford)
3. edX – Advanced Algorithms and Complexity (UC San Diego)
4. MIT OCW – Advanced Algorithms

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define topological sorting.
2. What is amortized analysis?
3. Define matroid.
4. State max-flow min-cut theorem.
5. Define Strassen's algorithm.
6. What is Floyd-Warshall algorithm?
7. Define Chinese Remainder Theorem.
8. What is FFT?
9. Define NP-complete problem.
10. State simplex algorithm.

**L2: Understand**

1. Explain BFS-based shortest path algorithm.
2. Describe strongly connected components.
3. Explain greedy paradigm with example.
4. Discuss Edmond's Blossom algorithm.
5. Explain Ford-Fulkerson method.
6. Describe Strassen's algorithm for matrix multiplication.
7. Discuss Floyd-Warshall dynamic programming approach.
8. Explain modular representation of integers.
9. Explain FFT algorithm in brief.
10. Discuss proof of NP-hardness.

**L3: Apply**

1. Apply Dijkstra's algorithm on a weighted graph.
2. Apply greedy algorithm to MST.
3. Implement augmenting path algorithm for matching.
4. Apply Ford-Fulkerson algorithm on a network.
5. Implement Strassen's multiplication for 2x2 matrices.
6. Apply Chinese Remainder Theorem with example.
7. Implement FFT for polynomial multiplication.
8. Apply Simplex algorithm to a linear programming problem.
9. Apply approximation algorithm to NP-complete problem.
10. Use amortized analysis on dynamic array operations.

**L4: Analyze**

1. Compare BFS and Dijkstra's algorithm.
2. Analyze correctness proof of strongly connected components algorithm.
3. Differentiate matroid greedy solution vs. non-matroid problems.
4. Analyze Edmond-Karp's algorithm complexity.
5. Compare Strassen's algorithm and standard matrix multiplication.
6. Analyze polynomial interpolation using modular arithmetic.
7. Compare DFT and FFT.
8. Differentiate NP-hard and NP-complete problems.
9. Analyze simplex algorithm performance.
10. Compare modern sorting techniques with classical algorithms.

**L5: Evaluate**

1. Evaluate time complexity of Dijkstra's algorithm.
2. Assess efficiency of Blossom algorithm.
3. Evaluate network flow algorithms.
4. Critically evaluate divide and conquer paradigm in matrix computations.
5. Evaluate importance of Floyd-Warshall algorithm in dynamic programming.
6. Compare FFT vs. naive DFT.
7. Evaluate NP-complete problem classifications.
8. Assess role of approximation algorithms.
9. Evaluate modern data structures for problem-solving.
10. Compare recent searching techniques vs. traditional ones.

**L6: Create**

1. Design a new greedy algorithm for a scheduling problem.
2. Create a graph matching algorithm for bipartite graphs.
3. Develop a modified Ford-Fulkerson implementation.
4. Create a program for Strassen's multiplication.
5. Design a dynamic programming solution for knapsack.
6. Create FFT-based integer multiplication algorithm.
7. Develop a polynomial interpolation solver.
8. Construct a case study on NP-complete problems.
9. Design approximation algorithm for vertex cover.
10. Create a new hybrid sorting algorithm.

**Chairperson**
**Board of Studies (CSE)**

**MTCS1202** **Advanced Computer Architecture** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To impart the concepts and principles of parallel and advanced computer architectures.
2. To develop the design techniques of Scalable and multithreaded Architectures.
3. To Apply the concepts and techniques of parallel and advanced computer architectures to design modern computer systems

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS1202.1 | Understand computational models and advanced computer architectures. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS1202.2 | Analyze concepts of parallel computer models, multiprocessors, and SIMD systems. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS1202.3 | Apply concepts of scalable architectures, pipelining, and superscalar processors. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L3, L4 |
| MTCS1202.4 | Evaluate multiprocessor interconnects, cache coherence, synchronization, and message-passing mechanisms. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L4, L5 |
| MTCS1202.5 | Design modern high-performance architectures using vector and SIMD processing. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**UNIT – I** **(15 Hours)**

Theory of Parallelism, Parallel computer models, The State of Computing, Multiprocessors and Multicomputers, Multivector and SIMD Computers, PRAM and VLSI models, Architectural development tracks, Program and network properties, Conditions of parallelism, Program partitioning and Scheduling, Program flow Mechanisms, System interconnect Architectures.

**Self-Learning Topics:** Flynn's taxonomy of parallelism, Real-world case studies of PRAM models. **CO's – CO1.**

**UNIT – II** **(10 Hours)**

Principles of Scalable performance, Performance metrics and measures, Parallel Processing applications, Speed up performance laws, Scalability Analysis and Approaches, Hardware Technologies, Processes and Memory Hierarchy, Advanced Processor Technology, Superscalar and Vector Processors **CO's – CO2**

**Self-Learning Topics:** Amdahl's Law vs Gustafson's Law in scalability analysis.

**UNIT – III** **(15 Hours)**

Shared-Memory Organizations, Sequential and weak consistency models, Pipelining and superscalar techniques, Linear Pipeline Processors, Non-Linear Pipeline Processors, Instruction Pipeline design, Arithmetic pipeline design, superscalar pipeline design. **CO's – CO3**

**Self-Learning Topics:** Hazards in pipelining, Modern superscalar architectures (Intel/AMD).

**UNIT – IV** **(10 Hours)**

Parallel and Scalable Architectures, Multiprocessors and Multicomputers, Multiprocessor system interconnects, cache coherence and synchronization mechanism, Three Generations of Multicomputers, Message-passing Mechanisms, Multivetor and SIMD computers. **CO's – CO4**

**Self-Learning Topics:** Directory-based cache coherence, Case studies on MPI/OpenMP.

**UNIT – V** **(10 Hours)**

Vector processing principles, including vectorization of computations, pipelining, and performance considerations. Multivector multiprocessors – architecture, design strategies, and applications for high-performance computing. Compound vector processing techniques for handling complex operations and multiple data streams efficiently. SIMD (Single Instruction, Multiple Data) computer organizations, their structure, operation, and role in parallel computation. Study of advanced SIMD-based systems such as the Connection Machine CM-5, its architecture, processing model, interconnection networks, and applications in scientific computing and large-scale data processing. **CO's – CO5**

**Self-Learning Topics:** GPUs as modern SIMD processors, CUDA programming.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOK:**

1. Advanced Computer Architecture, Kai Hwang, 2nd Edition, Tata McGraw Hill Publishers.

**REFERENCES:**

1. Computer Architecture, J.L. Hennessy and D.A. Patterson, 4th Edition, ELSEVIER.
2. Advanced Computer Architectures, S.G.Shiva, Special Indian edition, CRC, Taylor &Francis.
3. Introduction to High Performance Computing for Scientists and Engineers, G. Hager and G. Wellein, CRC Press, Taylor & Francis Group.
4. Advanced Computer Architecture, D. Sima, T. Fountain, P. Kacsuk, Pearson education.
5. Computer Architecture, B. Parhami, Oxford Univ. Press.

**Online Learning Resources/Virtual Labs:**

1. NPTEL – Advanced Computer Architecture
2. Coursera – Computer Architecture Specialization
3. edX – High Performance Computing
4. MIT OCW – Parallel Computing and Architecture

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|

| | | |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define parallel computer models.
2. What is PRAM model?
3. Define SIMD architecture.
4. What is pipelining?
5. Define cache coherence.
6. What is scalability in performance?
7. Define vector processor.
8. What is interconnection network?
9. Define weak consistency model.
10. State Amdahl's Law.

**L2: Understand**

1. Explain the state of computing in parallel systems.
2. Discuss multiprocessors vs. multicomputers.
3. Explain performance metrics in scalable architectures.
4. Describe superscalar processor features.
5. Discuss pipeline hazards with examples.
6. Explain shared memory organization.
7. Describe message-passing mechanism.
8. Explain vector processing principles.
9. Discuss the connection machine CM-5.
10. Explain Gustafson's Law.

**L3: Apply**

1. Apply BFS for parallel graph traversal.
2. Implement speedup performance law for a problem.
3. Apply pipelining to an arithmetic computation.
4. Demonstrate cache coherence problem with example.
5. Apply vector processing to matrix multiplication.
6. Use superscalar techniques for instruction scheduling.
7. Apply program partitioning techniques.
8. Demonstrate MPI message passing.
9. Apply SIMD to image processing.

10. Implement scheduling for multiprocessors.

**L4: Analyze**

1. Compare PRAM and VLSI models.
2. Analyze performance using Amdahl's Law vs Gustafson's Law.
3. Compare linear vs. non-linear pipelines.
4. Analyze synchronization methods in multiprocessors.
5. Differentiate vector and scalar processing.
6. Compare different cache coherence mechanisms.
7. Analyze scalability approaches.
8. Compare superscalar vs. VLIW processors.
9. Analyze multiprocessor interconnect mechanisms.
10. Differentiate SIMD and multivector systems.

**L5: Evaluate**

1. Evaluate performance of multiprocessors vs multicomputers.
2. Assess benefits of superscalar processors.
3. Critically evaluate message-passing models.
4. Evaluate the effectiveness of vector processing.
5. Compare cache coherence protocols.
6. Evaluate GPU architecture as SIMD processor.
7. Assess pipelining efficiency in instruction design.
8. Critically evaluate three generations of multicomputers.
9. Evaluate scalability analysis methods.
10. Compare parallel processing applications.

**L6: Create**

1. Design a pipeline processor for matrix operations.
2. Create a scheduling mechanism for multiprocessors.
3. Develop a scalable architecture for HPC applications.
4. Construct a case study on SIMD implementation.
5. Design cache coherence protocol.
6. Build a parallel FFT using SIMD.
7. Develop a Superscalar pipeline simulation.
8. Create an interconnection network design.
9. Design a multivector multiprocessor.
10. Create a high-performance architecture using hybrid parallel models.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12031** **ENTERPRISE CLOUD CONCEPTS** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Knowledge on significance of cloud computing and its fundamental concepts and models.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12031.1 | Understand the fundamental importance of cloud computing concepts and architectures. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS12031.2 | Illustrate cloud security fundamentals and enabling technologies. | 3 | 3 | 2 | 2 | | 2 | 3 | 2 | L2, L3 |
| MTCS12031.3 | Analyze various cloud computing mechanisms and architectures. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L3, L4 |
| MTCS12031.4 | Evaluate cloud-inspired enterprise transformations and IT strategies. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L4, L5 |
| MTCS12031.5 | Develop strategies for transitioning to cloud-centric enterprise models. | 3 | 3 | 3 | 2 | 1 | 3 | 3 | 3 | L5, L6 |

## SYLLABUS

**Unit – I** **(15 Hours)**

**Understanding Cloud Computing:**

Origins and influences, Basic Concepts and Terminology, Goals and Benefits, Risks and Challenges.

**Fundamental Concepts and Models:**

Roles and Boundaries, Cloud Characteristics, Cloud Delivery Models, Cloud Deployment Models.

**Self-Learning Topics:** History of cloud computing adoption, Real-world examples of cloud delivery models. **CO's – CO1**

**Unit – II** **(10 Hours)**

**Cloud-Enabling Technology:**

Broadband Networks and Internet Architecture, Data Center Technology, Virtualization Technology CLOUD COMPUTING MECHANISMS:

**Cloud Infrastructure Mechanisms:** Logical Network Perimeter, Virtual Server, Cloud Storage Device, Cloud Usage Monitor, Resource Replication **CO's – CO2**

**Self-Learning Topics:** Role of virtualization in cloud environments, Modern hypervisors (VMware, KVM, Hyper-V).

**Unit – III** **(15 Hours)**

**Cloud Management Mechanisms**: Remote Administration System, Resource Management System, SLA Management System, Billing Management System, Case Study Example

**Cloud Computing Architecture**

**Fundamental Cloud Architectures:** Workload Distribution Architecture, Resource Pooling Architecture, Dynamic Scalability Architecture, Elastic Resource Capacity Architecture, Service Load Balancing Architecture, Cloud Bursting Architecture, Elastic Disk Provisioning Architecture, Redundant Storage Architecture, Case Study Example **CO's – CO3**

**Self-Learning Topics:** Containerization (Docker, Kubernetes) as modern cloud management mechanism.

**Unit – IV** **(10 Hours)**

**Cloud-Enabled Smart Enterprises**

Introduction, Revisiting the Enterprise Journey, Service-Oriented Enterprises, Cloud Enterprises, Smart Enterprises, The Enabling Mechanisms of Smart Enterprises

**Cloud-Inspired Enterprise Transformations**

Introduction, The Cloud Scheme for Enterprise Success, Elucidating the Evolving Cloud Idea, Implications of the Cloud on Enterprise Strategy, Establishing a Cloud-Incorporated Business Strategy **CO's – CO4**

**Self-Learning Topics:** Case study of cloud adoption in global enterprises (Amazon, Netflix).

**UNIT-V Transitioning to Cloud-Centric Enterprises** **(10 Hours)**

The Tuning Methodology, Contract Management in the Cloud Cloud-Instigated IT Transformations Introduction, Explaining Cloud Infrastructures, A Briefing on Next-Generation Services, Service Infrastructures, Cloud Infrastructures, Cloud Infrastructure Solutions, Clouds for Business Continuity, The Relevance of Private Clouds, The Emergence of Enterprise Clouds

**Self-Learning Topics:** Multi-cloud strategies, Hybrid cloud adoption in enterprises**CO's –CO5**

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Erl Thomas, Puttini Ricardo, Mahmood Zaigham, Cloud Computing: Concepts, Technology & Architecture 1st Edition,
2. Pethuru Raj, Cloud Enterprise Architecture, CRC Press

**REFERENCE:**

1. James Bond, The Enterprise Cloud, O'Reilly Media, Inc.

**Online Learning Resources/Virtual Labs:**

1. **NPTEL – Cloud Computing by Prof. Rajkumar Buyya.**
2. **Coursera – Cloud Computing Specialization (University of Illinois).**
3. **edX – Cloud Computing MicroMasters (UMGC & USMx).**
4. **AWS Academy – Cloud Foundations & Solutions Architect Training.**

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |

| | | |
|---|---|---|
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define cloud computing and its basic terminology.
2. What are cloud delivery models?
3. Define virtualization in the context of cloud computing.
4. What are the risks of cloud adoption?
5. Define cloud infrastructure mechanisms.
6. What is SLA in cloud computing?
7. Define elastic resource capacity.
8. What is service load balancing?
9. Define cloud bursting.
10. What is private cloud?

**L2: Understand**

1. Explain goals and benefits of cloud computing.
2. Discuss various cloud deployment models with examples.
3. Explain virtualization technology in cloud.
4. Discuss resource replication mechanism.
5. Explain resource management system in cloud.
6. Describe dynamic scalability architecture.
7. Explain smart enterprises enabled by cloud.
8. Discuss cloud-incorporated business strategy.
9. Explain cloud infrastructures for business continuity.
10. Describe relevance of enterprise private clouds.

**L3: Apply**

1. Apply cloud delivery models to e-commerce applications.
2. Demonstrate usage of cloud storage devices in enterprise solutions.
3. Apply billing management system to a SaaS product.
4. Use workload distribution architecture for balancing requests in cloud.
5. Apply containerization (Docker/Kubernetes) in cloud management.
6. Implement elasticity in cloud storage for data-intensive applications.
7. Demonstrate cloud bursting with hybrid deployment.
8. Apply operational cloud strategies to smart enterprises.
9. Use SLA management system in a case study.
10. Apply cloud-based disaster recovery for business continuity.

**L4: Analyze**

1. Compare cloud deployment models (public, private, hybrid).
2. Analyze risks and challenges of cloud adoption.
3. Compare virtualization and containerization approaches.
4. Analyze service load balancing architectures.
5. Compare SLA vs billing management systems.
6. Analyze transition strategy from traditional IT to cloud-centric enterprise.
7. Differentiate resource pooling and workload distribution.
8. Analyze implications of cloud on enterprise strategy.
9. Compare smart enterprises vs service-oriented enterprises.
10. Analyze security issues in cloud infrastructures.

**L5: Evaluate**

1. Evaluate benefits and limitations of cloud scalability.
2. Assess efficiency of resource replication in cloud.
3. Evaluate business impact of SLA management.
4. Critically evaluate cloud bursting strategy.
5. Evaluate operational profiles of smart enterprises.
6. Compare hybrid cloud vs multi-cloud adoption.
7. Evaluate elasticity in dynamic workloads.
8. Assess cloud security mechanisms.
9. Evaluate effectiveness of contract management in cloud.
10. Compare enterprise cloud adoption case studies.

**L6: Create**

1. Design a cloud deployment model for an online learning platform.
2. Create SLA and billing policies for SaaS application.
3. Develop elastic architecture for e-commerce website.
4. Design a hybrid cloud adoption strategy.
5. Create a smart enterprise model using cloud mechanisms.
6. Design cloud-based disaster recovery system.
7. Build a resource management system for cloud operations.
8. Develop operational profiles for cloud-enabled enterprises.
9. Create case study of a cloud-inspired enterprise transformation.
10. Develop a cloud-centric business continuity model.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12032** **CYBER SECURITY** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To understand various types of cyber-attacks and cyber-crimes
2. To learn threats and risks within context of the cyber security
3. To have an overview of the cyber laws & concepts of cyber forensics
4. To study the defensive techniques against these attacks

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12032.1 | Analyze and evaluate the cyber security needs of an organization. | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | L3, L4 |
| MTCS12032.2 | Understand cyber security regulations and roles of international law. | 3 | 2 | 1 | 2 | | 2 | 2 | 2 | L1, L2 |
| MTCS12032.3 | Design and develop a security architecture for an organization. | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | L4, L5 |
| MTCS12032.4 | Understand fundamental concepts of data privacy and privacy attacks. | 3 | 2 | 3 | 2 | 2 | 2 | 3 | 3 | L2, L3, L5 |

**SYLLABUS**

**UNIT -I** **(15 Hours)**

Introduction to Cyber Security: Basic Cyber Security Concepts, layers of security, Vulnerability, threat, Harmful acts, Internet Governance – Challenges and Constraints, Computer Criminals, CIA Triad, Assets and Threat, motive of attackers, active attacks, passive attacks, Software attacks, hardware attacks, Cyber Threats-Cyber Warfare, Cyber Crime, Cyber terrorism, Cyber Espionage, etc., Comprehensive Cyber Security Policy. **CO's – CO1**

**Self-Learning Topics:** Evolution of cyber security, Case studies on recent global cyber-attacks.

**UNIT – II** **(10 Hours)**

Cyberspace and the Law & Cyber Forensics: Introduction, Cyber Security Regulations, Roles of International Law. The INDIAN Cyberspace, National Cyber Security Policy.

Introduction, Historical background of Cyber forensics, Digital Forensics Science, The Need for Computer Forensics, Cyber Forensics and Digital evidence, Forensics Analysis of Email, Digital Forensics Lifecycle, Forensics Investigation, Challenges **CO's-CO2**

**Self-Learning Topics:** Recent Indian cyber laws, Global conventions (e.g., Budapest Convention).

**UNIT – III** **(15 Hours)**

Cybercrime: Mobile and Wireless Devices: Introduction, Proliferation of Mobile and Wireless Devices, Trends in Mobility, Credit card Frauds in Mobile and Wireless Computing Era, Security Challenges Posed by Mobile Devices, Registry Settings for Mobile Devices, Authentication service Security, Attacks on Mobile/Cell Phones, Organizational security Policies and Measures in Mobile Computing Era, Laptops. **CO's – CO1, CO3**

**Self-Learning Topics:** Mobile malware case studies, Android/iOS security models.

**UNIT- IV** **(10 Hours)**

Cyber Security: Organizational Implications: Introduction, cost of cybercrimes and IPR issues, web threats for organizations, security and privacy implications, social media marketing: security risks and perils for organizations, social computing and the associated challenges for organizations **CO's – CO1, CO3**

**Self-Learning Topics:** Social engineering attacks on enterprises, Insider threats.

**UNIT – V** **(10 Hours)**

Privacy Issues: Basic Data Privacy Concepts: Fundamental Concepts, Data Privacy Attacks, Data linking and profiling, privacy policies and their specifications, privacy policy languages, privacy in different domains- medical, financial, etc

**Cybercrime: Examples and Mini-Cases**

Examples: Official Website of Maharashtra Government Hacked, Indian Banks Lose Millions of Rupees, Parliament Attack, Pune City Police Bust Nigerian Racket, e-mail spoofing instances. Mini-Cases: The Indian Case of online Gambling, An Indian Case of Intellectual Property Crime, Financial Frauds in Cyber Domain. **CO's – CO4**

**Self-Learning Topics:** GDPR & Indian Data Privacy Laws.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Nina Godbole and Sunit Belpure, Cyber Security Understanding Cyber Crimes, Computer Forensics and Legal Perspectives, Wiley
2. B.B. Gupta, D.P. Agrawal, Haoxiang Wang, Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives, CRC Press, ISBN 9780815371335,2018.

**REFERENCES:**

1. Cyber Security Essentials, James Graham, Richard Howard and Ryan Otson, CRC Press.
2. Introduction to Cyber Security, Chwan-Hwa(john) Wu,J. David Irwin, CRC Press T&FGroup

**Online Learning Resources / Virtual Labs:**

1. NPTEL – Introduction to Cyber Security.
2. Coursera – Cybersecurity Specialization (University of Maryland).
3. edX – Cybersecurity Fundamentals (RIT).
4. Cybrary – Cyber Security Hands-on Labs.

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |

| | | |
|---|---|---|
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define vulnerability, threat, and harmful acts.
2. What is the CIA triad?
3. List types of cyber-attacks.
4. Define cyber espionage with an example.
5. What is digital forensics?
6. Define cybercrime in mobile devices.
7. What is IPR in cyber security?
8. Define data linking and profiling.
9. Mention examples of cyber terrorism.
10. What is a cyber security policy?

**L2: Understand**

1. Explain the challenges of Internet Governance.
2. Discuss roles of international law in cyberspace.
3. Explain the digital forensics lifecycle.
4. Discuss authentication security in mobile devices.
5. Explain web threats to organizations.
6. Explain data privacy policies in financial systems.
7. Describe risks of social media for organizations.
8. Discuss email spoofing as a cybercrime.
9. Explain active vs passive attacks.
10. Explain motive of attackers.

**L3: Apply**

1. Apply forensic analysis to trace email fraud.
2. Demonstrate mobile device security measures.
3. Apply privacy policy in healthcare data systems.
4. Apply SLA-based security for mobile enterprises.
5. Illustrate data linking in a financial fraud scenario.
6. Apply registry settings for mobile device protection.
7. Demonstrate laptop security in enterprise networks.
8. Apply profiling techniques in cybercrime detection.
9. Illustrate risk assessment for social computing.
10. Apply digital forensics in online gambling fraud case.

**L4: Analyze**

1. Analyze cost implications of cybercrimes.

2. Compare active and passive cyber-attacks.
3. Analyze risks in social media marketing.
4. Differentiate between Indian and International cyber laws.
5. Analyze cyber forensics challenges.
6. Compare malware attacks in Android vs iOS.
7. Differentiate between software vs hardware attacks.
8. Analyze organizational risks of insider threats.
9. Compare privacy in medical vs financial domains.
10. Analyze Indian case studies of cybercrimes.

**L5: Evaluate**

1. Evaluate effectiveness of cyber security policies.
2. Critically assess Indian National Cyber Security Policy.
3. Evaluate effectiveness of authentication in mobile devices.
4. Assess privacy risks in data profiling.
5. Compare international vs Indian cybercrime regulation.
6. Evaluate forensic investigation methods.
7. Assess enterprise risks in cloud-based services.
8. Evaluate cyber security challenges in IPR.
9. Critically assess case study: Indian Banks losing millions.
10. Evaluate strategies against phishing and spoofing.

**L6: Create**

1. Develop a comprehensive cyber security policy for an enterprise.
2. Create an organizational cybercrime response plan.
3. Design a forensic investigation model for email spoofing.
4. Create a mobile device security framework.
5. Develop a risk management strategy for social media threats.
6. Create data privacy policy for healthcare system.
7. Design a training program for cyber awareness.
8. Develop a case study on financial fraud detection.
9. Create a strategy to mitigate insider threats.
10. Design a cyber forensics workflow for online gambling.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12033** **Parallel computing** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To introduce the foundations of parallel Computing
2. To learn various parallel computing architectures and programming models
3. To gain knowledge of writing efficient parallel programs

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12033.1 | Understand the concepts of parallel architectures and programming platforms. | 3 | 2 | 2 | 2 | | 2 | 3 | 2 | L1, L2 |
| MTCS12033.2 | Select appropriate data structures to efficiently model information in a problem. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L2, L3 |
| MTCS12033.3 | Develop efficient parallel algorithms for computational and real-world problems. | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | L3, L4 |
| MTCS12033.4 | Implement parallel code, test correctness, and analyze performance. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L4, L5 |

## SYLLABUS

**Unit I** **(15 Hours)**

Parallel Computing: Introduction, motivation, and scope – need for parallelism in modern computing, challenges, and applications. Parallel Programming Platforms – taxonomy of parallel computers, Flynn's classification (SISD, SIMD, MISD, MIMD), shared-memory and distributed-memory platforms, clusters, multicores, GPUs, and hybrid architectures. Basic Communication Operations – message passing and point-to-point communication, collective communication operations such as broadcast, scatter, gather, reduction, and synchronization techniques. Performance metrics for parallel systems – speedup, efficiency, scalability, and cost-effectiveness. Overview of parallel programming models – shared memory, message passing (MPI), data parallel, and hybrid models. **CO's – CO**

**Self-Learning Topics:** Flynn's Taxonomy of parallel architectures, Case studies of supercomputers.

**Unit II** **(10 Hours)**

Principles of parallel algorithm design including task decomposition, data decomposition, load balancing, mapping tasks to processors, and synchronization strategies. Design paradigms such as divide-and-conquer, pipelining, and parallel recursion. Analytical modeling of parallel programs – performance evaluation techniques, computation and communication cost analysis, speedup and efficiency metrics, Amdahl's Law and Gustafson's Law, modeling of parallel

overheads, scalability analysis, and predicting execution time for different parallel architectures. Techniques for estimating resource utilization, communication contention, and optimizing parallel program performance through careful algorithm design

**Self-Learning Topics:** Performance metrics (speedup, efficiency, scalability). **CO's-CO2**

**Unit III** **(15 Hours)**

Programming using the Message Passing Paradigm with MPI – concepts of message passing, point-to-point and collective communication, MPI library routines, synchronization, communication patterns, and examples of MPI programs for solving parallel computation problems. Programming Shared Address Space Platforms using PThreads – thread creation, termination, synchronization using mutexes and condition variables, handling shared data, avoiding race conditions and deadlocks, and implementing parallel algorithms on multi-core systems. Comparative analysis of message passing and shared memory approaches, challenges in debugging and performance tuning, and practical applications demonstrating efficient utilization of parallel architectures. **CO's – CO3**

**Self-Learning Topics:** Hybrid MPI+OpenMP programming, CUDA overview for GPUs.

**Unit IV** **(10 Hours)**

Dense matrix algorithms including parallel implementations of matrix-vector multiplication and matrix-matrix multiplication, strategies for data distribution, load balancing, communication minimization, and performance optimization on parallel architectures. Sorting algorithms – issues in parallel sorting, challenges in data partitioning, comparison-based and non-comparison-based methods, parallel implementations of bubble sort, quick sort, bucket sort, enumeration sort, and radix sort. Analysis of time complexity, communication overhead, and scalability for each sorting algorithm. Practical considerations for mapping algorithms to shared-memory and distributed-memory systems, and techniques to improve efficiency and minimize synchronization delays in parallel computation. **CO's – CO3, CO4**

**Self-Learning Topics:** Parallel merge sort and parallel external sorting.

**Unit V** **(10 Hours)**

Parallel graph algorithms including minimum spanning tree computation using Prim's algorithm, and single-source shortest path determination using Dijkstra's algorithm, with emphasis on parallelization strategies, data distribution, and synchronization. Parallel search algorithms including depth-first search (DFS) and breadth-first search (BFS), techniques for handling large graphs efficiently, load balancing, and minimizing communication overhead in parallel environments. Analysis of algorithm complexity, scalability, and performance on shared-memory and distributed-memory architectures. Applications in network analysis, scientific computing, and large-scale data processing, demonstrating the practical use of parallel graph and search algorithms. **CO's – CO3, CO4**

**Self-Learning Topics:** Parallel PageRank algorithm, GPU-based graph traversal.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOK:**

1. Introduction to Parallel Computing, Second Edition, Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta, Addison-Wesley, 2003, ISBN: 0201648652

**REFERENCES:**

1. Parallel Computing – Theory and Practice, Second Edition, Michaek J. Quinn, Tata McGraw-Hill Edition.
2. Parallel Computers – Architectures and Programming, V. Rajaraman, C. Siva Ram Murthy, PHI.

**Online Learning Resources / Virtual Labs:**

1. NPTEL – Parallel Computing (IIT Kanpur).
2. Coursera – Parallel Programming in C++ and MPI.
3. edX – High Performance Computing (HPC).
4. Virtual Labs – Parallel Computing & Distributed Systems Simulations.

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define parallel computing and its scope.
2. What are basic communication operations in parallel systems?
3. List the principles of parallel algorithm design.
4. Define MPI and its significance.
5. What is PThreads?
6. Define speedup and efficiency.
7. What is the purpose of radix sort?
8. Define Prim’s algorithm.
9. What is Dijkstra’s algorithm used for?
10. What are DFS and BFS in graph traversal?

**L2: Understand**

1. Explain Flynn's taxonomy of parallel architectures.
2. Describe analytical modeling of parallel programs.
3. Discuss the importance of MPI in distributed computing.
4. Explain issues in parallel sorting.
5. Compare bubble sort and quick sort in parallel execution.
6. Discuss parallel DFS vs BFS.
7. Explain scalability in parallel computing.
8. Discuss communication costs in parallel platforms.
9. Explain shared address space programming.
10. Explain the motivation behind parallel computing.

**L3: Apply**

1. Implement MPI program for matrix multiplication.
2. Write a PThreads program for parallel summation.
3. Apply parallel bucket sort on a dataset.
4. Implement DFS using parallel processing.
5. Apply MPI reduce operation for vector operations.
6. Write MPI code for broadcasting messages.
7. Implement BFS using multithreading.
8. Apply Prim's algorithm in parallel.
9. Write a parallel enumeration sort program.
10. Apply PThreads for sorting an array.

**L4: Analyze**

1. Analyze performance trade-offs between MPI and PThreads.
2. Compare sequential vs parallel matrix multiplication.
3. Analyze efficiency of quick sort vs radix sort in parallel.
4. Differentiate DFS and BFS traversal performance in parallel.
5. Analyze scalability challenges in MPI.
6. Compare analytical models for parallel programs.
7. Analyze communication overhead in distributed memory systems.
8. Differentiate parallel vs distributed computing.
9. Compare MST and SSSP algorithms.
10. Analyze load balancing in parallel sorting.

**L5: Evaluate**

1. Evaluate MPI as a parallel programming model.
2. Critically assess PThreads for shared memory systems.
3. Evaluate radix sort efficiency on large datasets.
4. Assess performance of BFS in graph traversal.
5. Evaluate the impact of communication costs on scalability.
6. Compare synchronous and asynchronous MPI communication.
7. Evaluate cache coherence in shared memory systems.
8. Assess speedup laws in parallel systems.
9. Critically analyze GPU-based parallel processing.
10. Evaluate case study of supercomputers.

**L6: Create**

1. Develop an MPI program for distributed matrix multiplication.
2. Create a hybrid MPI + OpenMP program.
3. Design a parallel sorting algorithm for large datasets.
4. Create a parallel graph traversal for PageRank.
5. Develop PThreads program for producer-consumer problem.
6. Create a custom analytical model for performance estimation.
7. Design a parallel shortest path algorithm.
8. Create a parallel system for transaction processing.
9. Develop parallel BFS for social network graph analysis.
10. Create CUDA-based implementation for matrix multiplication.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12034** **Large Language Models** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To introduce the foundations of transformer architectures and their evolution into LLMs.
2. To equip students with skills to train, fine-tune, and deploy LLMs for various tasks.
3. To explore ethical, legal, and societal implications of LLMs in real-world applications.
4. To expose students to state-of-the-art LLM frameworks, evaluation techniques, and research trends.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12034.1 | Understand the architecture and inner workings of transformer-based LLMs. | 3 | 2 | 1 | 1 | | 2 | 3 | 2 | L1, L2 |
| MTCS12034.2 | Apply prompt engineering and fine-tuning techniques for domain-specific tasks. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | L2, L3 |
| MTCS12034.3 | Evaluate LLM performance using standard metrics and benchmarks. | 3 | 3 | 3 | 2 | 2 | 3 | 3 | 3 | L3, L4 |
| MTCS12034.4 | Identify challenges in LLM training, deployment, and scaling. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L4, L5 |
| MTCS12034.5 | Analyze ethical, legal, and societal implications of LLM usage. | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | L4, L5 |

## SYLLABUS

**UNIT 1 – Foundations of Large Language Models** **(15 Hours)**

Introduction to LLMs: Definition, scope, and historical evolution from statistical NLP to transformers.The Transformer architecture: Attention mechanisms, self-attention, multi-head attention. Pretraining

objectives: Masked language modeling (MLM), Causal language modeling (CLM). Evolution of LLMs:

BERT, GPT series, T5, LLaMA, Mistral. **CO's – CO1**

**Self-Learning Topics:** Encoder-decoder architectures, positional encodings.

**UNIT 2 – Training and Fine-Tuning LLMs** **(10 Hours)**

Pretraining datasets and tokenization: BPE, SentencePiece, WordPiece. Fine-tuning approaches: Full fine-tuning, LoRA, adapters, instruction tuning. Domain adaptation and few-shot/zero-shot learning. Data augmentation for LLMs and prompt-based tuning. Tools & frameworks: LangChain, LlamaIndex, Hugging Face Transformers. **CO's – CO2**

**Self-Learning Topics:** Reinforcement Learning with Human Feedback (RLHF).

**UNIT 3 – Prompt Engineering and Applications** **(15 Hours)**

**Principles of prompt design:** Zero-shot, few-shot, and chain-of-thought prompting. System

prompts, role prompting, and context length optimization.

**Use cases:** Text generation, summarization, code generation, question answering, chatbots. Tools &frameworks: Lang Chain, Llama Index, Hugging Face Transformers. **CO's – CO2, CO3**

**Self-Learning Topics:** Advanced prompting techniques, agent-based LLM applications.

**UNIT 4 – Evaluation and Deployment of LLMs (10 Hours)**

**Evaluation metrics:** Perplexity, BLEU, ROUGE, METEOR, human evaluation.

**Benchmark datasets:** GLUE, SuperGLUE, HELM, BIG-bench.

**Deployment strategies**: API-based deployment, on-prem deployment, inference optimization. Scaling and latency considerations; quantization and pruning for LLMs. **CO's – CO3, CO4**

**Self-Learning Topics:** Distillation of large models for edge deployment.

**UNIT 5 – Ethics, Safety, and Future Directions (10 Hours)**

Bias, fairness, and toxicity in LLMs. Hallucination problem and mitigation techniques. Legal and regulatory issues: Copyright, data privacy, AI Act. Trends in LLM research: Multimodal LLMs, retrieval-augmented generation (RAG), open-source LLM ecosystems. **CO's – CO5**

**Self-Learning Topics:** AI governance frameworks and ethical guidelines.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Vaswani, A. et al. (2017) Attention Is All You Need – NIPS Conference Paper.
2. Lewis, P. et al. (2021) Language Models are Few-Shot Learners – OpenAI Research Paper.
3. Tunstall, L., von Werra, L., & Wolf, T. (2022) Natural Language Processing with Transformers- O'Reilly Media.

**REFERENCE BOOKS:**

1. Bommasani, R. et al. (2021) On the Opportunities and Risks of Foundation Models – Stanford CRFM.
2. Jurafsky, D., & Martin, J. H. (2023) Speech and Language Processing (3rd Edition draft) – Pearson.
3. Mollick, E., & Mollick, L. (2024) Co-Intelligence: Living and Working with AI – Little, Brown Spark.
4. Hugging Face Documentation – https://huggingface.co/docs/

**Online Learning Resources / Virtual Labs:**

1. Hugging Face Transformers Course.
2. Stanford CS324: Large Language Models.
3. DeepLearning.AI – Generative AI with LLMs (Coursera).
4. OpenAI API Documentation

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define Large Language Models (LLMs).
2. What is self-attention in transformers?
3. Define masked language modeling (MLM).
4. What is tokenization?
5. Name popular LLMs such as BERT, GPT, and T5.

**L2: Understand**

1. Explain the role of attention mechanisms in transformers.
2. Differentiate between full fine-tuning and LoRA.
3. Explain zero-shot vs few-shot prompting.
4. Discuss the importance of BLEU and ROUGE metrics.
5. Explain bias and fairness issues in LLMs.

**L3: Apply**

1. Implement a BPE tokenizer on a sample dataset.
2. Apply instruction tuning for a chatbot use case.
3. Use Hugging Face to fine-tune a summarization model.
4. Demonstrate zero-shot prompt for text classification.
5. Apply pruning to reduce LLM inference time.

**L4: Analyze**

1. Analyze the trade-offs between MLM and CLM pretraining.
2. Compare chain-of-thought prompting with zero-shot prompting.
3. Analyze evaluation metrics for LLM-generated text.
4. Compare API-based deployment with on-prem deployment.
5. Discuss the scalability challenges of LLMs.

**L5: Evaluate**

1. Evaluate the impact of hallucination in LLM applications.
2. Assess the effectiveness of quantization in inference optimization.
3. Critically evaluate legal challenges related to LLM usage.
4. Compare open-source vs closed-source LLM ecosystems.
5. Evaluate multimodal LLMs in terms of practical applications.

**L6: Create**

1. Develop a chatbot using prompt-based tuning.
2. Create an LLM pipeline with LangChain for retrieval-augmented generation.
3. Design an evaluation framework combining automatic and human metrics.
4. Develop a fine-tuned GPT model for domain-specific summarization.
5. Build a pipeline to detect and mitigate bias in LLM outputs.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12041** **Bioinformatics** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Knowledge on concepts of bioinformatics and biological motivations of sequence analysis

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12041.1 | Understand the central dogma & XML (Bio XML) for Bioinformatics. | 3 | 2 | 2 | 2 | 1 | | 3 | 2 | L1, L2 |
| MTCS12041.2 | Analyze Perl (Bioperl) for solving Bioinformatics problems. | 3 | 3 | 2 | 2 | 1 | 2 | 3 | 3 | L2, L3 |
| MTCS12041.3 | Illustrate database technology, architecture and its interfaces for biological data. | 3 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | L3, L4 |
| MTCS12041.4 | Understand sequence alignment algorithms and phylogenetic analysis methods. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L3, L4 |

## SYLLABUS

**UNIT -I :** **(12 Hrs)**

The Central Dogma & XML (Bio XML) for Bioinformatics: Watson's definition, information flow, from data to knowledge, Convergence, the organization of DNA, the organization of Proteins, Introduction, Differences between HTML and XML, fundamentals of XML, fundamentals of XML namespaces. Introduction to DTDs, Document type Declarations, Declaring elements, declaring attributes, working with entities XML Schemas, Essential Concepts, working with simple types, working with complex types, Basic namespaces issues.

**Self-Learning Topics:** XML in biomedical data exchange (Bio XML). **CO's – CO1**

**UNIT -II :** **(12 Hrs)**

Perl (Bioperl) for Bioinformatics: Representing sequence data, program to store a DNA sequence, concatenating DNA fragments, Transcription, Calculating the reverse complement in Perl, Proteins, files, reading proteins in files, Arrays, Flow control, finding motifs, counting Nucleotides, exploding strings into arrays, operating on strings, writing to files, subroutines and bugs. **CO's – CO2**

**Self-Learning Topics:** Using Bioperl modules for sequence alignment.

**UNIT -III :** **(10 Hrs)**

Databases: Flat file, Relational, object oriented databases, object Relational and Hypertext, Data life cycle, Database Technology, Database Architecture, Database Management Systems and Interfaces. **CO's – CO3**

**Self-Learning Topics:** NCBI, EMBL, and PDB bioinformatics databases.

**UNIT -IV :** **(12 Hrs)**

Sequence Alignment Algorithms: Biological motivations of sequence analysis, the models for

sequence analysis and their biological motivation, global alignment, local alignment, End free-space alignment and gap penalty, Sequence Analysis tools and techniques. **CO's – CO4**

**Self-Learning Topics:** BLAST, FASTA tools.

**UNIT -V : (14 Hrs)**

Phylogenetic Analysis: Introduction, methods of Phylogenetic analysis, distance methods, the neighbor- Joining (NJ) method, The Fitch/ Margoliash method, character-based methods, Other methods, Tree evaluation and problems in phylogenetic analysis, Clustering, Protein structure visualization and Protein structure prediction. **CO's – CO4**

**Self-Learning Topics:** Molecular Evolutionary Genetics Analysis (MEGA) tool.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. S.C. Rastogi, N. Mendiratta, "Bioinformatics Methods and Applications", CBS publications, 2004
2. James D. Tisdall, "Beginning Perl for Bioinformatics" O'Reilly media, 1st Edition, 2001

**REFERENCE BOOKS:**

1. D.R. Westhead, J.H. Parish, "Bioinformatics" Viva books private limited, New Delhi (2003)
2. Att Wood, "Bioinformatics" Pearson Education, 2004
3. Bryan Bergeron, M.D, "Bioinformatics Computing" Pearson Education, 2003

**Online Learning Resources / Virtual Labs:**

1. NCBI Online Bioinformatics Resources.
2. EMBL-EBI Training Portal.
3. Coursera – *Bioinformatics Specialization* (UC San Diego).
4. MIT OpenCourseWare – *Computational Biology*

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define the Central Dogma of molecular biology.
2. What is Bio XML?
3. List major bioinformatics databases.
4. Define global and local sequence alignment.
5. State the neighbor-joining method.

**L2: Understand**

1. Explain differences between HTML and XML.
2. Describe DNA transcription in Perl with an example.
3. Explain database architecture for bioinformatics.
4. Differentiate between gap penalties in sequence alignment.
5. Explain the concept of phylogenetic trees.

**L3: Apply**

1. Write a Perl program to count nucleotides in a DNA sequence.
2. Use Bioperl to find motifs in protein sequences.
3. Demonstrate usage of FASTA for sequence analysis.
4. Apply relational databases to store genomic data.
5. Construct a phylogenetic tree using NJ method.

**L4: Analyze**

1. Analyze XML schemas in representing biological datasets.
2. Compare local vs global alignment with examples.
3. Analyze flat file vs relational databases for bioinformatics.
4. Examine challenges in phylogenetic analysis.
5. Compare Fitch/Margoliash and Neighbor-Joining methods.

**L5: Evaluate**

1. Evaluate the role of Bioperl in bioinformatics research.
2. Critically evaluate different database technologies for biological data.
3. Assess the advantages of character-based phylogenetic methods.
4. Evaluate sequence alignment tools (BLAST vs FASTA).
5. Judge the accuracy of protein structure prediction techniques.

**L6: Create**

1. Create an XML schema for representing protein data.
2. Develop a Perl script for reverse complement of a DNA strand.
3. Design a relational schema for a bioinformatics database.
4. Create a pipeline integrating BLAST and phylogenetic analysis.
5. Build a visualization for protein structures using open-source tools.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12042** **Adhoc Sensor Networks** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

1. To understand the challenges of routing in ad-hoc and sensor networks
2. To understand various broadcast, mutlicast and geocasting protocols in ad hoc and sensor networks
3. To understand basics of Wireless sensors, and Lower Layer Issues and Upper Layer Issues of WSN

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12042.1 | Understand the concepts of sensor networks and applications. | 3 | 2 | 2 | 2 | 1 | | 3 | 2 | L1, L2 |
| MTCS12042.2 | Compare and analyze MAC and routing protocols for ad-hoc networks. | 3 | 3 | 3 | 2 | 2 | 2 | 3 | 2 | L2, L3 |
| MTCS12042.3 | Understand and evaluate transport protocols for sensor and ad-hoc networks. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L3, L4 |

**SYLLABUS**

**UNIT – I** **(12 Hrs)**

**Introduction to Ad Hoc Networks**

Characteristics of MANETs, Applications of MANETs and Challenges of MANETs. Routing in MANETs, Criteria for classification, Taxonomy of MANET routing algorithms, Topology-based routing algorithms-Proactive: DSDV, WRP; Reactive: DSR, AODV, TORA; Hybrid: ZRP; Position- based routing algorithms-Location Services-DREAM, Quorum-based, GLS; Forwarding Strategies,Greedy Packet, Restricted Directional Flooding-DREAM, LAR; Other routing algorithms-QoS Routing, CEDAR. **CO's – CO1, CO2**

**Self-Learning Topics:** Case study on AODV routing.

**UNIT – II** **(10 Hrs)**

**Data Transmission**

Broadcast Storm Problem, Rebroadcasting Schemes-Simple-flooding, Probability-based Methods, Area-based Methods, Neighbour Knowledge-based: SBA, Multipoint Relaying, AHBP. Multicasting: Tree-based: AMRIS, MAODV; Mesh-based: ODMRP, CAMP; Hybrid: AMRoute, MCEDAR. **CO's – CO2**

**Self-Learning Topics:** Performance comparison of multicast algorithms.

**UNIT – III** **(12 Hrs)**

Geocasting techniques for efficient message delivery in mobile ad hoc networks, principles of location-based multicast (LBM) for data transmission, and strategies for route creation using protocols such as GeoTORA and MGR. Overview of TCP operation over ad hoc networks, challenges posed by dynamic topologies, variable link quality, and mobility. Examination of TCP

behavior in MANETs, including issues like congestion misinterpretation and packet loss, and proposed solutions and enhancements for reliable transport, congestion control, and improved throughput in mobile ad hoc environments.

**Self-Learning Topics:** Impact of mobility on TCP performance. **CO's – CO2, CO3**

**UNIT – IV** **(12 Hrs)**

Basics of wireless sensor networks, their significance, and challenges in design and deployment. Applications of sensor networks across environmental monitoring, healthcare, industrial automation, and military domains. Classification of sensor networks based on topology, mobility, and application requirements. Architecture of sensor networks including sensor nodes, sink nodes, and gateways, highlighting energy constraints and communication requirements. Detailed study of lower layer issues including the physical layer – modulation, transmission, and energy-efficient design; MAC layer – access control, collision avoidance, and duty-cycling; link layer – error detection, reliability mechanisms, and medium access; routing layer – energy-aware routing, multi-hop communication, and protocols for efficient data delivery in sensor networks.

**CO's – CO1, CO2**

**Self-Learning Topics:** IEEE 802.15.4 standard for WSN.

**UNIT – V** **(14 Hrs)**

Upper layer issues of wireless sensor networks (WSNs) including transport layer mechanisms for reliability, congestion control, and efficient data delivery. High-level application layer support for data aggregation, query processing, event detection, and distributed decision-making. Strategies for adapting to the inherent dynamic nature of WSNs including node mobility, network topology changes, and energy constraints. Integration of WSN protocols across layers to ensure robust communication and fault tolerance. **CO's – CO3**

**Self-Learning Topics:** TinyOS and Contiki OS for WSN applications.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Ad Hoc and Sensor Networks – Theory and Applications, Carlos Corderio Dharma P.Aggarwal, World Scientific Publications, March 2006, ISBN – 981-256-681-3
2. Wireless Sensor Networks: An Information Processing Approach, Feng Zhao, Leonidas Guibas, Elsevier Science, ISBN – 978-1-55860-914-3 (Morgan Kauffman)

**REFERENCES:**

1. C. Siva Ram Murthy, B.S. ManojAd Hoc Wireless Networks: Architectures and Protocols
2. Taieb Znati Kazem Sohraby, Daniel Minoli, Wireless Sensor Networks: Technology, Protocols and Applications, Wiley.

**Online Learning Resources / Virtual Labs:**

1. NPTEL Course – *Ad Hoc and Sensor Networks*.
2. Coursera – *Wireless Sensor Networks for IoT*.
3. MIT OpenCourseWare – *Wireless Communications and Networking*.
4. IEEE Xplore tutorials on WSN and MANET routing.

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Levels**

**L1: Remember**

1. Define MANET.
2. List applications of ad hoc networks.
3. State the broadcast storm problem.
4. Define geocasting.
5. List classifications of sensor networks.

**L2: Understand**

1. Explain proactive vs reactive routing algorithms.
2. Describe the problem of TCP in MANETs.
3. Explain the architecture of WSN.
4. Differentiate between tree-based and mesh-based multicasting.
5. Explain DREAM routing protocol.

**L3: Apply**

1. Demonstrate how AODV works for dynamic routing.
2. Apply MPR scheme to reduce broadcast storm problem.
3. Implement LBM for geocasting in a case study.
4. Use probability-based rebroadcasting for energy efficiency.
5. Apply routing protocols for WSN in IoT applications.

**L4: Analyze**

1. Analyze the challenges of MANET routing compared to WSN routing.
2. Compare proactive, reactive, and hybrid routing strategies.
3. Analyze the limitations of TCP over MANET.
4. Compare MAC protocols in WSN.
5. Analyze hybrid multicast protocols with examples.

**L5: Evaluate**

1. Evaluate QoS routing in MANETs.
2. Assess the performance of multicasting algorithms.

3. Judge the suitability of GeoTORA in high-mobility environments.
4. Evaluate the scalability of routing algorithms in WSN.
5. Critically assess fault tolerance mechanisms in WSN.

**L6: Create**

1. Design a small ad-hoc network scenario using AODV.
2. Create a simulation model for broadcast storm problem.
3. Design a hybrid routing protocol combining reactive and proactive methods.
4. Build a testbed for evaluating WSN MAC layer protocols.
5. Create a topology for WSN applied to smart agriculture.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12043** **Robotic Process Automation** **3 0 0 3**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Aim of the course is to make learners familiar with the concepts of Robotic Process Automation.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PSO1 | PSO2 | DoK |
|---|---|---|---|---|---|---|---|---|---|---|
| MTCS12043.1 | Describe RPA, its applications, and bot creation methods. | 3 | 2 | 2 | 2 | 1 | | 3 | 2 | L1, L2 |
| MTCS12043.2 | Identify and explain Web Control Room and Client functionalities. | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | L2, L3 |
| MTCS12043.3 | Understand device handling, workload management, and API integration in RPA. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L3, L4 |
| MTCS12043.4 | Apply recorders, task editors, and automation commands to develop bots. | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | L3, L4 |
| MTCS12043.5 | Implement advanced RPA commands for integration, error handling, and reporting. | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | L4, L5 |

## SYLLABUS

**Unit I** **(10 Hrs)**

Introduction to Robotic Process Automation & Bot Creation Introduction to RPA and Use cases – Automation Anywhere Enterprise Platform – Advanced features and capabilities – Ways to create Bots **CO's – CO1**

**Self-Learning Topics:** RPA use cases in Banking, Insurance, and Healthcare.

**Unit II** **(12 Hrs)**

Web Control Room and Client Introduction - Features Panel - Dashboard (Home, Bots, Devices, Audit, Workload, Insights) - Features Panel – Activity (View Tasks in Progress and Scheduled Tasks) - Bots (View Bots Uploaded and Credentials) **CO's – CO2**

**Self-Learning Topics:** Compare Automation Anywhere and UiPath dashboards.

**Unit III** **(12 Hrs)**

Devices (View Development and Runtime Clients and Device Pools) - Workload (Queues and SLA Calculator) - Audit Log (View Activities Logged which are associated with Web CR) - Administration (Configure Settings, Users, Roles, License and Migration) - Demo of Exposed API's – Conclusion – Client introduction and Conclusion. **CO's – CO3**

**Self-Learning Topics:** Role-based access control in RPA tools.

**Unit IV** **(14 Hrs)**

Bot Creator Introduction – Recorders – Smart Recorders – Web Recorders – Screen Recorders - Task Editor – Variables - Command Library – Loop Command – Excel Command – Database Command - String Operation Command - XML Command **CO's – CO4**

**Self-Learning Topics:** Automation of repetitive Excel tasks using Bot Creator.

**Unit V** **(14 Hrs)**

Terminal Emulator Command - PDF Integration Command - FTP Command - PGP Command - Object Cloning Command - Error Handling Command - Manage Windows Control Command - Workflow Designer - Report Designer **CO's – CO5**

**Self-Learning Topics:** Error handling best practices in RPA projects.

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**TEXT BOOKS:**

1. Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool - UiPath: Create Software robots. with the leading RPA tool – UiPath Kindle Edition.

**REFERENCES:**

1. Robotic Process Automation A Complete Guide - 2020 Edition Kindle Edition.

**Online Learning Resources / Virtual Labs:**

1. UiPath Academy – Free online training for RPA.
2. Automation Anywhere University – Hands-on tutorials and certification.
3. Coursera – *Robotic Process Automation Specialization.*
4. EdX – *Enterprise Automation with RPA.*
5. NPTEL – *Intelligent Automation and RPA* modules**.**

**Internal Assessment Pattern**

| Cognitive Level | Internal Assessment #1(%) | Internal Assessment #2(%) |
|---|---|---|
| L1 | 35 | -- |
| L2 | 40 | -- |
| L3 | 25 | 20 |
| L4 | -- | 35 |
| L5 | -- | 35 |
| L6 | -- | 10 |
| **Total (%)** | **100** | **100** |

**Sample Short and Long Answers questions of Various Cognitive Level**

**L1: Remember**

1. Define RPA.
2. List three features of Automation Anywhere platform.
3. State the purpose of Web Control Room.

4. What is a bot?
5. Define Audit Log in RPA.

**L2: Understand**

1. Explain various ways to create bots.
2. Describe the Dashboard features in Web Control Room.
3. Differentiate between Smart, Web, and Screen Recorders.
4. Explain role of SLA calculator in workload management.
5. Describe importance of Object Cloning Command.

**L3: Apply**

1. Apply RPA to automate a simple login form.
2. Demonstrate how to schedule a task using Web Control Room.
3. Use loop command to process multiple records in Excel.
4. Apply PDF integration command to extract data.
5. Automate FTP file upload using RPA.

**L4: Analyze**

1. Analyze security challenges in RPA implementation.
2. Compare device management with and without device pools.
3. Analyze advantages of recorders in bot creation.
4. Compare different error handling techniques.
5. Analyze use cases of PGP command in RPA.

**L5: Evaluate**

1. Evaluate the performance of bots in workload queues.
2. Judge the reliability of terminal emulator command in real-time tasks.
3. Evaluate pros and cons of using exposed APIs in RPA.
4. Assess suitability of UiPath vs Automation Anywhere for enterprise use.
5. Critically evaluate the role of workflow designer in RPA.

**L6: Create**

1. Design a bot to extract invoice data from PDF and store in Excel.
2. Create a workflow using task editor for automating report generation.
3. Design a bot that integrates with FTP server to transfer files.
4. Build an RPA solution for automating HR employee onboarding.
5. Develop a complete RPA project that uses multiple commands for end-to-end automation.

**Chairperson**
**Board of Studies (CSE)**

**MTCS1205** **Advanced Algorithms Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- To impart knowledge in advanced algorithmic techniques.
- To develop skills in designing and analyzing algorithms for computational problems.
- To explore different algorithmic paradigms such as divide and conquer, greedy, dynamic programming, and graph algorithms.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS1205.1 | Analyze the performance of algorithms using theoretical and empirical methods. | 3 | 2 | - | L2,L3 |
| MTCS1205.2 | Apply advanced algorithmic strategies to solve computational problems. | 3 | 3 | 2 | L2, L3,L4 |
| MTCS1205.3 | Design efficient algorithms for real-time applications. | 3 | 4 | 3 | L3, L4 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**List of Experiments**

Experiment 1: Implement assignment problem using Brute Force method.

Experiment 2: Perform multiplication of long integers using Divide and Conquer method.

Experiment 3: Implement a solution for the Knapsack problem using the Greedy method.

Experiment 4: Implement Gaussian elimination method.

Experiment 5: Implement LU decomposition.

Experiment 6: Implement Warshall's algorithm for transitive closure.

Experiment 7: Implement the Rabin–Karp algorithm for pattern matching.

Experiment 8: Implement the Knuth–Morris–Pratt (KMP) algorithm for pattern matching.

Experiment 9: Implement the Horspool algorithm for string matching.

Experiment 10: Implement the Max-Flow problem using Ford–Fulkerson method.

**Text Book**

1. S. Sridhar – *Design and Analysis of Algorithms*, Oxford University Press.

**Reference Books:**

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein – *Introduction to Algorithms*, PHI Pvt. Ltd./ Pearson Education.

2. Ellis Horowitz, Sartaj Sahni, Rajasekharam – *Fundamentals of Computer Algorithms*, Universities Press.
3. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman – *Design and Analysis of Algorithms*, Pearson Education.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12061** **Enterprise Cloud Concepts Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- To provide knowledge on the significance of cloud computing and its fundamental concepts.
- To understand cloud service models and deployment models.
- To impart hands-on experience with virtualization, cloud platforms, and cloud applications.
- To explore cloud resource management, security, and big data processing in cloud environments.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS12061.1 | Understand the importance of cloud architecture and service models. | 3 | 2 | - | L2,L3 |
| MTCS12061.2 | Illustrate the fundamental concepts of cloud security and cloud platforms. | 3 | 3 | 2 | L2, L3,L4 |
| MTCS12061.3 | Analyze and implement various cloud computing mechanisms and services. | 3 | 4 | 3 | L3, L4 |
| MTCS12061.4 | Apply cloud computing techniques for application deployment and data processing. | 3 | 4 | 3 | L3, L4 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**List of Experiments**

Experiment 1: Install VirtualBox/VMware Workstation and configure different flavors of Linux/Windows OS on top of Windows 7/8.
Experiment 2: Install a C compiler in the virtual machine created using VirtualBox and execute simple programs.
Experiment 3: Install Google App Engine, create a “Hello World” app and simple web applications using Python/Java.
Experiment 4: Find a procedure to transfer files from one virtual machine to another.
Experiment 5: Launch a virtual machine using TryStack (Online OpenStack Demo Version).
Experiment 6: Install Hadoop Single Node Cluster and run simple applications like Word Count.

**E-Resources:**

1. https://www.iitk.ac.in/nt/faq/vbox.htm
2. https://www.google.com/urlsa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjqrNG0za73AhZt1YBHZ21DWEQFnoECAMQAQ&url=http%3A%2F%2Fwww.cs.columbia.edu

%2F~sedwards%2F classes%2F2015%2F1102-fall%2Flinuxvm.pdf&usg=AOvVaw3xZPuF5xVgk-AQnBRsTtHz

3. https://www.cloudsimtutorials.online/cloudsim/

4. https://edwardsamuel.wordpress.com/2014/10/25/tutorial-creating-openstack-instance-in-trystack/

5. https://www.edureka.co/blog/install-hadoop-single-node-hadoop-cluster

**Chairperson**
**Board of Studies (CSE)**

**MTCS12062** **Cyber Security Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- To get practical exposure of Cyber security threats.
- To learn about cyber forensic tools.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS12062.1 | Perform port scanning, footprinting, and identify cyber threats/attacks. | 3 | 2 | - | L2,L3 |
| MTCS12062.2 | Use forensic tools (Wireshark, Autopsy, Snort, FTK, | 3 | 3 | 2 | L2, L3,L4 |
| MTCS12062.3 | Use forensic tools (Wireshark, Autopsy, Snort, FTK, Network Miner) for monitoring, analysis, and investigation. | 3 | 3 | 3 | L3, L4 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**List of Experiments**

Experiment 1: Perform an experiment for port scanning with NMAP.

Experiment 2: Setup a honeypot and monitor the honeypot on the network.

Experiment 3: Install Jcrypt / Cryptool (or equivalent) and demonstrate Symmetric & Asymmetric crypto, Hash, and Digital/PKI signatures.

Experiment 4: Generate minimum 10 passwords of length 12 characters using OpenSSL.

Experiment 5: Implement footprinting – gathering target information using Dmitry-Dmagic and UAtester.

Experiment 6: Use sniffers (Wireshark) for monitoring network communication.

Experiment 7: Use Snort to perform real-time traffic analysis and packet logging.

Experiment 8: Perform email analysis using Autopsy tool.

Experiment 9: Perform Registry analysis and get boot time logging using Process Monitor tool.

Experiment 10: Perform File type detection using Autopsy tool.

Experiment 11: Perform Memory capture and analysis using FTK Imager tool.

Experiment 12: Perform Network analysis using Network Miner tool..

**Text Books**

1. Real Digital Forensics for Handheld Devices – E. P. Dorothy, Auerbach Publications, 2013.
2. Handbook of Digital Forensics and Investigation – E. Casey, Academic Press, 2010.

**Reference Books:**

1. The Basics of Digital Forensics: The Primer for Getting Started in Digital Forensics – J. Sammons, Syngress Publishing, 2012.
2. Malware Forensics Field Guide for Windows Systems: Digital Forensics Field Guides – C. H. Malin, E. Casey, J. M. Aquilina, Syngress, 2012.
3. The Best Damn Cybercrime and Digital Forensics Book Period – J. Wiles, A. Reyes, Syngress, 2007.

**Chairperson**
**Board of Studies (CSE)**

**MTCS12063** **Parallel computing Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Introduce the foundations of parallel computing.
- Learn various parallel computing architectures and programming models.
- Gain knowledge of writing efficient parallel programs.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS12063.1 | Understand the concepts of parallel architectures. | 3 | 2 | - | L2 |
| MTCS12063.2 | Select data structures that efficiently model problem information. | 3 | 3 | 2 | L3 |
| MTCS12063.3 | Develop efficient parallel algorithms for computational problems. | 3 | 3 | 3 | L3, L4 |
| MTCS12063.4 | Implement correct and efficient parallel code, and analyze performance. | 3 | 3 | 3 | L3, L4, L5 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**List of Experiments**

Experiment 1: Design a parallel program to implement Matrix-Vector and Matrix-Matrix Multiplication using MPI library.

Experiment 2: Design a parallel program to implement Bubble Sort using OpenMP and Pthread Programming Constructs.

Experiment 3: Design a parallel program to implement Quick Sort using OpenMP and Pthread Programming Constructs.

Experiment 4: Design a parallel program to implement Bucket Sort using OpenMP and Pthread Programming Constructs.

Experiment 5: Design a parallel program to implement Prim's Algorithm using OpenMP and Pthread Programming Constructs.

Experiment 6: Design a parallel program to implement DFS Algorithm using OpenMP and Pthread Programming Constructs.

Experiment 7: Design a parallel program to implement BFS Algorithm using OpenMP and Pthread

Programming Constructs.

Experiment 8: Design a parallel program to implement Dijkstra's Algorithm using MPI library.

**Text Books**

1. Michael J. Quinn – *Parallel Programming in C with MPI and OpenMP*, Tata McGraw-Hill.
2. Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar – *Introduction to Parallel Computing*, Pearson.

**Reference Books:**

1. Peter S. Pacheco – *Parallel Programming with MPI*, Morgan Kaufmann.
2. B. Wilkinson, M. Allen – *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Pearson.
3. Ian Foster – *Designing and Building Parallel Programs*, Addison Wesley**.**

**Chairperson**
**Board of Studies (CSE)**

**MTCS12064** **Large Language Models Lab** **0 0 4 2**

(Computer Science & Engineering)

**Course Objectives:**

The main objectives of the course are to

- Introduce students to the foundations of Large Language Models (LLMs).
- Explore fine-tuning, prompt engineering, and model adaptation techniques.
- Provide hands-on skills to apply LLMs for real-world tasks in NLP.
- Develop awareness of ethical and societal implications of LLM deployment.

**Course Outcomes:**

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

| Course Code | Course Outcomes (COs) | PO1 | PO2 | PO3 | DoK |
|---|---|---|---|---|---|
| MTCS12064.1 | Understand the architecture and working principles of large language models. | 3 | 2 | - | L2 |
| MTCS12064.2 | Apply prompt engineering and fine-tuning techniques for specific NLP tasks. | 3 | 3 | 2 | L3, L4 |
| MTCS12064.3 | Implement real-world applications using LLM APIs and open-source frameworks. | 3 | 3 | 3 | L3, L4 |
| MTCS12064.4 | Analyze ethical, societal, and performance aspects of LLM-based systems. | 3 | 3 | 3 | L4, L5 |

Board of Studies : Computer Science and Engineering

Approved in BOS No: 01, --, August, 2025

Approved in ACM No: 01

**List of Experiments**

Experiment 1: Introduction to transformer architectures – explore Hugging Face models.

Experiment 2: Implement text classification using a pre-trained LLM.

Experiment 3: Perform Named Entity Recognition (NER) using transformer-based models.

Experiment 4: Fine-tune a BERT model for sentiment analysis.

Experiment 5: Implement a question-answering system using GPT-based models.

Experiment 6: Develop a text summarization application using pre-trained LLMs.

Experiment 7: Build a conversational chatbot using an open-source LLM.

Experiment 8: Experiment with prompt engineering techniques for different NLP tasks.

Experiment 9: Evaluate bias and fairness issues in outputs from LLMs.

Experiment 10: Deploy an LLM-based application (e.g., sentiment analysis API) on cloud/edge platforms.

**Text Books**

1. Lewis Tunstall, Leandro von Werra, Thomas Wolf – *Natural Language Processing with Transformers*, O'Reilly.
2. Palash Goyal, Sumit Pandey, Karan Jain – *Deep Learning for Natural Language Processing*, Apress.

**Reference Books:**

1. Jacob Eisenstein – *Introduction to Natural Language Processing*, MIT Press.
2. Tom B. Brown et al. – *Language Models are Few-Shot Learners (GPT-3 paper)*, NeurIPS.
3. Ian Goodfellow, Yoshua Bengio, Aaron Courville – *Deep Learning*, MIT Press.

**Chairperson**
**Board of Studies (CSE)**