2nd Year of 2-Year
MASTER OF COMPUTER APPLICATIONS (MCA)
PROGRAMME
Postgraduate Curriculum Framework
under NEP 2020
(w.e.f. 2025)

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF MATHEMATICAL SCIENCES
UNIVERSITY OF DELHI
DELHI
110007

Master of Computer Application (MCA) Program Details

Affiliation

The proposed programme shall be governed by the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, Delhi-110007.

Programme Structure and Objectives

The MCA is a four-semester program spanning two years. It has two structures – Structure 1 **MCA with coursework and Structure 4 - MCA with** Academic/Industry Project**.**

First three semesters comprise of course - work common to both the structures. In structure 4, students are required to do an industry project with at least one supervisor from the department in the fourth semester,.

The Programme objectives of MCA with Coursework are to

- **E**quip the students with the **foundations of** computer science.
- **Enable the students to follow the career path of their choice by choosing** courses from a wide list of specialised **c**ourses with progression**.**
- **Prepare the students to take up a career in the highly competitive IT industry with Applications of Computer Science.**

The Programme objectives of MCA with Academic/Industry Project are to

- **E**quip the students with the **foundations of** computer science.
- **Enable the students to follow the career path of their choice by choosing** courses from a wide list of specialised **c**ourses with progression**.**
- Provide students **with hands-on experience on live projects**, fostering practical skills and enabling them to prepare technical reports.
- **Prepare the students to take up a career in the highly competitive IT industry with applications of computer science.**

Project: In Structure 4, each student shall carry out a project in the fourth semester of MCA with internship. The project may be carried out in an external organisation (academic institution/industry). In such a case, a supervisor shall be appointed from the external organisation. The project will be carried out under the overall supervision of one or more (jointly) teachers of the department. The project work will be evaluated jointly by the internal supervisor and an examiner to be appointed by the department in consultation with the internal supervisor

The project evaluation shall be as follows:

- Mid-semester evaluation: 25% weight
- End-semester evaluation
    - Dissertation: 35% weight
    - Viva-voce: 40% weight

Justification for Structure IV : Since the inception of the MCA program in 1984, it has

had one full semester devoted to hands-on projects primarily in the industry. The format has led to a strong alumni base enjoying high positions and packages in the IT industry. Our alumni are currently placed in prestigious global companies like Apple, Facebook, Microsoft, Adobe, Paytm, Amazon to name a few. **The main attraction of our programs for the recruiters is its full time fourth semester Industry Internship.** During this period, students learn by doing and get acquainted with the environment of their work place while still earning their degree. Companies get a trained workforce when the students finally join the job.

With the emphasis of skill development and experiential learning being one of the emphasis of NEP, we have introduced Structure 4 which is different from Structure 1 only in Semester IV which allows the students to earn their 22 credits from an academic project/industry internship.

Semester III

Semester III (Common to both structures)					
	Number of core courses				**2**
Course Code	**Course Title**	**Credits in each core course**			
		Theory	**Tutorial**	**Practical**	**Total**
DSC301	Data Communication and Computer Networks	3	0	1	**4**
DSC302	Information Security	3	0	1	**4**
SBC301	Prompt Engineering	0	0	2	**2**
	Total credits in core course				10
	Number of DSE/GE courses				3**
	DSE 5	3	0	1	**4**
	DSE 6	3	0	1	**4**
	DSE 7 / GE 3	3	0	1	**4**
	Total credits in GE/DSE				**12**
	Total credits in Semester III				**22**

****Select three DSEs or two DSEs and one GE**
Note: All DSEs are offered as GE, subject to the fulfilment of the prerequisites and availability of seats.

List of DSE for Semester III (common to both structures)		
Course Code	**Course Title**	**L-T-P**
DSE301	Link Prediction	3-0-1
DSE302	Recommender Systems	3-0-1
DSE303	Time Series Data Analysis	3-0-1
DSE304	Quantum Computing and Applications	3-0-1
DSE305	Digital Forensics	3-0-1
DSE306	Human-Computer Interaction	3-0-1
DSE307	Influence Maximization	3-0-1
DSE308	Deep Learning (only for structure 4)	3-0-1

Semester IV

Semester IV (Structure 1 Coursework)					
	Number of core courses				**2**
Course Code	**Course Title**	**Credits in each core course**			
		Theory	**Tutorial**	**Practical**	**Total**
DSC401	Software Engineering	3	0	1	**4**
DSC402	Deep Learning	3	0	1	**4**
SBC401	Communication Skills	0	0	2	**2**
	Total credits in core course				10
	Number of DSE /GE courses				3**
	DSE 8	3	0	1	**4**
	DSE 9	3	0	1	**4**
	DSE 10 / GE 4	3	0	1	**4**
	Total credits in GE/DSE				12
	Total credits in Semester IV				**22**

****Select three DSEs or two DSEs and one GE**
Note: All DSEs are offered as GE, subject to the fulfilment of the prerequisites and availability of seats.

List of DSE for Semester IV (Structure 1 - Coursework)		
Course Code	**Course Title**	**L-T-P**
DSE401	Computer Graphics	3-0-1
DSE402	Compiler Design	3-0-1
DSE403	Cloud Computing	3-0-1
DSE404	Theory of Computation	3-0-1
DSE405	Wireless and Mobile Communication	3-0-1
DSE406	Advance Classification Methods	3-0-1
DSE407	Incident Response and Threat Hunting	3-0-1

Semester IV (Structure 4 - Academic/Industry Project)					
	Number of core courses				**0**
Course Code	**Course Title**	**Credits in each core course**			
		Theory	**Tutorial**	**Practical**	**Total**
	Academic Project/Industry Internship	0	0	22	**22**
	Total credits in Semester IV				**22**

In **Structure 4 - MCA with** Academic/Industry Project, the following outcomes must be achieved by the end of the fourth semester in addition to the outcomes mentioned for Structure 1:

i) Completion of experimentation/fieldwork or similar tasks.

ii) Submission of the project report.

SEMESTER – III

DSC301: DATA COMMUNICATION AND COMPUTER NETWORKS [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSC301	4	3	0	1	Graduation	NIL

Course Objectives:

The course introduces the basic concepts of data communication techniques, computer networks, various protocols, and their applications. It also aims to develop skills for interworking between computer networks and switching components in telecommunication systems.

Course Learning Outcomes:

On completing this course, the student will be able to:

1. Understand the basics of computer networks, such as network types, topologies, layered architectures, and the specific functions of each layer in the OSI and TCP/IP models.
2. Analyze the flow and structure of network packets to understand data transmission across a communication channel.
3. Design/implement various protocols in a simulated networking environment.
4. Simulate error detection and correction techniques in a noisy communication channel to ensure reliable data transmission.
5. Develop foundational skills to avail network services efficiently and design new networking applications.

Syllabus:

Unit-I **(15 hours)**

Introduction: Data communications systems and their components, Computer networks, Types of connections, Local Area Networks, Wide Area Networks, History, standards, and applications of the Internet; Network Models and Principles of protocol layering, TCP/IP protocol suite, OSI model; Layers and their functions.

Physical Layer: Data and signals, periodic analog signals, Digital signals, Bit rate, Bit length, Transmission impairment, Noisy and noiseless channels; Digital Transmission – Digital to digital conversion, Line coding, Block coding, Scrambling; Analog Transmission – Digital to analog conversion, Amplitude shift keying, Frequency shift keying, Phase shift keying, Analog to analog conversion, Amplitude modulation, Frequency modulation, Phase modulation; Bandwidth Utilization – Multiplexing and Spectrum Spreading, Guided and Unguided Transmission Media, Circuit-switched, datagram and virtual-circuit networks.

Unit-II **(12 hours)**

Data Link Layer: Nodes and links, Sublayers of data link layer and their functions, Link layer addressing and Address Resolution Protocol (ARP); Error detection and correction, Block coding, Cyclic codes, Checksum; Data Link Control – connectionless and connection-oriented services, stop-and-wait protocol, piggybacking, point-to-point protocol; Media Access Control - Random access (ALOHA, CSMA/CD, CSMA/CA), controlled access (polling, token passing) and channelization protocols (FDMA, TDMA, CDMA); Wireless LAN, hubs, switches and routers; Virtual LAN.

Unit-III **(10 hours)**

Network Layer: Packetizing, Routing and forwarding, Circuit switching, Packet switching, Delay and throughput; IPv4 addresses, Forwarding of IP packets, Internet Protocol (IPv4), Classful and classless addressing, Datagram format, fragmentation, IPv6 address space, IPv6 packet format, Extension header, Transition from IPv4 to IPv6.

Unit-IV **(8 hours)**

Transport Layer: Connectionless and connection-oriented protocols; Stop-and-Wait, Go-back-N, and Selective-Repeat protocols; Port numbers, User Datagram Protocol (UDP), User Datagram format, UDP applications; Transmission Control Protocol (TCP), TCP segment format, connection establishment, data transfer, and termination; Data flow and flow control.

Application Layer: World Wide Web (WWW), File transfer protocol, Electronic mail, Domain Name System (DNS).

Readings:

1. Tanenbaum, A. S. *Computer networks* (5th ed.). Pearson Education India, 2013.
2. Forouzan, B. A. *Data communications and networking* (5th ed.). McGraw-Hill Education, 2017.
3. Kurose, J. F. *Computer networking: A top-down approach* (9th ed.). Pearson, 2024.

List of Practicals:

1. Explore the network by capturing the packets using network packet analyzer, apply a filter for HTTP packets and report the observed findings, such as version, IP address of client and server, status code, size of packet, protocols, in a PDF. **(2 hours)**
2. Explore the ARP packets (request and reply) using a network packet analyzer and retrieve their corresponding fields. ARP Request: payload, opcode, query. Response: Payload, opcode, response, source and destination address. **(2 hours)**
3. Simulate ARP and RARP networking protocols using a Python script. **(4 hours)**
4. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for the noisy channel**. (4 hours)**
5. Write a python program to implement Checksum and Hamming (7, 4) coding for error detection and error correction in data transmission. Simulate single-bit errors, verify data integrity and display corrected output. **(4 Hours)**
6. Simulate and implement the Stop and Wait protocol for the noisy channel. **(2 hours)**
7. Simulate and implement the Go Back N and Selective Repeat sliding window protocols. **(4 hours)**
8. Write a python script to demonstrate the transmission of an IP Packet between two nodes in a network**. (2 hours)**
9. Write a Python script to demonstrate a 3-way handshake synchronization process of the TCP protocol between two nodes. Set the Urgent Flag to '1' and Destination Port to '5394' in the packet. Also, verify the details of the packet received on the receiver end. **(4 hours)**
10. Write a python program to simulate the DNS protocol to resolve domain name to the corresponding IP address. **(2 hours)**

DSC302: INFORMATION SECURITY [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSC201	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:

This course aims to provide a comprehensive understanding of security principles, threats, and cryptographic techniques used in modern computing systems. It covers security models, authentication mechanisms, and key management protocols to safeguard data and

communication. The course also emphasizes real-world applications of security frameworks, to enhance secure communication and data protection.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to:

1. analyze the distinction between system protection and security, emphasizing their role in safeguarding digital assets
2. implement symmetric and asymmetric encryption algorithms
3. describe the role and implementation of digital signatures.
4. understand how cryptographic methods and security models are applied in practical systems like secure communication and e-commerce

Syllabus:

Unit-I (4 hours)

Overview of Security: Protection versus security; aspects of security– confidentiality, data integrity, availability, privacy; user authentication, access controls. Security Threats and Models: Program threats – worms, viruses, Trojan horse, trap door, stack and buffer overflow; system threats – intruders; communication threats- tapping and piracy.

Unit-II (20 hours)

Cryptography: Substitution, transposition ciphers, symmetric-key algorithms: Data Encryption Standard, Advanced Encryption Standard, IDEA, block cipher operation, stream ciphers: RC-4. Public key encryption: knapsack, RSA, El Gamal, Elliptic curve cryptography.

Unit-III (15 hours)

Integrity and authentication: Message Integrity - MDC, MAC, cryptographic hash function - iterated hash functions, compression functions, SHA-512, Digital signatures -RSA digital signature scheme, ElGamal digital signature scheme, digital signature standard (DSS), elliptic curve digital signature scheme, entity authentication.

Unit-IV (6 hours)

Key Management and Intrusion Detection & Prevention: Symmetric key distribution, Kerberos, Diffie-Hellman key agreement, public key distribution, public key infrastructures, and Intrusion Detection and Prevention Systems.

Readings:

1. Stalling, William. *Cryptography and Network Security Principles and Practices*. 8th edition, Pearson education, 2023
2. Forouzan, Behrouz A., and Debdeep Mukhopadhyay. *Cryptography and Network Security.* 3rd edition. McGraw-Hill Education, 2015.
3. Elbirt, Adam. J. *Understanding and Applying Cryptography and Data Security*. CRC Press, Taylor Francis Group, 2015.
4. Pfleeger, Charles P. SL Pfleeger, Jonathan Margulies. *Security in Computing*. 5th edition. Prentice-Hall of India, 2018.

List of practical's:

Students may choose a suitable programming language to do the following lab exercises.

1. Implement the extended Euclidean algorithm **(2 hours)**
2. Implement Encryption and decryption using the Additive Cipher **(2 hours)**
3. Cryptanalysis of the Additive Cipher **(2 hours)**
4. Implement Encryption and decryption using the Multiplicative Cipher **(2 hours)**
5. Cryptanalysis of the Multiplicative Cipher **(2 hours)**
6. Implementation of columnar transposition cipher for encryption and decryption of messages. **(2 hours)**
7. Encryption and decryption of files using python's cryptography library to demonstrate symmetric encryption and asymmetric encryption algorithms discussed in the class **(10 hours)**
8. Computation of SHA-512 Hash and HMAC for file integrity verification using a Shared Secret Key. **(4 hours)**
9. Verification of file integrity and displaying "Integrity Verified" or "Integrity Compromised" Based on various hashing algorithms. **(4 hours)**

SBC301: Prompt Engineering [0-0-2]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
SBC301	2	0	0	2	Graduation	NIL

Course Objective:

The objective of this course is to provide hands-on experience in understanding, designing, and optimizing prompts for large language models (LLMs), covering foundational and advanced prompting techniques.

Course Learning Outcomes:

Upon successful completion of this course, students will

1. understand the foundational concepts of prompting and LLM behaviour.
2. design structured, effective prompts using templates, constraints, roles, and examples.
3. apply advanced prompting methods such as chain-of-thought, few-shot prompting, ReAct, and self-consistency.
4. debug, evaluate, and optimize prompts for accuracy, clarity, and reliability.
5. design task-specific prompt workflows for reasoning, transformation, and creative tasks.

Syllabus:

Unit I: Foundations of Prompting

Introduction to LLMs and text generation; tokens and context windows; deterministic vs.

stochastic decoding; Basic prompt structures—role prompting, instruction prompting, format constraints, delimiting techniques; Prompt templates, input–output structuring, controlling style and tone; Evaluating prompts—hallucinations, ambiguity, and prompt robustness; Prompt debugging essentials.

Unit II: Advanced Prompt Patterns

Few-shot prompting, example ordering and construction; Chain-of-Thought (CoT) prompting, self-consistency, multi-step reasoning; ReAct Prompting (Reason + Act); Prompt compression, prompt optimization strategies; Hallucination mitigation and safety prompting.

Readings:

1. Phoenix, J., & Taylor, M. (2024). *Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs*. O'Reilly Media.
2. DAIR.AI. (2024). *Prompt Engineering Guide*. https://www.promptingguide.ai
3. *Prompt engineering (OpenAI Documentation)*. https://platform.openai.com/docs/guides/prompting
4. Kojima, T., Gu, S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). *Large language models are zero-shot reasoners*. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*. https://arxiv.org/abs/2205.11916
5. Wei, J., Wang, X., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., & Zhou, D. (2022). *Self-Consistency improves chain of thought reasoning in language models*. In *International Conference on Learning Representations (ICLR 2023)*. https://arxiv.org/abs/2203.11171
6. Yao, S., Zhao, J., Yu, D., Du, N., & Shafran, I. (2022). *ReAct: Synergizing reasoning and acting in language models*. https://arxiv.org/abs/2210.03629

List of Practicals

1. Accessing LLM playground/API and experimenting with temperature, top-p, max tokens; Observe behavior changes with different parameters. **(4 hours)**
2. Write prompts for summarization, rephrasing, extraction; Refine prompts using delimiters and explicit constraints. **(4 hours)**
3. Design prompts with roles (teacher, analyst, developer, critic); Test how role affects style, depth, and tone. **(4 hours)**
4. Generate structured outputs (tables, bullet lists, JSON-like format); Enforce formatting rules using clear instructions and delimiters. **(4 hours)**
5. Create task-specific prompt templates; Experiment with placeholders and reusable modules. **(4 hours)**
6. Study cases of ambiguous or misleading prompts; Debug hallucinations and refine instructions. **(4 hours)**
7. Apply criteria like clarity, consistency, correctness; Score outputs against evaluation rubrics. **(4 hours)**
8. Design 1-shot, 2-shot, and 5-shot examples; Experiment with example selection and ordering. **(4 hours)**
9. Apply few-shot prompts to classification, reasoning, summarization; Compare

performance against zero-shot. **(4 hours)**

10. Add intermediate reasoning steps; Use CoT for math problems, logic puzzles, and complex decision tasks. **(4 hours)**
11. Generate multiple reasoning paths; Compare and reconcile different answers. **(4 hours)**
12. Implement ReAct-style prompts for stepwise reasoning; Use reasoning traces followed by actions. **(4 hours)**
13. Shorten prompts while preserving performance; Experiment with paraphrasing, constraint tightening, and anchor examples. **(4 hours)**
14. Identify unsafe outputs; Apply safety prompting patterns and refusal formats. **(4 hours)**
15. Students build a small prompt-based system, such as: **(4 hours)**
    - a reasoning assistant
    - a structured information extractor
    - a creative writing pipeline
    - a domain-specific Q&A prompt workflow

    Includes demonstration and evaluation.

List of DSEs for Semester III

DSE301: Link Prediction [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE301	**4**	**3**	**0**	**1**	Graduation	**Basic knowledge of Graph Theory and social network analysis**

Course Objectives:

The student learns the theoretical foundations and important properties of link prediction in networks. The students learn and explore similarity-based, probabilistic, and learning-based methods for link prediction. They can apply link prediction techniques in real-world domains such as social networks, biological networks, and recommendation systems.

Course Learning Outcomes:

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

1. Understand basic concept and importance of link prediction.
2. Simulate link prediction approaches in network datasets.
3. Design and implement link prediction experiments on real-world datasets.
4. Evaluate link prediction models using standard performance measures.

Syllabus:

Unit-I **(9 hours)**

Introduction: Definition and motivation for link prediction; Applications of link prediction: social networks, biological networks, recommendation systems, citation networks; Problem formulation and evaluation framework, dimensionality reduction, and community detection.

Unit-II **(15 hours)**

Similarity-based methods: Local similarity: Common Neighbors, Jaccard Coefficient, Adamic-Adar, Resource Allocation Index; Global similarity: Katz Index, Hitting Time, Rooted PageRank, SimRank, Path-based and walk-based similarity measures; Structural and topological features for link prediction.

Unit-III **(15 hours)**

Probabilistic and statistical methods: Local probabilistic model, Probabilistic relational model (PRM), Hierarchical structure model (HSM), Stochastic block model (SBM), Exponential random graph model (ERGM).

Other methods: Dimensionality reduction-based link prediction, Information Diffusion-based Link Prediction, Learning-based frameworks, Information theory-based, Clustering-based, Structural perturbation method (SPM)

Unit-IV **(6 hours)**

Evaluation metrics: Recall, Area under the precision–recall curve (AUPR), Area under the receiver operating characteristics curve AUROC), and Average precision

Readings:

1. Newman, M. E. J. *Networks: An introduction*. Oxford University Press, 2010.
2. Easley, D., & Kleinberg, J. *Networks, crowds, and markets*. Cambridge University Press, 2010.
3. Barabási, A.-L. *Network science*. Cambridge University Press, 2016.
4. Golbeck, J. *Analyzing the social web*. Morgan Kaufmann, 2013.
5. Wasserman, S., & Faust, K. *Social network analysis: Methods and applications*. Cambridge University Press, 2012.
6. Kolaczyk, E. D. *Statistical analysis of network data: Methods and models*. Springer, 2009.
7. Estrada, E. *The structure of complex networks: Theory and applications*. Oxford University Press, 2011.
8. Srinivas, V., & Mitra, P. *Link prediction in social networks: Role of power law distribution*. Springer International Publishing, 2016.

List of Practicals:

1. Extract structural features (degree, clustering coefficient, centrality) and apply the dimensionality reduction method PCA to visualise nodes in a reduced feature space. (**4 hours**)
2. Implement Common Neighbours, Jaccard Coefficient, Adamic–Adar, and Resource Allocation Index manually and validate with NetworkX built-in functions. (**4 hours**)
3. Write a Python function to compute the Katz Index using matrix operations (numpy.linalg.inv), with damping factor β. (**2 hours**)
4. Compute Rooted PageRank (Personalized PageRank) from a given root node using the random walk with restart method. (**2 hours**)
5. Compute Hitting Time between all pairs of nodes using BFS-based random walks. (**2 hours**)
6. Implement SimRank recursively and analyze how similarity scores evolve over iterations. (**4 hours**)
7. Generate all paths between node pairs up to length 3 and calculate similarity based on path counts. (**2 hours**)
8. Simulate Independent Cascade (IC) or Linear Threshold (LT) models and estimate missing links based on influence overlap. (**4 hours**)
9. Implement Recall, Precision, AUPR, AUROC, and Average Precision. (**4 hours**)
10. Compare accuracy (AUC, precision) between attribute-based PRM and structure-based probabilistic models. (**2 hours**)

DSE302: RECOMMENDER SYSTEMS [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE302	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:
The student learns the fundamental and advanced concepts of recommender systems, including collaborative filtering, content-based, hybrid, and context-aware techniques. The student develops analytical and technical skills to design, implement, and evaluate recommender algorithms using real-world datasets and modern machine learning frameworks.

Course Learning Outcomes:
On completing this course, students will be able to:

1. explain the theoretical foundations, types, and working principles of recommender systems.
2. apply collaborative filtering, content-based, and hybrid methods to build personalised recommendation models.
3. analyze user–item interactions and contextual factors influencing recommendation quality.
4. evaluate recommender system performance using standard metrics, validation methods, and benchmark datasets.
5. design and create advanced recommender systems integrating deep learning, graph-based, and sequential modeling techniques for real-world applications.

Syllabus:

Unit-I **(11 hours)**

Introduction to Recommender Systems: Purpose, significance, and real-world applications; types of recommender systems. Data sources including user profiles, item attributes, and implicit and explicit feedback. Data preprocessing techniques for effective recommendation. Evaluating Recommender Systems: Online and offline evaluation methodologies; performance metrics such as RMSE, MAE, Precision, Recall, F1-score, NDCG, Average Reciprocal Rank, and Top-K measures. Beyond-accuracy evaluation covering fairness, coverage, diversity, novelty, and serendipity to ensure a comprehensive assessment of recommender systems.

Unit-II **(14 hours)**

Collaborative Filtering – Neighbourhood-Based Approaches: Introduction, user-based and item-based models, similarity measures, and prediction functions; efficiency analysis and comparative evaluation; clustering and dimensionality reduction techniques; regression-based and graph-based extensions. Latent Factor Models and Matrix Factorization: Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), Stochastic Gradient Descent (SGD), and Alternating Least Squares (ALS) with regularization.

Unit-III **(10 hours)**

Content-Based Recommender Systems: Introduction and Basic Components, Preprocessing and Feature Engineering, Collecting User Preferences, Supervised Feature Selection and Weighting Techniques, Learning User Profiles and Filtering Approaches. Content-Based vs. Collaborative Recommendations. Hybrid recommender systems combining collaborative and content-based filtering.

Unit-IV **(10 hours)**

Advanced Topics – Deep Learning-Based Recommender Systems: Neural Approaches, Autoencoders, Sequence-Aware Models (RNN, LSTM, Transformer), Graph Neural Network-Based Recommenders, Explainable Recommendations, Fairness and Bias Mitigation, Multi-Modal and Context-Aware Recommendation, Emerging Trends and Research Directions.

Readings:

1. Aggarwal, Charu C., *Recommender Systems: The Textbook*, Springer International Publishing, 2016.
2. Ricci, Francesco, Lior Rokach, and Bracha Shapira (Ed.), *Recommender Systems Handbook*, Third Edition, Springer, 2022.
3. Advances in recommender systems using recent research papers.

List of Practicals:

1. Implement a user-based collaborative filtering recommender system using a given dataset (e.g., MovieLens). **(4 hours)**
2. Implement an item-based collaborative filtering recommender and compare its performance with the user-based model. **(2 hours)**
3. Perform matrix factorisation using Singular Value Decomposition (SVD) on a user-item rating matrix and predict missing ratings. **(4 hours)**
4. Implement non-negative matrix factorisation (NMF) on a dataset and analyse the latent factors for top-N recommendation. **(2 hours)**
5. Build a content-based recommender system using item features (e.g., genres, tags, descriptions) and evaluate using cosine similarity. **(2 hours)**
6. Implement a hybrid recommender system combining collaborative and content-based approaches and evaluate improvements over individual models. **(4 hours)**
7. Build a sequence-aware recommendation model using RNN or LSTM and evaluate next-item prediction performance. **(6 hours)**
8. Implement a graph-based recommender system using a Graph Neural Network (GNN) and analyse user-item interaction patterns. **(6 hours)**

DSE303: Time Series Data Analysis [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE303	**4**	**3**	**0**	**1**	Graduation	**Basic knowledge of programming and statistics**

Course Objectives:

This course provides a comprehensive understanding of time series analysis, covering fundamental principles, statistical modeling techniques, and forecasting methods. The course balances theoretical concepts with hands-on applications using real-world datasets in R and Python.

Course Learning Objectives:

By the end of this course, students will be able to:

1. Understand the basic properties and components of time series data.
2. Perform stationarity tests and data transformations.
3. Apply statistical and machine learning models for time series forecasting.
4. Analyse multivariate time series data and apply feature engineering techniques.
5. Implement real-world forecasting applications across different domains.

Syllabus:

Unit-I **(10 Hours)**

Fundamentals of Time Series Analysis: Introduction to Time Series Data, Time Series Components, Trend, Seasonality, Cyclicality, and Noise, Time Series Data Visualization and Preprocessing, Stationarity and Unit Root Tests (ADF, KPSS).

Unit-II **(12 Hours)**

Time Series Models and Forecasting: Moving Average and Exponential Smoothing Models, Autoregressive (AR), Moving Average (MA), and ARMA Models, ARIMA and Seasonal ARIMA (SARIMA), Model Selection and Diagnostics (AIC, BIC, Residual Analysis), Performance Metrics.

Unit-III **(11 Hours)**

Advanced Time Series Methods: State-Space Models and Kalman Filters, Spectral Analysis and Fourier Transforms, Long Memory Processes and Fractional Differencing, and Nonlinear Time Series Models.

Unit-IV **(12 Hours)**

Multivariate Time Series and Machine Learning Approaches: Vector Autoregression (VAR) and Cointegration, Dynamic Factor Models and PCA, Machine Learning for Time Series (LSTMs, XGBoost, Prophet), Feature Engineering and Dimensionality Reduction.

Readings:

1. Brockwell, Peter J., and Richard A. Davis, eds. *Introduction to time series and forecasting*. New York, NY: Springer New York, 2002.
2. Montgomery, Douglas C., Cheryl L. Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.
3. Beran, Jan., *Mathematical Foundations of Time Series Analysis*. Switzerland: Springer, 2017.
4. Wei, William W. S., *Multivariate Time Series Analysis and Applications*. United Kingdom: Wiley, 2019.
5. Hyndman, Rob J., Athanasopoulos, George. *Forecasting: Principles and Practice*. United Kingdom: OTexts, 2021. [https://otexts.com/fpp3/index.html]

List of Practicals:

1. Write a program that reads a time series with missing timestamps, fills or interpolates missing values, resamples the data at a different frequency (e.g., converting daily data

into monthly averages), and removes outliers using z-score or IQR. Plot the data before–after pre-processing **(2 Hours).**

2. Write a program that loads a time series dataset (such as AirPassengers or daily stock prices) and then plots the raw data along with a rolling mean and rolling standard deviation. The program should also generate autocorrelation (ACF) and partial autocorrelation (PACF) plots. **(2 Hours)**
3. Write a program to perform additive and multiplicative decomposition on a given time series using STL or classical decomposition. The program should extract and plot the trend, seasonal, and residual components separately, and print a short interpretation describing the nature of seasonality and trend. **(2 Hours)**
4. Write a program that visually checks stationarity using rolling mean/variance plots and then performs ADF and KPSS tests on the original series. The program should apply first-order differencing or log transformation until the series becomes stationary and then re-run the tests, printing the p-values and conclusions. **(2 Hours)**
5. Write a program to apply simple moving average, weighted moving average, and exponential smoothing to a time series. The program should compare smoothed curves on a single plot and show how different window sizes or smoothing factors affect the trend. It should also print summary statistics comparing original and smoothed series. **(2 Hours)**
6. Write a program that analyzes the ACF and PACF plots to choose tentative orders for AR, MA, and ARMA models. Then fit AR(p), MA(q), and ARMA(p,q) models, print model parameters, and perform residual analysis using the Ljung–Box test. Evaluate the performance of the model using MAE, RMSE, MAPE, and sMAPE. **(6 Hours)**
7. Write a program that identifies the required differencing order using ADF/KPSS and then fits an ARIMA model with manually selected or auto-selected orders. The program should also fit a SARIMA model if seasonality exists, compare AIC/BIC values, and display forecast plots along with residual diagnostics **(2 Hours).**
8. Write a program that builds a state-space representation of a time series (e.g., a local linear trend model), applies the Kalman filter to estimate hidden states, and plots filtered and smoothed estimates. The program should compare the Kalman-based trend with the trend obtained using classical smoothing **(2 Hours).**
9. Write a program that computes the periodogram of a time series, applies the Fast Fourier Transform (FFT), and identifies the dominant frequencies contributing to seasonality. The program should print the main repeating cycle (e.g., weekly, yearly) based on the spectral peaks**. (2 Hours)**
10. Write a program that loads two or more related time series (such as GDP, inflation, and interest rates), checks their stationarity, and fits a VAR model. The program should compute impulse response functions and plot how one variable responds over time to a shock in another variable. **(2 Hours)**
11. Write a program that applies PCA to a multivariate time series dataset, extracts major components, and reconstructs the series using those components. Then fit a dynamic factor model and compare the variance explained by PCA factors versus dynamic factors. **(2 Hours)**
12. Write a program that converts a time series into a supervised learning dataset by

generating lag features, rolling statistics, and date-time features. Train XGBoost or Prophet models on the processed data, generate forecasts, and compare their performance with ARIMA using common error metrics. **(2 Hours)**

13. Write a program that converts the time series into sliding windows, normalizes the data, and trains an LSTM or GRU neural network for next-step prediction. The program should plot predicted vs actual values and compare LSTM accuracy with traditional models such as ARIMA or exponential smoothing**. (2 Hours)**

DSE304: QUANTUM COMPUTING AND ITS APPLICATIONS [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSC304	**4**	**3**	**0**	**1**	Graduation	**Basic knowledge of Number theory or information security**

Course Objectives:

This course provides a foundation for quantum computing, post-quantum cryptography and quantum machine learning. It covers the fundamental concepts of quantum mechanics, quantum algorithms, and their applications in various areas, including cryptography, cybersecurity, machine learning, finance, the energy sector, etc. Students will gain a theoretical understanding of quantum computing and practical skills in implementing quantum algorithms for various tasks.

Course Learning Outcomes:

On completing this course, the student will be able to:

1. Understand the basic principles of quantum mechanics and their relevance to quantum computing.
2. Comprehend quantum algorithms and their applications.
3. Apply quantum optimisation techniques in problem-solving.
4. Demonstrate practical skills in quantum computing in various areas, including cryptography and machine learning.

Syllabus:

Unit-I **(10 hours)**

Fundamentals of Quantum Computing: Mathematical foundations: Vectors, Vector space, Inner product, Tensor Product; Qubits, Introduction to quantum mechanics and its relevance to Quantum gates, superposition principle, and entanglement Quantum parallelism and interference, No cloning theorem, quantum teleportation, Introduction to quantum error

Correction.

Unit-II **(15 hours)**

Quantum Algorithms and Post-Quantum Security: Phase Kick-Back Algorithm, Deutsch-Jozsa algorithm, Simon's algorithm, Bernstein-Vazirani, RSA algorithm and factorization attack on RSA, Shor's algorithm for integer factorization, Grover's algorithm for unstructured search, Hash preimage attack with Grover's algorithm, Quantum Fourier transform and its applications, Harrow–Hassidim–Lloyd (HHL) algorithm, Kyber Algorithm, Quantum attack resistant Digital Signature.

Unit-III **(10 hours)**

Quantum Machine Learning and Optimization: Quantum machine learning (QML) models – QSVM, QNN, QCNN, Quantum Linear Regression, Variational Quantum Classifier (VQC), Quantum k-means clustering; kernel methods, Quantum Boltzmann Machines; Quantum optimization techniques: QAOA, quantum annealing.

Unit-IV: **(10 hours)**

Introduction to quantum simulation tools and platforms: Google CIRQ, Amazon Braket, IBM Qiskit, Pennylane, Q#, Tensorflow quantum, Tket/pyket, XACC, Project Q, Quantum Development Kit (QDK).

Readings:

1. Noson S. Yanofsky and Mirco A. *Mannucci. Quantum computing for computer scientists*. Cambridge University Press, 2008.
2. Michael A Nielsen, Isaac L. Chuang, *Quantum computation and quantum Information*, Cambridge University Press.
3. Elias F. Combarro, Samuel González-Castillo, and Alberto Di Meglio. *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms*. Packt Publishing Ltd, 2023.

References:

1. Santanu Ganguly. Quantum Machine Learning: An Applied Approach. Apress, 2021.
2. https://docs.quantum.ibm.com/
3. https://quantumai.google/cirq/experiments/textbook_algorithms

List of practical:

1. Install and configure a quantum computing framework such as IBM Qiskit or Google Cirq, and create basic quantum circuits using single-qubit gates to perform measurements. **(3 Hours)**

2. Write a program to visualise qubit states and their rotations on the Bloch Sphere using suitable simulation tools. **(2 Hours)**

3. Implement a program to demonstrate the concepts of superposition and entanglement

using multi-qubit circuits. **(3 Hours)**

4. Write a program to simulate the process of quantum teleportation using an entangled qubit pair. **(3 Hours)**
5. Implement the Deutsch–Jozsa algorithm to identify whether a given function is constant or balanced. **(3 Hours)**
6. Write a program to implement Grover's search algorithm for an unstructured search problem and analyze its efficiency**. (4 Hours)**
7. Implement a simplified version of the Quantum Fourier Transform (QFT) for a small number of qubits and observe the results. **(3 Hours)**
8. Write a program to demonstrate the Quantum Approximate Optimisation Algorithm (QAOA) for solving simple optimisation problems. **(3 Hours)**
9. Write a program to generate and analyze quantum random numbers using measurement of superposition states. **(4 Hours)**
10. Write and execute quantum circuits on different quantum-computing platforms and compare their performance and features**. (2 Hours)**

DSE305: Digital Forensics [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE305	**4**	**3**	**0**	**1**	Graduation	**Basic knowledge of Computer Networks and operating systems**

Course Objective:

This course provides a comprehensive introduction to the principles, practices, and legal frameworks of digital forensics. Students will gain hands-on experience in the complete investigation lifecycle—from evidence acquisition and validation to in-depth analysis of computer, mobile, and network data. The course prepares students to conduct rigorous digital investigations and effectively report on findings in accordance with modern Indian legal standards.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to:

1. apply the forensic investigation process in compliance with Indian legal acts and ethical guidelines.
2. acquire and validate data from live and dead systems, including hard drives (disk imaging) and volatile memory (RAM).
3. analyse and reconstruct digital evidence from various file systems (FAT, NTFS, ext4), memory dumps, and core operating system artefacts (e.g., Windows Registry, browser history).
4. analyse evidence from specialised sources, including mobile devices (Android) and network packet captures (PCAPs).
5. Evaluate the challenges and methodologies associated with emerging threats, such as deepfake media, AI-generated evidence, and dark web artefacts.

Syllabus

Unit I: (8 hours)

Introduction: Definition, distinction between cyber forensics and Digital forensics, Evolution of computers and digital forensics, modern challenges in the AI and cloud era. Digital Footprints and Attack Vectors, Case Studies (Indian and Global), Digital forensics Investigation process: Identification, Preservation, Collection and Acquisition, Analysis, Documentation & Presentation, Numbering Systems, Data structure – endianness, character encoding, Data nature and state, Reporting: Best practices for making Forensic reports

Unit II: (10 hours)

Legal and Ethical Frameworks: IT Act, 2000 (amendments of 2008) and IT (Intermediary Guidelines and Digital Media Ethics Code) Rules, 2021 (and subsequent amendments), BNS 2023, BNSS 2023, BSA 2023, DPDP Act, 2023, Budapest Convention, ISO/IEC Standards in Forensics, Chain of custody, 65A/65B Certificates, Ethical dilemmas in Forensics, CERT-in Procedures and Coordination

Unit III: (15 hours)

Acquisition and Analysis: Acquisition: Live vs Dead Acquisition, Methods for Data Acquisition, Hashing (MD5, SHA) and Data Validation. File System Analysis: FAT, NTFS, ext4, methods for recovering data from deleted files, File carving, Memory forensics: Volatile data collection and analysis, analysing memory dumps for processes, network connections, OS Artifacts analysis: Time reconstruction, Email Header analysis and tracing, Windows registry analysis, browser history, anti-forensics technique and detection

Unit IV: (12 hours)

Advanced and Emerging Forensics: Mobile Forensics: Overview of acquisition and analysis of Android, unique challenges: diverse OS, encryption, complex appdata, Legal and ethical considerations, Network Forensics: analysing network traffic (PCAP files); Emerging Threats: Dark Web Forensics, Synthetic Media and Deepfake Detection; AI-generated Evidence:

Admissibility Challenges.

References:

1. Dejey, M. *Cyber forensics*. Oxford University Press India, 2018.
2. Moustafa, N. *Digital forensics in the era of artificial intelligence* (1st ed.). CRC Press, 2023.
3. Gogolin, G. *Digital forensics explained* (2nd ed.). CRC Press, 2021.
4. Kävrestad, J. *Fundamentals of digital forensics: Theory, methods, and real-life applications*. Springer International Publishing AG, 2020

List of Practicals

1. Consider a scenario in which you are a junior forensic analyst. You received a sealed 16GB USB drive (Evidence #001). Create a new case in Autopsy, fill out a chain of custody form to document its receipt, and then draft a sample 65B certificate for a hypothetical file (e.g., secret_ledger.xlsx) found on the drive to understand legal documentation. **(4 hours)**

2. Explore WinHex and Hex Workshop to analyse and examine various file headers. **(4 hours)**

3. Install FTK Imager and acquire the volatile memory (RAM) of your system. **(2 hours)**

4. Acquire non-volatile memory (hard disc) using FTK Imager **(2 hours)**

5. Analyse the disk image created in the previous lab by loading it into Autopsy. Examine the NTFS structure, with a focus on the Master File Table (MFT). Analyse MFT entries for deleted files, looking specifically at how the FILE_NAME attribute (flagged as ;in-use and/or deleted) and timestamp attributes change. Using this analysis, identify and recover five specific deleted files, documenting their original file paths. **(4 hours)**

6. In Gmail, select a SPAM email and analyse its header. Detect possible spoofing or malicious activity using a mail header analyser. https://mxtoolbox.com/EmailHeaders.aspx. Identify key header fields such as From, To, Subject, Date, Return-Path, Received, Message-ID, and SPF / DKIM / DMARC. Use tools like WHOIS or online IP lookup services to identify the geographical location and ownership of IP addresses found in the Received lines. Verify if any IPs are suspicious or if the hostname doesn't match the expected sending server. **(2 hours)**

7. Visit the website https://testphp.vulnweb.com/ and start capturing packets using Wireshark. Some HTTP packets will be captured. Look for form data that the user submitted to the website. Find the user credentials sent in the HTTP request. **(2 hours)**

8. Analyse a provided Windows memory dump using the Volatility framework. You will identify the correct OS profile, then run commands to extract the list of running

processes (pslist), identify active network connections (netscan), and dump command history (cmdscan) **(6 hours)**

9. Analyse a provided logical Android backup (android_backup.ab) using Autopsy. You will extract and query key databases to document recent calls (calllog.db), find specific text messages (mmssms.db), and locate the WhatsApp message database (msgstore.db). **(2 Hours)**

10. Using Registry Explorer and the RegRipper tool, analyse the SAM registry Hive and identify your own user SID. **(2 hours)**

DSE306: Human Computer Interaction [3-0-1]

Course Code	**Credits**	**Credit distribution of the course**			Eligibility Criteria	**Pre-requisite of the course (if any)**
		Lecture	**Tutorial**	**Practical/Practice**		
DSE306	4	3	0	1	Graduation	Programming language and prototypes of computer interfaces

Course Objectives:

The course aims to introduce students to the fundamental concepts of human-computer interaction and their application in society.

Course Learning Outcomes:

Upon successful completion of this course, the student will-

1. understand basic theories, tools and techniques in HCI.
2. empathize with the fundamental aspects of designing and evaluating interfaces.
3. apply appropriate HCI techniques to design systems that people use.

Syllabus:

UNIT I: **(12 Hours)**

HCI Foundations- Input–output channels, human memory, thinking: reasoning and problem solving, emotion, individual differences, psychology and the design of interactive systems; text entry devices, positioning, pointing and drawing, display devices, devices for virtual reality and 3D interaction, physical controls, sensors and special devices, paper: printing and scanning; models of interaction, frameworks and HCI, industrial interfaces, interaction styles, navigation in 3D and 2D, elements of the WIMP interface, learning toolbars, interactivity, the context of

the interaction; paradigms.

UNIT II: (14 Hours)

Design Process- Process of design, scenarios, navigation design, beware the big button trap, modes, screen design and layout, alignment and layout matter, checking screen colors, iteration and prototyping; HCI in the software process- the software life cycle, usability engineering, iterative design and prototyping, design rationale; design rules- principles to support usability, standards, golden rules and heuristics, HCI patterns; elements of windowing systems, programming the application, user interface management systems; evaluation through expert analysis, evaluation through user participation, choosing an evaluation method; universal design principles, multi-modal interaction, designing for diversity; requirements of user support, approaches to user support, designing user support systems.

UNIT III: (9 Hours)

Models and Theories- cognitive models- introduction, goal and task hierarchies, linguistic models, physical and device models, cognitive architectures; organizational issues, capturing requirements; communication and collaboration models- introduction, face-to-face communication, conversation, text-based communication; task analysis- introduction, task decomposition, knowledge-based analysis, entity–relationship-based techniques.

UNIT IV: (10 Hours)

Dialog design notations, diagrammatic notations, dialog semantics, dialog analysis and design; interaction models; rich contexts, low intention and sensor-based interaction; groupware-introduction and systems, meeting and decision support systems, shared applications and artifacts, frameworks for groupware, implementing synchronous groupware; virtual and augmented reality, information and data visualization.

Readings:

1. Dix, A., Finlay, J., Abowd, G. D., & Beale, R. *Human–computer interaction* (3rd ed.). Pearson, 2008.
2. Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. *Designing the user interface: Strategies for effective human–computer interaction* (6th ed.). Pearson, 2021.
3. Bhattacharya, S. *Human–computer interaction* (1st ed.). McGraw-Hill, 2019.

References:

1. Galitz, Wilbert O., *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*, 3rd edition, Wiley Publishing, 2007.

List of practical:

You may choose a suitable programming language and define all necessary/relevant inputs/ problems.

1. Implement a program for short-term memory and long-term memory. **(2 Hours)**

2. Implement a program to perform natural language interfaces. **(2 Hours)**

3. Write a program for a full-page word-processor that is or is not a direct manipulation interface for editing a document using Shneiderman's criteria. **(4 Hours)**

4. Write a program to find a book on guidelines. List the guidelines that are provided and classify them in terms of the activity in the software life cycle to which they would most likely apply. **(2 Hours)**

5. Implement a program for a keystroke level analysis for opening up an application in a visual desktop interface using a mouse as the pointing device, comparing at least two different methods for performing the task. Repeat the exercise using a trackball. **(4 Hours)**

6. Write a program to test whether adding colour coding to an interface will improve accuracy. **(2 Hours)**

7. You have been asked to compare user performance and preferences with two different learning systems, one using hypermedia and the other sequential lessons. Write a program to design a questionnaire to find out what the users think of the system. **(4 Hours)**

8. Implement a program for online support systems. **(2 Hours)**

9. Write a program for the operations in a standard drawing package (for example, draw, move, copy, delete, rotate). **(4 Hours)**

10. Write a program for Icon Design and VCR Remote Control. **(4 Hours)**

DSE307: Influence Maximization [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE307	**4**	**3**	**0**	**1**	Graduation	**Basic knowledge of Graph Theory and social network analysis**

Course Objectives:

This course provides a comprehensive introduction to the principles of information diffusion and influence propagation in networks. To learn classical and advanced influence maximization algorithms. To analyze efficient and effective algorithms for influence maximization under various constraints. To study evaluation metrics and performance analysis for influence maximization. To apply influence maximization techniques in real-world applications such as viral marketing, epidemic control, and opinion formation.

Course Learning Outcomes

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

1. Understand the basic concept of influence maximization in networked datasets.
2. Model and simulate the spread of influence using classical propagation models.
3. Identify influential nodes using various influence maximization algorithms.
4. Analyze and compare algorithmic efficiency and approximation guarantees of influence maximization algorithms.

Syllabus:

Unit-I **(8 hours)**

Introduction: Overview of social networks and types of social networks, Network structure and properties (degree, clustering, centrality), social network analysis concepts, viral marketing, rumor spreading, and epidemic diffusion; real-world applications, NP-hardness, monotone and sub-modularity properties.

Unit-II **(12 hours)**

Information Diffusion Models: Basics of information diffusion and influence propagation, Independent Cascade (IC) model, Linear Threshold (LT) model, SIR: variants and extensions (weighted, time-aware, competitive, signed), Influence spread and expected diffusion, Analytical and simulation-based evaluations.

Unit-III **(13 hours)**

Classical Influence Maximization Algorithms: Simulation-based Algorithms: Greedy algorithms and approximation guarantees, CELF, and CELF++ optimization; Heuristic-based Algorithms: Degree, PageRank, Centrality-based and Community-based; Sampling-based Algorithms; Path-based Algorithms. Performance evaluation metrics.

Unit-IV **(12 hours)**

Advanced Influence Maximization Algorithms: Budgeted Influence Maximization, Influence maximization in Dynamic and Temporal Networks, Competitive Influence Maximization, Location-aware Influence Maximization, Topic-aware Influence Maximization, Conformity and Semantic-based Influence Maximization.

Readings:

1. Newman, M.E.J. *Networks: An introduction*. Oxford University Press, 2010.
2. Easley D. and Kleinberg J. *Networks, Crowds, and Markets.* Cambridge University Press, 2010.
3. Barabási A.L. *Network Science.* Cambridge University Press, 2016.
4. Kolaczyk, E. D. *Statistical Analysis of Network Data: Methods and Models.* Springer, 2009.
5. Estrada, E. *The Structure of Complex Networks: Theory and Applications.* Oxford University Press, 2011
6. Chen, W. Castillo, C and Lakshmanan, L.V.S. *Information and influence propagation in social networks*, Springer Nature, 2014

List of Practicals:

1. Create a graph from different types of input files and display the nodes and edges properties. **(4 hours)**
2. Implement the Independent Cascade Model (ICM) to observe the spread of influence from a given set of seed nodes. **(2 hours)**
3. Implement the Linear Threshold Model (LTM) to observe the spread of influence from a given set of seed nodes. **(2 hours)**
4. Select top-k nodes based on degree centrality and evaluate influence spread under the IC model. **(2 hours)**
5. Compare influence spread using degree, closeness, betweenness, and eigenvector centrality as seed criteria. **(4 hours)**
6. Implement the Greedy and CELF algorithms to efficiently select the top-k influential nodes for maximizing the influence spread. **(4 hours)**
7. Detect communities using the Louvain or Girvan–Newman algorithm. Select seed nodes evenly from the identified communities and compare the influence spread. **(2 hours)**
8. Implement Greedy algorithm for Budgeted Influence Maximization and Fairness Influence Maximization. **(2 hours)**
9. Implement algorithms of Multiple Influence Maximization (MIM) to find the seed set. **(4 hours)**
10. Determine the potential fairness issues that may exist in advanced influence maximisation algorithms. **(4 hours)**

DSE308/DSC 402: Deep Learning [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSC308	**4**	**3**	**0**	**1**	Graduation	**Basic Knowledge of Machine Learning Algorithms**

Course Objectives:
This course introduces students to modern deep learning architectures and training methodologies used to solve real-world problems. Students will learn to design neural network models, understand optimization and regularization strategies, and work with major deep learning frameworks such as TensorFlow and PyTorch. Emphasis is placed on applying deep learning techniques to computer vision, natural language processing, and generative modeling.

Course Learning Outcomes:
On completing this course, students will be able to:

1. Describe feedforward neural networks, deep architectures, and their learning procedures.
2. Design and train single-layer and multi-layer deep neural networks while tuning hyperparameters effectively.
3. Analyze and evaluate the performance of deep learning models across vision, NLP, and generative tasks.

Syllabus:

Unit-I **(14 hours)**

Introduction: Historical context and motivation for deep learning; deep feedforward neural networks, regularizing a deep network, model exploration, and hyperparameter tuning. Convolution Neural Networks: Introduction to convolution neural networks: stacking, striding, and pooling, applications like image and text classification.

Unit-II **(17 hours)**

Introduction to Natural Language Processing (NLP), Traditional NLP Techniques, Sequence Modeling: Recurrent Nets: Unfolding computational graphs, recurrent neural networks (RNNs), bidirectional RNNs, encoder-decoder sequence to sequence architectures, deep recurrent networks. Large Language Models: Transformer Architecture, Pre-training and Fine-tuning Language Models, Ethical Considerations and Bias in Language Models, Applications of Large Language Models (Text Generation, Sentiment Analysis, Question Answering)

Unit-III **(10 hours)**

Autoencoders: Undercomplete autoencoders, regularized autoencoders, sparse autoencoders, denoising autoencoders, representational power, layer, size, and depth of autoencoders, stochastic encoders and decoders. Generative Adversarial Networks (GANs): Introduction to Generative Adversarial Networks, GAN Architectures (DCGAN, CycleGAN), Applications of GANs (Image Generation, Style Transfer)

Unit-IV **(4 hours)**

Structuring Machine Learning Projects: Orthogonalization, evaluation metrics, train/dev/test distributions, size of the dev and test sets, cleaning up incorrectly labelled data, bias and variance with mismatched data distributions, transfer learning, multi-task learning.

Readings:

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Heaton, J. (2015). Deep Learning and Neural Networks. Heaton Research Inc.
3. Hall, M. L. (2011). Deep Learning. VDM Verlag.
4. Deng, L., & Yu, D. (2009). Deep Learning: Methods and Applications. Now Publishers Inc.

List of Practicals:

1. Implement a deep neural network using PyTorch or TensorFlow on a simple dataset. Apply techniques for regularization, such as dropout and early stopping. Tune hyperparameters like learning rate, number of layers, and batch size. **(4 hours)**
2. Design and train a convolutional neural network on an image dataset. Apply stacking, striding, pooling, and normalization techniques. Evaluate model performance using accuracy and confusion matrix. **(4 hours)**
3. Preprocess text data, tokenize and vectorize inputs, and implement a CNN-based model for sentiment classification. **(2 hours)**
4. Implement a basic RNN to model sequences, such as character-level language modeling. Visualize unfolding of the computational graph and train the model on a small text corpus. **(4 hours)**
5. Build a sequence-to-sequence model using an encoder-decoder architecture with bidirectional RNNs for tasks like machine translation. Evaluate output quality using BLEU score or accuracy. **(2 hours)**
6. Use Transformers to load a pre-trained BERT or GPT model. Fine-tune it on a downstream task such as sentiment analysis or question answering datasets. **(4 hours)**
7. Analyze generated outputs from a large language model for biased or unethical responses. **(2 hours)**
8. Implement different types of autoencoders (undercomplete, sparse, denoising) on the MNIST dataset. Visualize reconstructed images and evaluate reconstruction error. (**4 hours)**
9. Build and train a simple GAN on MNIST or CelebA dataset. Generate synthetic images and compare visual quality with real images. Track generator and discriminator loss over time. (**2 hours)**
10. Use transfer learning with a pre-trained CNN (e.g., ResNet) to classify a small custom dataset. Discuss best practices for train/dev/test splits, evaluation metrics, and handling label noise. Experiment with multi-task learning if time permits. (**2 hours)**

SEMESTER – IV

DSC401: SOFTWARE ENGINEERING [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSC401	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:

This course aims to develop professional competence in software engineering by introducing engineering principles, process models, software design approaches, quality assurance techniques, and project management practices. Students will gain the ability to analyze, design, develop, test, and maintain reliable and high-quality software systems.

Course Learning Outcomes:

On completing this course, the student will be able to:

1. Understand and apply software process models, including agile development approaches.
2. Use modeling techniques and design principles to develop structured and component-based software solutions.
3. Apply quality assurance methods, reviews, metrics, and testing techniques to ensure software quality.
4. Demonstrate the ability to plan, manage, and monitor software projects using industry-standard practices.

Syllabus

Unit-I: Software Process & Agile Development (9 Hours)

Software process models; generic process model; prescriptive and specialized process models; process assessment and improvement, Agile development: concepts and principles of agility, cost of change, agile process models, Extreme Programming (XP), Scrum, feature-driven

development, and other agile approaches.

Unit-II: Requirements Engineering & Software Design (12 Hours)

Principles governing software engineering practice; core design and modeling principles, Requirements engineering: elicitation, analysis, negotiation, validation; building requirements models; use case modeling; UML diagrams; class-based modeling; flow-oriented modeling; behavioral modeling; patterns in requirements modeling, Design concepts: design process, abstraction, modularity, coupling and cohesion, design patterns, Component-level design: structure and behavior of components, class-based design, component-based development, traditional and object-oriented design approaches.

Unit-III: Software Quality Management & Testing (13 Hours)

Quality concepts: definition, software quality, quality dilemmas, achieving quality, Review techniques: review spectrum, informal reviews, formal technical reviews, defect amplification, review metrics, Software Quality Assurance (SQA): objectives, tasks, metrics, statistical quality assurance, software reliability, Testing strategies: validation, system testing, black-box and white-box testing, debugging practices, testing for conventional and object-oriented software, Software Configuration Management: SCM repository and process, Product metrics: metrics for requirements, design, testing, maintenance, and overall product quality.

Unit-IV: Software Project Management (11 Hours)

Project management concepts: management spectrum, people, product, process, and project dimensions, Process and project metrics: software measurement, metrics for quality, productivity, and estimation; metrics for small organizations, Project scheduling: task sets, task networks, scheduling techniques, Gantt charts, critical path method, earned value analysis, Risk management: reactive and proactive strategies; risk identification, analysis, projection, refinement, mitigation, monitoring, and planning; RMMM plan.

Readings:

1. Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach (7th Ed.). McGraw-Hill.
2. Sommerville, I. (2015). Software Engineering (10th Ed.). Pearson Education.
3. Mall, R. (2014). Fundamentals of Software Engineering (4th Ed.). PHI.
4. Jalote, P. (2005). An Integrated Approach to Software Engineering (3rd Ed.). Narosa Publishing House.

Additional References:

1. Aggarwal, K. K., & Singh, Y. (2008). *Software Engineering* (4th Ed.). New Age International Publishers.

2. Godbole, N. S. (2007). *Software Quality Assurance: Principles and Practice*. Alpha Science.

List of practicals:

You may choose a suitable programming language and define all necessary/relevant inputs/ problems.

1. Defining a software problem and establishing project scope. **(2 Hours)**
2. Preparing documentation for the selected project, including SRS, design documents, testing plan, configuration management plan, and risk management plan. **(8 Hours)**
3. Implementing the design or coding phase for the chosen software project. **(8 Hours)**
4. Conducting a structured software review (peer review / technical review). **(4 Hours)**
5. Applying software quality checks and relevant product metrics. **(2 Hours)**
6. Performing software testing using appropriate test strategies and documenting results. **(4 Hours)**
7. Executing software verification activities according to standards. **(2 Hours)**
8. Performing software estimation and proposing improvements based on metrics and testing outcomes. **(4 Hours)**

SBC 401:COMMUNICATION SKILLS [1-1-0]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
SBC401	2	1	1	0	Graduation	NIL

Course Objective:

This course aims to build essential communication competencies required in academic and professional settings. It focuses on strengthening verbal, non-verbal, written, and digital communication; enhancing listening, reading, and analytical skills; and developing clarity, coherence, and professionalism in both interpersonal and workplace communication. The course prepares students to confidently handle presentations, interviews, team communication, and the creation of structured technical documents while fostering ethical and culturally sensitive communication practices.

Course Learning Outcomes:

Upon successful completion of this course, students will:

1. Explain key principles of verbal, non-verbal, written, and digital communication and their role in academic and workplace contexts.
2. Apply effective communication techniques to produce clear, structured, and professional documents such as emails, reports, summaries, proposals, and résumés.
3. Analyze interpersonal and workplace communication situations, including interviews, presentations, group discussions, and team interactions, for clarity, intent, and effectiveness.
4. Evaluate communication practices—such as media texts, presentations, and written

documents—using criteria of coherence, accuracy, tone, ethics, and audience appropriateness.

5. Create persuasive and professional oral and written communication outputs, including presentations, structured arguments, and workplace documents that demonstrate confidence, clarity, and ethical awareness.

Syllabus:

Unit I: Foundations & Interpersonal Communication (8 hours)

Nature, scope, and process of communication; verbal, non-verbal, digital, and intercultural communication; major communication barriers and strategies to overcome them; essential listening and reading skills including comprehension, summarizing, inference, and note-making; fundamentals of oral presentations with focus on structure, tone, clarity, and audience awareness; storytelling for impact; professional spoken communication including interviews (technical and HR), group discussions, debates, conflict resolution, and teamwork in diverse workplaces.

Unit II: Written & Professional Communication (7 hours)

Principles of technical and academic writing; professional emails, notices, memos, minutes, and digital communication etiquette; summaries, analytical writing, and structured document development; informational and analytical reports, proposals, and basic project documentation; survey and questionnaire design; preparation of CVs, résumés, and Statements of Purpose; writing for media such as feature articles and press releases; ethical and leadership-oriented communication including bias, representation, and responsible messaging.

Readings:

1. Adler, R. B. & Proctor, R. F. (2013). Interplay: The process of interpersonal communication (16th ed.). Oxford University Press.
2. Bovee, C. L., & Thill, J. V. (2017). Business communication today (14th ed.). Pearson.
3. Raman, M., & Sharma, S. (2015). Technical communication: Principles and practice (3rd ed.). Oxford University Press.
4. McCarthy, P., & Hatcher, C. (2020). Presentation skills for students (4th ed.). Sage.

Tutorials

Case Study 1 (2 Hours) – Communication Breakdown in a Technical Interview

- Students analyse excerpts from technical and HR interview transcripts to identify unclear responses, weak articulation, and gaps in technical explanation. They rewrite improved responses and conduct a short mock interview.

Case Study 2 (2 Hours) – Miscommunication in a Multicultural Software Team

- A scenario involving an international development team shows misinterpreted emails, tone issues, and unclear task delegation. Students diagnose communication barriers and propose culturally sensitive revisions to improve clarity.

Case Study 3 (2 Hours) – Non-Verbal and Digital Cues in Remote Meetings

- Learners study a recorded or scripted virtual meeting with poor camera presence, weak tone, poor turn-taking, and digital distractions. They identify ineffective non-verbal cues and redesign the meeting communication plan.

Case Study 4 (2 Hours) – Analytical Summary from Technical Content

- Using a short technical document (e.g., system logs, analytics data, research abstract), students prepare a concise summary and inference-based note-making sheet demonstrating clarity and coherence.

Case Study 5 (2 Hours) – Workplace Document Reconstruction

- Students examine flawed workplace documents, such as emails, memos, notices, and meeting minutes, and correct issues related to tone, structure, coherence, ambiguity, and professionalism. Tasks include rewriting documents using standard templates.

Case Study 6 (2 Hours) – Report and Proposal Writing for a Mini Project

- A scenario on a small technical project (e.g., improving app usability) is provided. Students prepare a short informational or analytical report, including problem definition, findings, and a proposal section with recommendations.

Case Study 7 (2 Hours) – Résumé, SOP, and Job Role Alignment

- Students are given a job description and three sample résumés/SOPs. They analyse alignment, coherence, and tone; then rewrite a résumé/SOP tailored for the role while maintaining ethical communication practices.

Case Study 8 (1 Hour) – Media Bias and Ethical Communication

- Students examine two short news/tech-media snippets on the same topic (AI, cybersecurity, data privacy). They identify bias, tone shifts, and misrepresentation and suggest an ethically balanced rewrite.

List of DSEs for Semester IV

DSE401: Computer Graphics [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE401	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:

The course is designed to introduce students to the basic concepts and theory of computer graphics. The aim is to introduce students regarding various object rendering algorithms, projections, and latest concepts related to virtual reality.

Course Learning Outcomes:

On completing this course, the student will be able to:

1. acquire familiarity with the concepts and relevant mathematics of computer graphics.
2. ability to implement various algorithms to scan, convert the basic geometrical primitives, transformations, area filling, clipping.
3. describe the importance of viewing and projections.
4. ability to design basic graphics application programs.
5. familiarize with fundamentals of animation and virtual reality technologies.
6. be able to design applications that display graphic images to given specifications.

Syllabus:

UNIT I **(10 hours)**

Application Areas of Computer Graphics: Overview of graphics systems and devices; points and lines, line drawing algorithms, mid-point circle and ellipse algorithms; filled area primitives, polygon filling algorithms; curve generation: Bezier and B-Spline Curves.

UNIT II **(11 hours)**

2-D Geometrical Transforms: Translation, scaling, rotation, reflection and shear transformations composite transforms, transformations between coordinate systems; 2-D Viewing: viewing pipeline, viewing coordinate reference frame, window to viewport coordinate transformation, viewing functions, Homogeneous coordinates and matrix representation; Line Clipping Algorithms: Cohen-Sutherland and Cyrus Beck Line Clipping Algorithms, Sutherland–Hodgeman polygon clipping algorithm; 3-D object representation: polygon surfaces, quadric surfaces, spline representation.

UNIT III **(13 hours)**

3-D Geometric Transformations: Translation, rotation, scaling, reflection and shear transformations, composite transformations, 3-D viewing: viewing pipeline, viewing coordinates, view volume, general projection transforms and clipping; Visible Surface; Detection Methods: Classification, back-face detection, depth-buffer; scanline, depth sorting, BSP-tree methods, area sub-division and octree methods illumination models and surface rendering methods: basic illumination models, polygon rendering methods computer animation: design of animation sequence, general computer animation functions key frame animation, animation sequence, motion control methods, morphing, mesh warping.

UNIT IV **(11 hours)**

Virtual Reality: Basic concepts, classical components of VR system, types of VR systems, three-dimensional position trackers, navigation and manipulation interfaces, gesture interfaces. input devices, graphical rendering pipeline, haptic rendering pipeline, OpenGI rendering pipeline; applications of virtual reality.

Readings:

1. Donald Hearn, Pauline Baker, Warren Carithers, Computer Graphics with Open GL, Pearson, 4th Edition, 2011.
2. R. K Maurya, Computer Graphics with Virtual Reality Systems, Wiley, 3rd Edition, 2018
3. P. Shirley, M. Ashikhmin and S. Marschner, Fundamentals of Computer Graphics, 3rd Edition, CRC Press, 2009.

List of practical:

You may choose a suitable programming language and define all necessary/relevant inputs/ problems.

1. Write a program to draw basic geometric shapes: lines, circles, rectangles, and ellipses. (**2 hours**)

2. Write a program to generate ellipses using the Midpoint method. (**2 hours**)

3. Write a program to implement the polygon filling using Flood-fill, Boundary-fill and Scan-line algorithms. (**4 hours**)

4. Write a program for the Polygon Clipping using the Sutherland-Hodgeman algorithm. (**2 hours**)

5. Write a program to implement Curve generation using B-spline and Bezier curves. (**2 hours**)

6. Write a menu-driven program of 2D transformation: Translation, Scaling, Rotation, Mirror Reflection, Shearing, and Window-Viewport. (**4 hours**)

7. Write a program to implement Line Clipping using the Cohen-Sutherland algorithm and the Bisection Method. (**2 hours**)

8. Write a program to implement the 3D geometric transformations: Translation, Scaling and rotation. (**2 hours**)

9. Write a program to create 3D Scenes and 3D Projections. (**2 hours**)

10. Write a program to implement texture mapping on a 3D model, and observe the changes in the rendered output. (**4 hours**)

11. Write a program to animate a fish, or write a program to create an interactive animation using any authoring tool. (**2 hours**)

12. Write a program to implement the Back face removal algorithms using the Depth-Buffer algorithm and the Scan-line algorithm. (**2 hours**)

DSE402: Compiler Design [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE402	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Learning Objectives

The basic objective of this course is to understand the basic principles of compiler design, its various constituent parts, algorithms, and data structures required to be used in the compiler. It also aims to understand the use of basic compiler-building tools.

Course Learning Outcomes

On successful completion of the course, the students will be able to:

1. Describe the concepts and different phases of compilation.
2. Represent language tokens using regular expressions and context free grammars.
3. Describe the working of lexical analyzers.
4. Understand the working of different types of parsers and parse a particular string.
5. Describe intermediate code representations using syntax trees and DAG's as well as use this knowledge to generate intermediate code in the form of three address code representations.
6. Apply optimization techniques to intermediate code and generate machine code for high level language program.
7. Use Lex and Yacc automated compiler generation tools.

Syllabus

Unit I: (10 Hours)

Introduction to Compilation and Lexical Analysis: Phases of a compiler, Role of a lexical analyzer, Specification and recognition of tokens, Symbol table management, Error reporting and recovery in lexical analysis, Regular expressions and regular definitions, Lexical Analyzer Generator – *Lex*

Unit II: (15 Hours)

Syntax Analysis (Parsing Techniques) Context-Free Grammars (CFGs), Elimination of left recursion and left factoring, Top-down parsing techniques – Recursive Descent and LL Parsers, Bottom-up parsing techniques –, Shift-Reduce, LR Parsers (SLR, Canonical LR, LALR), Yet Another Compiler Compiler (YACC)

Unit III: (12 Hours)

Syntax-Directed Translation and Intermediate Code Generation: Syntax Directed Definitions (SDDs), Evaluation orders for SDDs (S-attributed, L-attributed definitions),

Intermediate representations – Syntax Trees and Three-Address Code (TAC), Types and declarations, Translation of expressions, control flow statements (loops, conditionals), Type checking

Unit IV: (8 Hours)

Runtime Environment, Code Generation, and Optimization: Storage organization and runtime environment, Activation records and stack allocation, Issues in code generation, Design of a simple code generator, Principal sources of optimization, peephole optimisation techniques

Readings

1. Alfred, V. A., Monica, S. L., & Jeffrey, D. U. *Compilers principles, techniques & tools*. Pearson Education, 2007
2. Chattopadhyay, S.. Compiler design. PHI Learning Pvt. Ltd, 2022

List of Practicals:

1. Write a Lex program to count the number of tabs, spaces and punctuation characters in an input file. **(2 hours)**
2. Write a Lex program to count the number of uppercase, lowercase and numeric characters separately. **(2 hours)**
3. Write a Lex program that implements a Vigenère cipher (given a short key); encrypt the input text and print the cipher text. **(2 hours)**
4. Write a Lex program to find and print the top 5 longest identifiers in a source file (identifiers = letters followed by letters/digits/underscores). **(2 hours)**
5. Write a Lex program that extracts all string literals (delimited by double quotes) from a source file and prints them with their line numbers. **(2 hours)**
6. Write a Lex program to count all multi-line comments (`/* ... */`) and single-line comments starting with `//` in a C file. **(2 hours)**
7. Write a YACC program to parse arithmetic expressions with `+ - * / ^` and parentheses. **(2 hours)**
8. Write a YACC program to accept conditional statements of the form `if (E) S else S` and translate them into simple three-address code (TAC) with labels and gotos. **(4 hours)**
9. Write a YACC program that recognises strings of the form `a^n b^n c^n (n≥1)` and accepts/rejects them. **(4 hours)**
10. Write a YACC program that parses simple `for` loops `for (init; cond; incr) stmt` and produces TAC for loop entry, body and exit. **(4 hours)**
11. Write a program (C++/Python) that constructs a DFA for identifiers and tests a list of input strings, printing accept/reject. **(4 hours)**

DSE403: Cloud Computing [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE403	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:

This course aims to equip the students with parallel and distributed computing and cloud computing concepts. Students will learn about cloud computing's characteristics, benefits, and historical developments. They will learn cloud computing architecture, service models (IaaS, PaaS, SaaS), deployment models, and emerging paradigms like edge computing and mobile cloud computing.

Course Learning Outcomes :

Upon successful completion of this course, a student will be able to:

1. describe cloud computing's characteristics, benefits, and historical developments, including distributed systems and virtualisation.
2. compare and contrast cloud computing architectures, service models, and deployment models.
3. analyze cloud economics, address open challenges.
4. discuss emerging paradigms like edge computing and mobile cloud computing
5. develop a cloud comping application.

Syllabus:

Unit-I **(11 hours)**

Introduction: Introduction to parallel and distributed computing; cloud computing: characteristics and benefits; historical developments and evolution of cloud computing: distributed systems, virtualization, Web 2.0, service-oriented computing, utility computing.

Unit-II **(11 hours)**

Virtualization: Cloud computing reference model, characteristics of virtualized environments, taxonomy of virtualization techniques, virtualization and cloud computing, pros and cons of virtualization; technology examples: Xen: paravirtualization; VMware: full virtualization, Microsoft Hyper-V.

Unit-III **(12 hours)**

Cloud Computing Architecture and Service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS); Deployment models: Public, Private, Hybrid, Community; IaaS: Introduction to IaaS, Resource Virtualization i.e. Server, Storage and Network virtualization; PaaS: Introduction to PaaS, Cloud platform & Management of Computation and Storage; SaaS: Introduction to SaaS, Cloud Services, Web services, Web 2.0, Web OS; Case studies related to IaaS, PaaS and SaaS, Economics of the cloud.

Unit-IV **(11 hours)**

Current Topics: Open Challenges in Cloud Computing; Introduction to emerging computing paradigms and research challenges: edge computing, mobile cloud computing, fog computing,

etc.; Introduction to IoT cloud; study on simulators related to cloud computing and emerging computing paradigms.

Readings:

1. Buyya, R., Vecchiola, C., & Thamarai Selvi, S. *Mastering cloud computing*. McGraw Hill, 2013.
2. Sosinsky, B. *Cloud computing bible*. Wiley, 2010.
3. Hwang, K., Fox, G. C., & Dongarra, J. *Distributed and cloud computing: From parallel processing to the Internet of Things*. Morgan Kaufmann, 2011.

List of Practicals:

1. Exploring Distributed Systems : Install and run a simple distributed framework (e.g., Hadoop single-node) to understand parallelism. **(2 hours)**
2. Cloud Characteristics Demo : Use AWS/GCP free tier to demonstrate elasticity, scalability, and pay-per-use features. **(2 hours)**
3. Web 2.0 & Service-Oriented Computing : Build a basic REST API and consume it with a web client to show service orientation. **(2 hours)**
4. Utility Computing Case Study : Analyze Google Workspace or Microsoft 365 as examples of utility computing and document benefits. **(2 hours)**
5. Virtual Machine Setup : Install VMware/VirtualBox and configure a Linux VM to demonstrate virtualization basics. **(2 hours)**
6. Paravirtualization with Xen : Install Xen and run a guest OS to explore paravirtualization concepts. **(2 hours)**
7. Hyper-V Demo : Configure Microsoft Hyper-V on Windows and create a virtual machine. **(2 hours)**
8. Virtualization Comparison : Benchmark performance differences between VMware, Xen, and Hyper-V using sample workloads. **(2 hours)**
9. IaaS Hands-On : Launch and manage a virtual server instance on AWS EC2 or Azure VM. **(2 hours)**
10. Storage Virtualization : Configure cloud storage (AWS S3, Google Cloud Storage) and test access control. **(2 hours)**
11. PaaS Deployment : Deploy a sample web app using Heroku or Google App Engine. **(2 hours)**
12. SaaS Exploration : Evaluate SaaS applications (Salesforce, Google Docs) and document features. **(2 hours)**
13. Deployment Models Case Study : Compare public, private, and hybrid cloud setups using OpenStack vs AWS. **(2 hours)**
14. Edge/Fog Computing Simulation : Use iFogSim/CloudSim to simulate fog computing scenarios. **(2 hours)**
15. IoT Cloud Integration : Connect an IoT device (Raspberry Pi/Arduino sensor) to AWS IoT Core or Google IoT Cloud and visualize data. **(2 hours)**

DSE404: Theory of Computation [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE404	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Learning Objectives

The course introduces the theoretical models of computation and their limitations. The students will develop skills to analyze automata and their computational power to recognize languages. The students will learn to apply the knowledge of Automata Theory, Grammar, and Turing machines to solving problems in language translation.

Course Learning Outcomes

On completion of this course, the student will be able to:

1. describe the mathematical models of machines and discuss their limitations.
2. prove equivalence of different computation models equivalent to Turing machines.
3. reason out why the computers cannot solve some apparently simple tasks, for example the Halting problem.
4. design and implement a parser for a context-free grammar. .

Syllabus

Unit I: Foundations of Formal Languages and Automata (10 Hours)

Introduction to formal languages, Alphabets, strings, and languages, Operations on strings and languages, Deterministic and Non-deterministic Finite Automata (DFA, NFA), Regular expressions and equivalence with finite automata

Unit-II Regular Languages and Their Properties (10 Hours)

Regular languages and their properties, Closure properties of regular languages, Pumping Lemma for regular languages, Decision properties of regular languages, Applications of finite automata

Unit-III Context-Free Grammars and Pushdown Automata (13 Hours)

Context-Free Grammars (CFG), Parse trees and derivations, Ambiguity in grammars and languages, Pushdown Automata (PDA): deterministic and non-deterministic, Equivalence of CFGs and PDAs, Properties of Context-Free Languages (CFLs): normal forms, pumping lemma, closure, and decision properties

Unit-IV Turing Machines and Undecidability (12 Hours)

Turing Machines: definition, construction, and programming, Variants of Turing Machines and their equivalence, Recursive and recursively enumerable languages, Undecidable problem: Halting Problem

Readings:

1. J. E. Hopcroft, R. Motwani, and J. D. Ullman, **Introduction to Automata Theory,**

languages, and computation, 2016.
2. H.R. Lewis, C.H. Papadimitriou, C. Papadimitriou, **Elements of the Theory of Computation** (2nd ed.), Pearson Education, 2015
3. P. Linz, **Introduction to Automata Theory, Languages, and Computation**, Jones & Bartlett, 2016.

List of Practicals:

1. Write a program to simulate operations on strings and languages (concatenation, reversal, substring, prefix, suffix). **(2 Hours)**
2. Construct a DFA for a given regular language and test string acceptance using simulation (e.g., JFLAP or Python). **(2 Hours)**
3. Construct an NFA for a given regular language and convert it into an equivalent DFA.
4. Design a regular expression for a given language and verify its equivalence with a finite automaton using a simulator. **(2 Hours)**
5. Implement a program to minimize a DFA using the state equivalence method. **(2 Hours)**
6. Demonstrate closure properties of regular languages (union, concatenation, Kleene star) using examples and simulation. **(2 Hours)**
7. Apply the Pumping Lemma to prove that a given language is not regular (manual exercise + verification). **(2 Hours)**
8. Design and simulate a lexical analyzer using regular expressions (Lex/PLY/Python regex). **(2 Hours)**
9. Construct a Context-Free Grammar (CFG) for a given language and generate its parse tree. **(2 Hours)**
10. Show ambiguity in a grammar by deriving multiple parse trees for the same string. **(2 Hours)**
11. Design a Pushdown Automaton (PDA) to accept strings belonging to a simple context-free language (e.g., balanced parentheses). **(2 Hours)**
12. Convert a CFG to an equivalent PDA and verify acceptance using simulation (e.g., JFLAP). **(2 Hours)**
13. Construct a Turing Machine (TM) to perform a simple computation (e.g., binary increment, palindrome checking). **(2 Hours)**
14. Design and simulate a two-tape or multi-track Turing Machine and verify its equivalence to a single-tape TM. **(4 Hours)**

DSE405: Wireless and Mobile Communications [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE405	**4**	**3**	**0**	**1**	Graduation	**NIL**

Course Objectives:

This course aims to provide a comprehensive understanding of wireless communication principles, including propagation mechanisms, fading effects, and multiple access techniques. It explores IEEE 802.11 WLANs and cellular communication concepts such as frequency reuse, handoff strategies, and interference management. It provides insights into ad-hoc and sensor networks, regulatory aspects, and network design challenges, preparing students to apply these concepts in real-world scenarios and emerging wireless technologies.

Course Learning Outcomes:

On completion of the course, students will be able to:

1. analyze and simulate wireless communication scenarios to observe propagation effects, fading impacts, and the performance of multiple access techniques.
2. understand and evaluate cellular network design by analyzing frequency reuse, handoff strategies, interference management, and capacity enhancement techniques
3. compare GSM, CDMA, LTE, WiFi, and WiMAX while understanding regulatory and spectrum management
4. analyze the architecture and key factors influencing the deployment of wireless sensor networks (WSNs) and vehicular ad-hoc networks (VANETs)

Syllabus:

Unit-I **(10 hours)**

Introduction: Wireless communication systems, history and evolution (1G to 5G and beyond), Radio wave propagation mechanisms (reflection, diffraction, scattering), large-scale path loss models, small-scale fading and multipath propagation, doppler effect and delay spread, spread spectrum techniques, multiple access and duplexing techniques

Unit-II **(10 hours)**

Cellular mobile communication: frequency re-use and channel assignment strategies, handoff strategies, types, priorities, practical considerations, interference and system capacity, co-channel and adjacent channel interference, power control measures, grade of service, definition, standards, coverage and capacity enhancement in cellular network, cell splitting, sectoring, microcells

Unit-III **(10 hours)**

Wireless systems and standards: Global System for Mobile (GSM) - services and features, system architecture, radio sub-system, channel types (traffic and control), frame structure, signal processing, CDMA standards-frequency and channel specifications, Forward and Reverse CDMA channels, WiFi, WiMAX, UMB, UMTS, LTE, and recent trends, Regulatory issues (spectrum allocation, spectrum pricing, licensing, tariff regulation and interconnection issues)

Unit-IV **(15 hours)**

IEEE 802.11(Wireless LAN) Standards: IEEE 802.11 standards overview (802.11a/b/g/n), physical layer, medium access control layer-CSMA/CA mechanism, RTS/CTS mechanism for collision avoidance, hidden node and exposed node problems, QoS enhancements in IEEE 802.11, future trends in IEEE 802.11, IEEE 802.11 and IEEE 802.3 (Ethernet) interoperability, IEEE 802.11 in cellular networks (5G and Wi-Fi Offloading); Ad-hoc networks and sensor networks: introduction, challenges and issues, AODV, DSR, DSDV routing protocols; architecture and factors influencing the sensor network design; concept of MANET and VANET

Readings:

1. Rappaport, Theodore S. *Wireless communications: principles and practice*. Cambridge University Press, 2024.
2. Murthy, C. Siva Ram, and B. S. Manoj. *Ad hoc wireless networks: Architectures and protocols*. Pearson education, 2008.
3. Stallings, William. *Wireless communications & networks*. Pearson Education India, 2009.
4. Goldsmith, Andrea. *Wireless communications*. Cambridge university press, 2013.
5. Garg, Vijay. *Wireless communications & networking*. Elsevier, 2010.
6. Carlos,M.d. and Agrawal, D P. *Ad Hoc and Sensor Networks: Theory and Applications*, World scientific publishing company, 2011.

List of Practical's

1. Write a program to: **(4 hours)**
   a) determine the free-space path loss and the power received.
   b) endorse the statement: "Free Space Attenuation increases by 6 dB whenever the length of the path is doubled. Similarly, as frequency is doubled, free space attenuation also increases by 6 dB".
   c) plot a chart between PL (dB) and distance (Km) with varying frequencies.

2. Write a program to: **(4 hours)**
   (a) determine the two-ray path loss and the power received.
   (b) Endorse the statement: "The total attenuation increases by 12 dB when the separation distance is doubled" in the two-ray model.
   (c) plot a chart between PL(dB) and distance (Km) with varying frequencies for the two ray path loss model.
   (d) Further, plot a chart between PL(dB) and distance (Km) comparing the free space path loss model and two-ray model with varying frequencies.

3. Write a program to generate a PN (Pseudo-Noise) sequence using a Linear Feedback Shift Register (LFSR). **(2 hours)**

4. Write a program to: **(4 hours)**

a) Calculate the received power at a distance d using a log-normal shadowing model for both indoor and outdoor environments.
b) plot a graph for Log normal shadowing model between distance and path loss/received power
    i. varying path loss exponent
    ii. varying standard deviation of shadowing parameters.
c) Compare friss-space path loss model and log normal shadowing model

5. Implement a function to create a hexagonal grid with a specified cluster size and cell size. Visualise the hexagonal grid using matplotlib. **(2 hours)**

6. Simulate a cellular network with a given number of base stations (BS) and mobile users randomly distributed in the hexagonal cells. Assign frequencies to each base station. Implement a function to calculate the signal strength at each mobile user's location based on the distance from the serving base station. Consider the free space path loss model for signal propagation. **(4 hours)**

(a)Plot a heatmap or contour plot of signal strength across the cellular network.

(b) Observe how signal strength varies within and between cells.

7. Capture WiFiTraffic on your device using the tool Wireshark. Analyse the interframe gaps for DIFS, SIFS and EIFS. Further, for each data frame, identify the corresponding ACK frame. Check whether the ACK frame is immediately following the data frame and has the SIFS timing gap. If not, determine if there is any delay and discuss possible reasons. **(2 hours)**

8. Write a Python script to simulate cell splitting. Divide some existing cells into smaller cells to demonstrate how this process increases system capacity. **(2 hours)**

9. Simulate a wireless network with three nodes (A, B, C) where A and C are hidden terminals. Implement the RTS/CTS protocol to prevent collisions at B when both A and C attempt to send data simultaneously. **(4 hours)**

10. Capture Wi-Fi frames using Wireshark on your machine's Wi-Fi interface and observe basic WLAN information. Filter for beacon frames and identify the BSSID and SSID, then find and analyze at least one Data frame and one ACK frame. Note the packet's source and destination MAC addresses. **(2 hours)**

DSE406: Advance Classification Methods [3-0-1]

Course Code	Credits	Credit distribution of the course		Pre-requisite of the course (if any)

		Lecture	Tutorial	Practical/Practice	Eligibility Criteria	
DSE406	**4**	**3**	**0**	**1**	Graduation	**Basic understanding of machine learning concepts, and familiarity with Python programming.**

Course Objectives

By the end of this course, students will be able to apply various classification techniques and evaluate model performance using appropriate metrics. They will also gain expertise in advanced topics such as ensemble learning, semi-supervised and self-supervised methods, zero-shot and few-shot learning, and improving model robustness against adversarial attacks.

Course Outcomes

On completion of this course, students will be able to:

1. explain the fundamental and advanced classification techniques.
2. apply various binary, multi-class, and multi-label classification methods.
3. analyze and compare classification models using appropriate evaluation metrics.
4. design and optimize classification models for real-world applications.

Syllabus

Unit-I **(16 Hours)**

Review of classification problems and algorithms, Evaluation metrics, Handling imbalanced data, Cross-validation, and Model selection. Ensemble learning for classification: Weak learners and Strong learners, Model creation, Model combination; Bagging and Boosting; Random Forest; Combination Strategies, Meta-learning; Diversity in ensemble; Ensemble pruning.

Unit-II **(12 hours)**

Multi-class classification problem; Aggregation-based approach: One-versus-all, One-versus-one, Directed Acyclic Graph approach, Tree-based approach, Error-correcting output codes; multi-class SVMs; Class Imbalance in Multi-class Settings.

Unit-III **(12 hours)**

Multi-label Classification, Problem Transformation approach: Binary Relevance, Classifier Chains, Label Powerset, Algorithm Adaptation Methods; Neural Network Approaches for Multi-label Classification, Evaluation metrics for multi-label classification.

Unit-IV **(5 hours)**

Semi-supervised and Self-supervised Classification Methods, Zero-shot and Few-shot Learning, Adversarial Attacks and Robustness in Classification,

Readings

1. Zhou, Zhi-Hua. *Machine learning*. Springer nature, 2021.

2. Mohri, Mehryar, A. Rostamizadeh, and A. Talwalkar. Foundations of machine learning. The MIT Press. 2012
3. Charte, Francisco, Antonio J. Rivera, and María J. Del Jesus. *Multilabel classification: problem analysis, metrics and techniques*. Springer International Publishing, 2016.
4. Ian Goodfellow, Bengio, Yoshua, and Aaron Courville. *Deep learning*. MIT press, 2017.
5. Research papers on emerging classification techniques.

List of Practical's:

1. Implement basic classification models such as Logistic Regression, Decision Trees and SVM on benchmark datasets. Evaluate the performance of these classifiers using metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix. Perform k-fold cross-validation and interpret the results. **(2 hours)**
2. Use a dataset with imbalanced classes and then apply techniques like SMOTE, random oversampling/under sampling, and evaluate their impact on model performance using appropriate metrics like precision-recall and balanced accuracy. **(4 hours)**
3. Implement ensemble models, including Random Forest (bagging) and AdaBoost/Gradient Boosting (boosting), using scikit-learn or XGBoost. Compare performance, analyse feature importance, and visualise ensemble effects**. (4 hours)**
4. Create a diverse ensemble of traditional machine learning classifiers such as Logistic Regression, SVM and Decision Tree using stacking or voting. Analyze diversity among models and perform ensemble pruning based on accuracy or diversity criteria. **(4 hours)**
5. Use multi-class datasets and implement one-vs-all and one-vs-one strategies. Train binary classifiers accordingly and compare both strategies in terms of performance and scalability. **(2 hours)**
6. Implement multi-class classification using tree-based methods and error-correcting output codes (ECOC). Evaluate performance and interpret how each approach handles class boundaries. **(4 hours)**
7. Train and evaluate multi-class SVMs using one-vs-all or one-vs-one schemes. Use imbalanced versions of multi-class datasets to analyze the effect of class imbalance and apply weighted loss or resampling. (**2 hours)**
8. Implement multi-label classification approaches such as Binary Relevance, Classifier Chains, and Label Powerset using scikit-multilearn or similar libraries. Compare performance using hamming loss, precision, recall, and subset accuracy**. (4 hours)**
9. Build a neural network using PyTorch or similarly libraries for multi-label classification with appropriate activation function. Train the model on a multi-label dataset and evaluate using multi-label metrics. (**2 hours)**
10. Explore few-shot classification using pre-trained models and implement semi-supervised learning on a partially labelled dataset to improve performance with limited annotations. **(2 hours)**

DSE407: Incident Response and Threat Hunting [3-0-1]

Course In Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical /Practice		
DSE407	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

This course enables students to understand the modern cyber threat landscape and the principles of risk management. Students will learn to apply industry-standard frameworks, such as MITRE ATT&CK and NIST, to execute proactive, hypothesis-driven threat hunts.Students will also master the complete incident response lifecycle, from initial triage and containment to final eradication and recovery.

Course Learning Outcome

Upon successful completion of this course, students will be able to:

1. Analyse cyber threats, assess organisational risk, and apply conceptual frameworks (e.g., Cyber Kill Chain, MITRE ATT&CK, Pyramid of Pain) to deconstruct adversary TTPs.
2. Formulate and execute proactive, hypothesis-driven threat hunts by analysing endpoint (EDR), network (traffic), and authentication (Event Log) data.
3. Apply the complete NIST Incident Response lifecycle, including the triage of security events, selection of containment strategies, and execution of eradication and recovery plans.
4. Conduct post-incident reviews to identify lessons learned and integrate findings into a continuous monitoring strategy aligned with the NIST Cybersecurity Framework (CSF).

Syllabus:

Unit I: (8 hours)

Introduction to Threat Hunting and Risk Management: Threat landscape, attacker motivation, common attack methods, anatomy of an attack, indicators of compromise (IoC) and indicators of attack (IoA), understanding Advanced Persistent Threats (APT) and Tactics, Techniques and Procedures (TTP), relationship between vulnerabilities, threats and risk, measuring risk severity and risk assessment

Unit II: (10 hours)

Frameworks for analysis: The pyramid of pain, Cyber Kill Chain, diamond model of intrusion detection, MITRE ATT&CK for Mapping TTP, selection of the right model

Unit III (14 hours)

Proactive threat hunting: Shifting from reactive to proactive hunting, Alert-driven vs Hypothesis-driven analysis, Threat Hunting methodologies: Investigation based on known indicators of compromise or Indicators of Attack (IoA), Hypothesis-driven hunting, Anomaly-based/Statistical hunting, Reference models for Hunting: Sqrrl, TaHiti, PEAK, F3EAD, Data-driven hunting: Endpoint Detection and Response(EDR), Event log analysis: understanding event logs, account-related logs, auditing system configuration changes, Lateral Movement Analysis: Server message block, Kerberos attack, scheduled tasks, identification of security events

Unit IV **(13 hours)**

Incident Response and Recovery:introduction to NIST Incident Response Frameworks (NIST 800-61), building an incident response plan, SOC workflows, Triage fundamentals, event identification, escalation, Containment Fundamentals, Strategy Selection of Containment Eradication, Recovery and post-incident review: Removing the attacker's artefacts, vulnerability scanning, restoring systems via backup, Continuous monitoring strategy, incorporating continuous monitoring into the NIST CSF environment

References:

1. Anson, Steve,, *Applied Incident Response*, 2020 John Wiley & Sons, Inc.
2. Eric C. Thompson, *Cybersecurity Incident Response: How to contain, eradicate and recover from incidents*, 2018, Apress
3. Scott J. Roberts ; Rebekah Brown, *Intelligence Driven Incident Response: Outwitting the adversary*, O'Reilly Media Inc., 2017
4. The MITRE Corporation, MITRE ATT&CK framework, https://attack.mitre.org/

List of Practicals:

1. Choose one of the following APT groups: APT28 (often referred to as "Fancy Bear") and APT41 ("Wicked Panda" or "Double Dragon"). Compare these two prominent threat actors and make a document containing the details, such as group name, suspected country of origin, primary motivation, common targets, common TTPs and notable examples of their activities **(2 hours)**
2. Download Process Explorer and open Notepad through the command prompt. Take a screenshot of your Process Explorer. What does it show? How this technique is used to find malicious child processes **(2 hours)**
3. Install the SIEM (e.g. Splunk or Wazuh), Windows VM(Victim) and Kali Linux VM (Attacker). Run a simple command on the Windows VM and find the corresponding process creation event in the installed SIEM. Explore the SIEM interface. **(4 hours)**
4. Read any incident report. Identify the stages of the Cyber Kill Chain present in the report. Similarly, use the MITRE ATT&CK website to find the TTPs described in the report. Identify IoC from the report and place them on a diagram of the pyramid of pain. **(8 hours)**
5. A lab exercise based on the identification of lateral movement **(8 hours)**
6. A lab exercise based on the containment and eradication phase of the NIST IR lifecycle **(2 hours)**
7. A lab exercise based on restoring the system and improving defence. **(2 hours)**

Techniques of Report Writing [1-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
Skill	**2**	**1**	**0**	**1**	Graduation	**NIL**

Course Objectives:

The student learns professional-level report writing skills. Students will learn the structure, format, and style of various types of reports. They can also get proficiency in data interpretation, visual presentation, and documentation.

Course Learning Outcomes:

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

1. Understand the principles of effective technical and professional writing.
2. Prepare different types of reports (informational, analytical, technical, project, research).
3. Use visual aids (charts, tables, figures) appropriately.
4. Edit and proofread for clarity and professionalism.
5. Prepare digital reports using modern tools (MS Word, LaTeX, Google Docs).

Syllabus:

Unit-I **(7 hours)**

Foundations: Purpose of reports, Types of reports, Structure of reports: Title page, abstract, TOC, introduction, body, conclusion, recommendations, references; Difference between essays, articles, and reports.

Writing Techniques and Style: Paragraph structure and flow, Writing effective introductions and conclusions, Use of numbered sections, headings, and subheadings, integrating quotes, paraphrasing, and citing sources, and avoiding plagiarism.

Unit-II **(8 hours)**

Data, and Visual Representation: Data interpretation and summarisation, Use of tables, graphs, charts, diagrams, incorporating figures and captions, Formatting tools: MS Word styles, templates, referencing tools; Basics of LaTeX for structured reports.

Editing and Proofreading: Editing techniques: macro-editing, micro-editing, Common writing errors (grammar, punctuation, redundancy, logical gaps), Readability improvement techniques, Peer review and collaborative editing, Use of digital tools: Grammarly, Google Docs comments.

Readings:

1. Pal, Rajendra & Korlahalli, J.S., *Essentials of Business Communication.*
2. Raman, Meenakshi & Sharma, Sangeeta, *Technical Communication.*
3. Lesikar, R.V. *Report Writing for Business*.
4. Pfeiffer, William S. *Technical Communication: A Practical Approach*.
5. Bailey, Stephen. *Academic Writing: A Handbook for International Students*.

List of practicals:

1. Prepare an outline for a technical or business report. **(2 hours)**
2. Draft a field visit / industrial visit report. **(2 hours)**
3. Write an analytical report using the provided data. **(2 hours)**

4. Design at least five charts using the data from lab exercise 3 and integrate them into a report. **(2 hours)**
5. Edit and proofread a report. **(2 hours)**
6. A lab exercise related to preparing a research/problem-solving report. **(2 hours)**
7. Create a report in LaTeX with sections, figures, and references. **(3 hours)**