MASTER OF COMPUTER SCIENCE PROGRAMME

2nd Year of 2 year M.Sc. (Computer Science) /1 year M.Sc. (Computer Science)

Postgraduate Curriculum Framework

under

NEP 2020

(w.e.f Academic Year 2026)

Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi
Delhi-110007

MSc Computer Science Programme 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 M.Sc. Computer Science programme is a four-semester program spanning two years. It has four structures - **MSc Computer Science with Coursework, MSc Computer Science with Coursework and Research, MSc Computer Science with Research and MSc Computer Science with Academic/Industry Project.** The first year (Semesters I and II) are common to all four structures. In the second year, the student can choose the structure he/she wishes to follow.

  To be eligible for the "Research" track, students must have studied **research methodology either in** the first year or in a UG program.

  The Programme objectives of the M. Sc Computer Science with coursework are to
  - Equip the students with comprehensive knowledge of the current trends in computer science.
  - Enable the students to follow the career path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career in the highly competitive **IT industry with development skills.**

  The Programme objectives of M. Sc Computer Science with Coursework and Research are to
  - Equip the students with comprehensive knowledge of the current trends in computer science and **introduce them to computer science research.**
  - Enable the students to follow the career path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career
    - in the highly competitive IT industry **with research/development skills**.
    - computer science research.

  The Programme objectives of M.Sc. Computer Science with Research are to
  - Equip the students with comprehensive knowledge of the current trends in **computer science research**.
  - Enable the students to follow the research path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career in
    - the highly competitive IT industry **with research**
    - computer science research.

  The Programme objectives of M.Sc. Computer Science with Academic/Industry Project are to
  - Equip the students with comprehensive knowledge of the current trends in

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

Project: In Structure 4, each student shall carry out a minor project in the third semester and a major project in the fourth semester of M.Sc. Computer Science program. The major 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 (minor/major) 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 MSc program in 2004, it has had minor projects in Semester III and one full semester (Semester IV) devoted to hands-on projects (research/industry/academic). The format has led to a strong alumni base enjoying high positions and packages in the IT industry along with some opting for research careers in top schools of the country and abroad.. Our alumni in industry 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 provision for 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 similar to Structure 3 except that the research component has been replaced with industry-relevant content in Semesters III and IV.

Semester III of 2year MSc/ Semester I of 1year MSc

Structure 1 Coursework

Semester III/I (Structure 1 Coursework)		
	Number of core courses	2

Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
DSC302	Natural Language Processing	3	0	1	**4**
SBC 301	Prompt Engineering	0	0	2	**2**
	Total credits in core course				10
	Number of DSE/GE courses				3**
	DSE5	3	0	1	**4**
	DSE6	3	0	1	**4**
	DSE7/GE3	3	0	1	**4**
	Total credits in DSE/GE				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 the number of seats.

List of DSE for Semester III/I (Structure 1 Coursework)		
Course Code	**Course Title**	**L-T-P**
DSE301	Link Prediction	3-0-1
DSE302	Recommender System	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	Data Engineering	3-0-1

Semester IV of 2year MSc/Semester II of 1year MSc

Semester IV/II (Structure 1 Coursework)					
	Number of core courses				2
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC401	Reinforcement learning	3	0	1	**4**
DSC402	Computer Vision	3	0	1	**4**
SBC 401	Communication Skills	0	0	2	**2**
	Total credits in core course				10
	Number of DSE/GE courses				3**
	DSE8	3	0	1	**4**
	DSE9	3	0	1	**4**

	DSE10/GE4	3	0	1	**4**
	Total credits in DSE/GE				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 the number of seats.

List of DSE for Semester IV/II (Structure 1 Coursework)		
Course Code	**Course Title**	**L-T-P**
DSE401	Edge Computing	3-0-1
DSE402	Mobile Application Development Technology	3-0-1
DSE403	Compiler Design	3-0-1
DSE404	Incident Response and Threat Hunting	3-0-1
DSE405	Systems Modeling and Simulation	3-0-1
DSE406	Graph Neural Networks	3-0-1
DSE407	Cryptography	3-0-1
DSE408	Computational Linguistics	3-0-1

Structure 2 Course Work and Research

Semester III/I (Structure 2 Coursework and Research)					
	Number of core courses				2
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
DSC302	Natural Language Processing	3	0	1	**4**
	Research-1	0	0	6	**6**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE5	3	0	1	**4**
	DSE6/GE3	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester III				**22**

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

List of DSE for Semester III/I (Structure 2 Coursework and Research)

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	Data Engineering	3-0-1

Semester IV/II (Structure 2 Coursework and Research)					
	Number of core courses	2			
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC401	Reinforcement learning	3	0	1	**4**
DSC402	Computer Vision	3	0	1	**4**
	Research-2	0	0	6	**6**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE7	3	0	1	**4**
	DSE8/GE4	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester IV				**22**

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

List of DSEs for Semester IV/II (Structure 2 Coursework and Research)		
Course Code	**Course Title**	**L-T-P**
DSE401	Edge Computing	3-0-1
DSE402	Mobile Application Development Technology	3-0-1
DSE403	Compiler Design	3-0-1
DSE404	Incident Response and Threat Hunting	3-0-1
DSE405	Systems Modelling and Simulation	3-0-1
DSE406	Graph Neural Networks	3-0-1
DSE407	Cryptography	3-0-1
DSE408	Computational Linguistics	3-0-1

Structure 3 Research

Semester III/I (Structure 3 Research)					
	Number of core courses				1
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
	Advanced Research Methodology	1	0	1	**2**
SBC301	Tools for Research	2	0	0	**2**
	Research-1	0	0	10	**10**
	Total credits in core course				18
	Number of DSE courses				1
	DSE5	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester III				**22**

List of DSE for Semester III/I (Structure 3 Research)		
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	Data Engineering	3-0-1
DSE309	Natural Language Processing	3-0-1

Semester IV/II (Structure 3 Research)					
	Number of core courses				0
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
	Techniques for Research Writing	1	0	1	**2**
	Research-2	0	0	16	**16**
	Total credits in core course				18
	Number of DSE courses				1

		Theory	Tutorial	Practical	**Total**
	DSE6	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester IV				**22**

List of DSE for Semester IV/II (Structure 3 Research)		
Course Code	**Course Title**	**L-T-P**
DSE401	Edge Computing	3-0-1
DSE402	Mobile Application Development Technology	3-0-1
DSE403	Compiler Design	3-0-1
DSE404	Incident Response and Threat Hunting	3-0-1
DSE405	Systems Modelling and Simulation	3-0-1
DSE406	Graph Neural Networks	3-0-1
DSE407	Cryptography	3-0-1
DSE408	Computational Linguistics	3-0-1

Structure 4 - Academic/Industry Project

Semester III/I (Structure 4 Academic/Industry Project)					
	Number of core courses				1
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
	Web Development	0	0	2	**2**
SBC 301	Prompt Engineering	0	0	2	**2**
	Minor Project	0	0	10	**10**
	Total credits in core course				18
	Number of DSE courses				1
	DSE5	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester III				**22**

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

	Major Project	0	0	16	**16**
	Techniques of Report Writing*	0	0	2	**2**
	Total credits in core course				18
	Number of DSE courses				1
		Theory	Tutorial	Practical	**Total**
	DSE 6*	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester IV				**22**

List of DSE for Semester IV/II (Structure 4 Academic/Industry Project)		
Course Code	**Course Title**	**L-T-P**
DSE401	Edge Computing	3-0-1
DSE402	Mobile Application Development Technology	3-0-1
DSE403	Compiler Design	3-0-1
DSE404	Incident Response and Threat Hunting	3-0-1
DSE405	Systems Modelling and Simulation	3-0-1
DSE406	Graph Neural Networks	3-0-1
DSE407	Cryptography	3-0-1
DSE408	Computational Linguistics	3-0-1

* Credits for the course may be earned by the students in Semester II/III

For **Structure 4 (**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: Optimization Methods [3-0-1]

		Credit distribution of the course		

Course title & Code	Credits	Lecture	Tutorial	Practical/Practice	Eligibility Criteria	Pre-requisite of the course (if any)
DSC 301	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

The course aims to impart comprehensive knowledge about operations research and optimization techniques. Its objectives are to help students understand the fundamental concepts of operations research, to discuss and analyze the need for various optimization techniques, to design and develop appropriate mathematical models, and to formulate and optimize real-world problems using these models.

Course Learning Outcomes:

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

1. Understand the operation research concepts
2. Analyse the need of optimisation techniques
3. Able to design and develop mathematical models for realm applications
4. Inculcate modeling skills necessary to describe and formulate optimization problems

UNIT-I: (10 Hours)

Introduction to the modelling approach of operation research, introduction to linear programming, formulating and solving linear programming models, graphical method, simplex methods. Transportation problems: mathematical formulation, balanced transportation problem, unbalanced transportation problem. assignment problem – mathematical formulation, Hungarian assignment method, travelling salesman problem.

Unit II (10 Hours)

Game theory: Formulation of two-person, zero-sum games and games with mixed strategies. Sequencing problems: processing of n jobs through 2 machines, processing of 2 jobs through m machines. Integer Programming: Gomory's method, branch and bound method. Network Scheduling: Basic terms, critical path methods (CPM), program evaluation and review technique (PERT).

Unit III (15 Hours)

Queuing Theory: Characteristics of queuing systems, Poisson process and exponential distribution, steady state M/M/1, M/M/C (Models I, II, IV, V), applications of queuing theory.

Inventory Control: Inventory Costs, economic order quantity, deterministic inventory problems, EOQ problems with no shortage, with shortage, production problem with no shortage, with shortage.

Unit IV **(10 Hours)**

Search methods: one dimensional minimisation methods, fibonacci and golden section methods. Non-linear programming: One dimensional minimization method: unconstrained optimization techniques, constrained optimization techniques, graphical solution, Kuhn-Tucker conditions, quadratic programming problems.

Readings:

1. Frederick S. Hillier, and Gerald J. Lieberman, *Introduction to Operations Research. McGraw-Hill, 2015.*
2. Hamdy A Taha, *Operations research: An introduction*. Pearson Education India, 2013.
3. S.S. Rao, *Engineering optimization: Theory and Practice*, New Age International, 3rd edition, 2013.
4. K. Deb, *Optimization for Engineering Design: Algorithms and Examples*, PHI, 2nd Edition, 2012.
5. J. S. Arora, *Introduction to Optimum Design,* Academic Press, 4th Edition, 2017.

List of Practicals:

1. Formulation and graphical solution of linear programming problems **(2 hours)**
2. Solving linear programming problems using simplex method **(2 hours)**
3. Transportation problems: balanced/unbalanced cases **(2 hours)**
4. Assignment problem: Hungarian method and travelling salesman problem **(2 hours)**
5. Game theory: solving two-person zero-sum games with pure and mixed strategies **(2 hours)**
6. Sequencing problems: n jobs through 2 machines, 2 jobs through m machines **(2 hours)**
7. Branch and bound method for integer programming **(2 hours)**
8. Network scheduling: CPM and PERT analysis **(2 hours)**
9. Queuing theory: simulation of M/M/1 queue with Poisson arrivals and exponential service **(2 hours)**
10. Analysis of M/M/C queues (Models I, II, IV, V) **(2 hours)**
11. Inventory control: EOQ problems with and without shortages **(2 hours)**
12. Replacement problems: deteriorating items, group vs individual replacement **(2 hours)**
13. Information theory: entropy, mutual information, uncertainty measures **(2 hours)**
14. Search methods: one-dimensional minimization using Fibonacci and golden section methods **(2 hours)**
15. Non-linear programming: unconstrained optimization techniques and graphical solutions **(2 hours)**

DSC302, DSE309: NATURAL LANGUAGE PROCESSING [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSC302 DSE309	**4**	**3**	**0**	**1**	**Graduation**	**Basic understanding of probability and statistics and familiarity with Python programming**

Course Objectives:

The course provides a rigorous introduction to the essential components of a Natural Language Processing (NLP) system. The students will learn various statistical, machine learning, and deep learning techniques in NLP and apply them to solve diverse language-related tasks.

Course Learning Outcomes:

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

1. Understand natural language basics, its structure, history, and levels of NLP analysis.
2. Convert raw text into numerical representations using preprocessing and vectorization techniques.
3. Build and apply language models, including n-gram, neural, and transformer-based models.
4. Use vector semantics and embeddings to represent and analyze word meaning.

Syllabus:

UNIT I **(10 Hours)**

Introduction: Definition and features of natural language, NLP needs and major applications areas, History of NLP, Turing test, Chinese room test, Chomsky grammar, IBM Georgetown experiment, ELIZA, Types of analysis involved: lexical, syntactic, semantic, discourse, and pragmatic.

Unit II **(10 Hours)**

Text-to-Vector: Need and role, Text preprocessing, Stemming, Lemmatization, Spelling variation, Edit distance, Character-level encoding, Word-level one-hot vectors, Bag-of-Words (BoW), Term Frequency (TF), Inverse Document Frequency (IDF), TF-IDF vectors, Application: Text-vector similarity, classification, clustering, retrieval, topic modelling.

Unit III: **(10 Hours)**

Language Modeling: Introduction to Language Models (LM), Predictive probability of word sequences, Types of LMs: statistical vs neural vs transformer-based, N-gram Language Models: Unigram, bigram, trigram models, Markov assumption, Maximum likelihood

estimation for n-grams, Language Model Evaluation, Perplexity, Smoothing, BackOff, Interpolation Techniques.

Unit IV: **(15 Hours)**

Vector Semantics: Introduction and motivation, Distributional hypothesis, Sparse vs dense vectors, Word-embeddings, Count-Based Models, TF-IDF, Pointwise Mutual Information (PMI), Positive PMI, Word Embeddings, word2vec, Continuous BoW, Skip-gram, Negative sampling, Global vector approach (GloVe), fastText, Vector arithmetic, Neural network based embeddings, Limitations of static embeddings, Contextual representations, Encoder-based LM vectors (BERT, RoBERTa), Decoder-based LM vectors (GPT), Sentence-level and token-level contextual embeddings.

Readings:

1. Jurafsky, D., & Martin, J. H. *Speech and language processing*. Pearson, 2000.
2. Eisenstein, J. *Introduction to natural language processing*. MIT Press, 2019.
3. Goldberg, Y. *Neural network methods in natural language processing*. Morgan & Claypool Publishers, 2017.
4. Brownlee, J. *Deep learning for natural language processing*. Machine Learning Mastery, 2017.

List of Practicals:

1. Use standard Python NLP libraries (such as NLTK and SpaCy) to perform fundamental text preprocessing operations, including tokenization, stemming, lemmatization, and stop-word removal.(**2 hours)**

2. Apply advanced text preprocessing techniques using regular expressions to handle tasks such as pattern matching, text removal, and substitution in practical real-world scenarios. (**2 hours)**

3. Transform textual data into Bag-of-Words (BoW) and other frequency-based vector representations, and implement classification algorithms such as logistic regression and Naïve Bayes to evaluate model performance. (**2 hours)**

4. Explore the principles of information retrieval by designing a query-based document retrieval system using vector representations of text. (**4 hours)**

5. Perform document clustering using text-vector representations and apply unsupervised learning methods such as K-means clustering and hierarchical clustering. (**4 hours)**

6. Implement and analyze basic statistical language models, including bigram and trigram models, for text generation and processing. (**4 hours)**

7. Enhance bigram, trigram, and 4-gram models by applying techniques such as smoothing, back-off, and interpolation to improve model accuracy and generalisation. (**4 hours)**

8. Develop a text classification system using vector semantics by training Word2Vec embeddings and evaluating their effectiveness in downstream classification tasks. (**4 hours)**

9. Design and develop an end-to-end natural language processing system using a real-life textual dataset. (**4 hours)**

Advanced Research Methodology (1-0-1)

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

Course Learning Objectives:

This second-level course in research methodology aims to advance the research skills of the students. The course aims to develop analytical skills for computer science research and the skills for technical writing and communicating the research for publication, while maintaining ethical practices.

Course Learning outcomes:

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

1. Conceptualize research problems in terms of research questions, hypotheses, and conceptual models.
2. Select and apply appropriate research designs for answering research questions that address the research questions.
3. Critically analyse research from the literature in terms of the appropriateness of the study design, data analysis, results, and interpretation.

Syllabus:

Unit I **(6 Hours)**

Research Design and Philosophy: Deeper understanding of research philosophies (positivism, interpretivism, etc.); Advanced research designs, Advanced searching tools and referencing tools, Publication ethics (plagiarism, copyright, patents), conceptualising research questions and hypotheses.

Unit II **(9 Hours)**

Advanced Data Analysis and Research Writing: Advanced quantitative and qualitative data analysis (grounded theory, case studies), Multivariate data analysis techniques, Testing of hypothesis, Levels of significance, confidence limits and intervals, Type I and Type II error, Technical writing, Code of Ethics in Research (COPE guidelines).

References:

1. Creswell, J. W., & Creswell, J. D. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, 2017.
2. Kothari, C. R., & Garg, G. *Research methodology: Methods and techniques*. New Age International Publishers, 2019.
3. Oshima, A., & Hogue, A. *Introduction to academic writing* (3rd ed.). Pearson Education, Inc., 2007.
4. Relevant study material from ACM, IEEE, Elsevier, and Springer digital libraries.

List of Practicals

Identify three research papers in consultation with the project supervisor, ensuring that the code for the paper is available. Read the papers carefully and perform the following exercises for each paper. (10 x 3 = 30 hours)

1. Prepare a well structured summary of the paper, clearly stating the research gaps identified by the author, strengths and weaknesses of the research in your opinion.
2. Execute the available code for the problem on the datasets mentioned in the paper and reproduce the results.
3. Create the plots and result tables shown in the papers, and write your detailed interpretation. This is expected to be same as authors', but if there is something that you have observed and which was not stated by the authors, please highlight it.
4. Identify the research gaps in the work, and add as future work in the summary

WEB DEVELOPMENT [0-0-2]

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

Course Objective:

The objective of this course is to provide hands-on experience in building interactive, structured, and dynamic webpages using HTML, CSS, JavaScript, jQuery, and JSON. Students will gain practical abilities in web content creation, styling, scripting, and event-driven programming.

Course Learning Outcomes:

Upon successful completion of this course, students will

1. understand the foundations of web development and webpage structure.

2. create well-designed webpages using HTML elements, semantic tags, tables, links, lists, images, and forms.
3. style webpages using CSS selectors, layout techniques, and class/id attributes.
4. apply JavaScript for DOM manipulation, event handling, and form validation.
5. implement jQuery functions for simplified scripting, UI behaviour, and event-driven actions.
6. use JSON for structured data representation in browser-based applications.

Syllabus:

Unit I: Foundations of Web Development (HTML + CSS)

Introduction to Web Development: Internet basics, websites, planning webpages, HTML Basics: Elements, attributes, headings, paragraphs, text formatting, Images, links, lists, tables, div elements, semantic tags, HTML Forms: input controls, buttons, text area, dropdown, radio, checkboxes, CSS Basics: inline/internal/external CSS, CSS Selectors: id, class, element selectors, structural elements, Page layout, backgrounds, borders, padding, margins, Styling tables, lists, images, and form components

Unit II: Web Scripting (JavaScript + jQuery + JSON)

JavaScript Fundamentals: variables, data types, operators, Conditional statements, functions, DOM access, Form validation using JavaScript, Event Handling, jQuery Basics: selectors, functions, document ready, jQuery events: blur, change, focus, click, dblClick, hover, submit, jQuery DOM manipulation & effects, JSON Basics: syntax, arrays, objects, parsing, stringifying

Readings:

1. Minnick, J. (2017). *Responsible web design with HTML5 and CSS* (9th ed.). Cengage Learning.
2. Nixon, R. (2021). *Learning PHP, MySQL & JavaScript with jQuery, CSS and HTML5* (6th ed.). O'Reilly Media.
3. Bayross, I. (2010; Reprint 2022). *Web-enabled commercial application development using HTML, DHTML, JavaScript, Perl CGI* (4th rev. ed.). BPB Publications.
4. Duckett, J. (2014). *JavaScript and jQuery: Interactive front-end web development*. Wiley.

List of Practicals

HTML + CSS Practicals

(Use CSS to style the webpage components, wherever possible)

1. Design a product showcase page containing product cards (image, name, description, price) arranged in a table or grid using HTML only. **(2 hours)**
2. Create a webpage with internal sections (About, Gallery, Achievements) and implement internal jumping links using anchor tags. **(2 hours)**
3. Create a homepage for the Department with a header, navigation bar, and footer using semantic tags. Include a short introduction paragraph and two images placed side-by-side. Use lists to display the faculty details (including the date of joining the department). Design a navigation menu with dropdown submenus that utilize a CSS hover effect. **(6**

hours)

4. Create an Alumni registration form with relevant fields such as name, batch, and current designation. Style a registration form with CSS to format labels, inputs, spacing, and button styles. **(4 hours)**
5. Design a webpage using HTML to display your course timetable. **(2 hours)**
6. Apply CSS to an existing HTML page to style headings, lists, paragraphs, backgrounds, margins, padding, and borders. **(2 hours)**
7. Create a 3-column layout using CSS (float/flexbox) for "News", "Events", and "Notices". **(2 hours)**
8. Design a navigation menu with dropdown submenus using CSS hover effects. **(2 hours)**

JavaScript Practicals

9. Write a JavaScript code to display a random image from a pool of event images whenever the Department page is refreshed. **(2 hours)**
10. Write a JavaScript code to display the years of experience (computed based on their date of joining) for each faculty in the Department webpage designed in exercise #3. **(2 hours)**
11. Add another field in the Alumni registration form designed in exercise#4 to collect the feedback (not more than 300 characters) from Alumni. Develop a character-count tool that displays a live count of characters typed into a text area. Also, implement JavaScript validation for this form (minimum length, no empty fields, valid date of birth).**(4 hours)**
12. Write JavaScript to change the background color of a text box on focus. **(2 hours)**
13. Create a light/dark theme toggle using JavaScript to switch CSS classes. **(2 hours)**
14. Detect and display the key pressed by the user using the keypress event. **(2 hours)**
15. Write a script to hide/show an image when a button is clicked. **(2 hours)**
16. Create a basic digital clock using JavaScript and update it every second. **(2 hours)**

jQuery + JSON Practicals

17. Write a script to dynamically add items to a list when a user enters text and clicks "Add Item". **(2 hours)**
18. Create a jQuery script to fade in and fade out images on button click. **(2 hours)**
19. Use jQuery to validate the Alumni registration form designed in exercise #4.**(2 hours)**
20. Filter list items using jQuery: user types text, matching items remain visible while others hide. **(2 hours)**

21. Demonstrate jQuery event handling (focus, blur, change, click, dblclick, submit) using a mini interactive form. **(2 hours)**
22. Use jQuery hover events to display additional information about an image. **(2 hours)**
23. Implement a JSON-based student data viewer: **(2 hours)**
    a. Store student data in JSON
    b. Display name, roll number, and department dynamically on the webpage
24. Parse a JSON array of faculty members and generate an HTML list of names and their qualifications. **(2 hours)**
25. Create a jQuery animation that moves a box from left to right across the webpage. **(2 hours)**
26. Create a jQuery-based color palette: clicking a color button changes the background color of a div.

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, 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.

SBC301: Tools for Research [2-0-0]

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

Course Objectives:

The course aims to familiarize students with a comprehensive set of digital tools and platforms essential for conducting high-quality academic research. It focuses on equipping learners with the ability to efficiently search, organize, and manage scholarly literature using professional discovery and reference management tools. Students will also gain proficiency in using software for data handling, analysis, visualization, and document preparation, enabling them to produce well-structured and ethically sound research outputs. Additionally, the course emphasizes research integrity, collaborative workflows, and effective dissemination techniques to help students navigate the complete research lifecycle with confidence and competence.

Course Learning Outcomes

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

1. identify and use major tools for discovering scholarly literature.

2. manage references using professional citation management tools.
3. use document preparation and research organization software.
4. apply tools for data cleaning, visualization, and analysis.
5. understand plagiarism, research ethics, and publication workflows.

Syllabus

1. Create an effective research problem statement based on a given topic. (3 hours)
2. Perform a structured literature search using Boolean operators across three academic databases. (2 hours)
3. Build a reference library in Mendeley/Zotero and generate citations in two different styles. (3 hours)
4. Prepare a LaTeX document on Overleaf containing tables, figures, equations, and a BibTeX bibliography. (3 hours)
5. Use a PDF annotation tool and a digital note-taking app to organize insights from three research papers. (3 hours)
6. Design an online survey using Google Forms with proper question types and logic. (3 hours)
7. Clean a given dataset and handle missing or inconsistent data. (2 hours)
8. Perform descriptive and inferential statistical analysis using a data analysis tool. (2 hours)
9. Create visual analytics using Power BI/ Matplotlib based on a provided dataset. (3 hours)
10. Use Jupyter Notebook and GitHub to create a reproducible research workflow. (3 hours)
11. Create a research presentation and poster integrating results, tables, and visualizations. (3 hours)

Readings

1. Booth, Wayne C., et al. The Craft of Research. University of Chicago Press, 2016.
2. Walliman, Nicholas. Your Research Project: Designing and Planning Your Work. Sage, 2011.
3. Creswell, John W., and J. David Creswell. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. Sage, 2018.
4. Kumar, Ranjit. Research Methodology: A Step-by-Step Guide for Beginners. Sage, 2019.

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 visualize nodes in a reduced feature space. (**4 hours**)
2. Implement Common Neighbors, 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 personalized 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 – Neighborhood-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 metric.

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, 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		
DSE304	**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 title & 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 System**

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 Disk) 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, 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	Basic knowledge of Programming language

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		
DSE 307	**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: Data 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		
DSE 308	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

This course provides a comprehensive introduction to data engineering, focusing on the design, implementation, and management of data pipelines for analytics and decision-making. Students will gain hands-on experience with ETL processes, orchestration frameworks, and cloud-based data engineering services. Emphasis is placed on scalable data processing and real-world applications.

Course Learning Outcomes

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

1. Understand fundamental concepts of data engineering, data lifecycle, and the role of ETL/ELT in modern data systems.
2. Design and implement ETL pipelines for batch and streaming data using industry-standard tools.
3. Develop, schedule, and monitor workflows using orchestration tools such as Apache Airflow.
4. Apply cloud-native data engineering solutions (AWS Glue, GCP Dataflow, Azure Data Factory) for scalable and reliable data pipelines.

Syllabus

Unit-I **(10 hours)**

Foundations of Data Engineering: Introduction to Data Engineering and its role in modern analytics, Data lifecycle and architecture: OLTP vs OLAP, Data Lakes vs Data Warehouses, Types of data: structured, semi-structured, unstructured, ETL vs ELT processes, Data quality, governance, and lineage.

Unit-II **(10 hours)**

ETL and Data Transformation: Fundamentals of ETL pipelines, Data ingestion methods: batch and streaming, Data cleaning, preprocessing, and validation techniques, Tools for ETL: Apache Spark, Talend, AWS Glue basics, Performance optimization and error handling

Unit-III **(12 hours)**

Data Pipeline and Orchestration: Introduction to workflow orchestration, Orchestration vs scheduling, Apache Airflow: DAGs, operators, sensors, XComs, Building and deploying ETL pipelines with Airflow, Monitoring, logging, and troubleshooting pipelines, Case study: End-to-end automated pipeline

Unit-IV **(13 hours)**

Cloud Data Engineering: Overview of cloud data services, AWS Glue: architecture, job scripts, crawlers, integration with S3 and Redshift, GCP Dataflow: batch and streaming pipelines with Apache Beam, Azure Data Factory: pipelines, triggers, monitoring, Comparative analysis of AWS, GCP, and Azure services, Scalability, reliability, and cost optimization in cloud data engineering

Readings

1. Kleppmann, Martin. *Designing Data-Intensive Applications*. O'Reilly Media, 2017.
2. Karau, Holden, et al. *Learning Spark: Lightning-Fast Data Analytics*. O'Reilly Media, 2020.
3. Bilgin, Bas P. *Data Engineering with Python*. Packt Publishing, 2021.
4. Housley, Marc Lamberti. *Data Pipelines with Apache Airflow*. Manning Publications, 2021.
5. Zaharia, Matei. *Learning Apache Spark 2*. Packt Publishing, 2017.

List of Practicals:

1. Implement a basic ETL pipeline to extract data from CSV, transform it, and load it into a SQL database. **(2 hours)**
2. Perform batch and streaming data ingestion using Apache Spark. **(2 hours)**
3. Clean and preprocess a real-world dataset (missing values, outliers, normalization). **(2 hours)**
4. Build an Airflow DAG to automate a multi-step ETL pipeline. **(4 hours)**
5. Configure Airflow connections to an external database (PostgreSQL/MySQL) and schedule periodic ingestion tasks. **(2 hours)**
6. Implement an AWS Glue job to move and transform data from S3 to Redshift. **(4 hours)**
7. Design a GCP Dataflow pipeline for real-time streaming sensor/log data. **(4 hours)**
8. Create an Azure Data Factory pipeline to integrate on-premises SQL Server with Azure

Blob Storage. **(4 hours)**

9. Compare the performance and scalability of ETL pipelines across AWS Glue, GCP Dataflow, and Azure Data Factory. **(2 hours)**

10. Mini-project: Design, implement, and present a scalable end-to-end cloud-based data pipeline. **(4 hours)**

SEMESTER - IV

DSC401: Reinforcement 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		
DSC401	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

This course develops foundational and practical skills in reinforcement learning. It covers Markov Decision Processes (MDPs), dynamic programming, Monte Carlo methods, temporal-difference learning, function approximation, and policy gradient approaches. Students will also gain hands-on experience implementing RL algorithms using standard Python libraries.

Course Learning Outcomes:

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

1. Understand reinforcement learning task formulations, principles, and key components.
2. Apply dynamic programming, Monte Carlo, and temporal difference methods to solve RL problems.
3. Implement common RL algorithms using Python and established RL libraries.
4. Analyze and develop policy gradient methods ranging from basic to advanced variants.

Unit I: **(10 hours)**

Foundations of Reinforcement Learning Historical background of RL; basics and definitions; how learning happens through interaction; key terminology and assumptions; core elements of RL: policy, reward, value functions, action-value functions, Bellman equations; exploration–exploitation; categories of RL methods. Introduction to Python RL ecosystems:

NumPy, OpenAI Gym, Keras/TensorFlow, and PyTorch; code standards and reproducibility practices.

Unit II: (11 hours)

Markov Decision Processes & Dynamic Programming: Markov property; Markov Reward Processes (MRP); definition and formulation of Markov Decision Processes (MDPs); states, actions, transitions, rewards; episodes and returns; value functions and optimality; Bellman optimality equations, Dynamic Programming for RL: iterative policy evaluation, policy improvement, policy iteration, value iteration, generalized policy iteration; asynchronous DP; limitations and efficiency considerations.

Unit III: (11 hours)

Monte Carlo & Temporal-Difference Learning Monte Carlo (MC) prediction: first-visit and every-visit methods; MC control; exploring starts; on-policy and off-policy MC; importance sampling (ordinary and weighted), Temporal Difference (TD) learning: TD(0), SARSA, Q-learning; on-policy vs off-policy TD; convergence aspects; overview of TD(1) and TD(λ); comparison of MC, TD, and DP.

Unit IV: (13 hours)

Function Approximation, Policy Gradients & Deep RL Function approximation: gradient MC, semi-gradient TD(0), TD with eligibility traces, after-states, least-squares TD, Policy gradient methods: policy parameterization, REINFORCE algorithm, variance reduction techniques, baselines, advantage function; actor–critic frameworks, Introduction to Deep Reinforcement Learning: neural network-based value and policy functions; overview of DRL applications and modern algorithms.

Readings

1. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
2. Bilgin, E. (2020). *Mastering reinforcement learning with python* (pp. 1-544). Packt Publishing.
3. Winder, P. (2020). *Reinforcement Learning: Industrial Applications of Intelligent Agents*. O'Reilly Media.
4. Zai, A., & Brown, B. (2020). *Deep Reinforcement Learning in Action*. Manning Publications.

List of Practicals:

1. Implementing Dynamic Programming Policy Evaluation. **(2 hours)**
2. Dynamic Programming Policy Iteration method. **(2 hours)**
3. Dynamic Programming Value Iteration method. **(2 hours)**
4. Monte Carlo Prediction implementation. **(4 hours)**
5. Off-Policy Monte Carlo Control using Importance Sampling. **(4 hours)**

6. Implementing SARSA (On-policy TD learning). **(4 hours)**
7. Implementing Q-learning (Off-policy TD learning). **(4 hours)**
8. Policy Gradient: REINFORCE algorithm implementation. **(4 hours)**
9. Actor–Critic method implementation. **(4 hours)**

DSE207, DSC402: Computer Vision [3-0-1]

Course title & Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE 207 DSC 402	**4**	**3**	**0**	**1**	**Graduation**	**foundation in mathematics, basic machine learning concepts, and programming**

Course Objectives:

The primary objective of this course is to introduce the fundamental principles, scope, and applications of computer vision and image processing, including image acquisition, representation, enhancement, and color processing. In addition, the course will help students to develop an understanding of early, mid, and high-level vision techniques. The students will be exposed to modern deep learning–based computer vision methods, particularly convolutional neural networks, and major architectures used in contemporary vision applications.

Course Learning Outcomes:

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

1. recognize and describe both the theoretical and practical aspects of computing with images.
2. understand image transformations, segmentation, feature detection, motion, matching, recognition, extraction, and categorization from images.
3. understand the geometric relationships between 2D images and the 3D world.
4. analyze the performance of deep networks within an image.

Syllabus:

Unit I: **(11 Hours)**

Overview of Computer Vision and Image Processing: Introduction of computer vision, definition, applications, and importance; historical context and evolution; image acquisition, image formats and representations, image enhancement, image filtering and convolution, RGB, HSV, and other color spaces, color manipulation in images.

Unit II: **(12 Hours)**

Early and Mid-level Vision: translation, rotation, scaling; affine and perspective transformations, thresholding, region-based segmentation, segmentation by clustering, segmentation by fitting a model, segmentation and fitting using probabilistic methods, feature detection and matching: points and patches, edges, lines, edge detection, corner detection, histogram equalization, adaptive histogram equalization, template matching, scale-invariant feature transform (SIFT).

Unit III: **(10 Hours)**

High Level Vision: Geometric methods: model-based vision, smooth surfaces and their outlines, aspect graphs, range data; probabilistic and inferential methods: recognition by relations between templates, geometric templates from spatial relations, application, image-based rendering.

Unit IV: **(12 Hours)**

Dense Motion Estimation: Translational alignment, parametric motion, spline-based motion, optical flow, layered motion.

Deep learning-based Computer Vision Techniques: Deep learning for computer vision, basics of neural networks, convolutional neural networks (CNNs), architecture of CNNs, training and fine-tuning CNNs, LeNet, AlexNet, GooleNet, VGG-Net, ResNet, comparative analysis of different architecture.

Readings:

1. Gonzalez, Rafael C. and Woods, Richard E., *Digital Image Processing*, 4th edition, Pearson Education, 2018.
2. Forsyth and Ponce, *Computer Vision: A Modern Approach*, 2nd edition, Pearson, 2015.
3. Szeliski, Richard, *Computer Vision: Algorithms and Applications*, Springer-Verlag, 2010.
4. Prince, Simon J. D., *Computer Vision: Models, Learning, and Inference*, 1st edition, Cambridge University Press, 2012.
5. Goodfellow, Ian, *Deep Learning (Adaptive Computation and Machine Learning series)*, The MIT Press, 2016

References:

1. Hartley, Richard and Zisserman, Andrew, *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

List of practical:

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

1. Implement a program for filtering and convolution techniques on the image. **(2 Hours)**

2. Implement a program to perform all colour space conversion and colour manipulation techniques in images. **(2 Hours)**

3. Write a program for all image translation, rotation, and scaling techniques. **(3 Hours)**

4. Write a program for all image segmentation techniques. **(4 Hours)**

5. Implement a program to perform all feature detection and matching techniques in images. **(3 Hours)**

6. Implement a program to apply histogram equalization to a colour image, and apply contrast stretching to the colour example image. Experiment with different parameter values to find an optimum for the visualization of this image. **(2 Hours)**

7. Write a program for the scale-invariant feature transform of an image. **(3 Hours)**

8. Implement a program for geometric methods of the image. **(3 Hours)**

9. Implement a program for dense motion estimation techniques on the image. **(2 Hours)**

10. Implement a program to apply different architectures of CNN to images to find the result of the image and investigate the usefulness of different architectures of CNN. **(6 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.

Techniques for Research 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		
	2	**1**	**0**	**1**	**Graduation**	**NIL**

Course Objective:

This course aims to equip students with techniques for effective research writing, including structuring academic documents, presenting ideas clearly. The course also develops students' abilities to produce high-quality scholarly manuscripts using modern tools for drafting, editing, and citation management, while maintaining academic integrity.

Course Learning Outcomes

After completing this course, students will be able to:

- Employ effective writing techniques, citation styles, and scholarly communication practices for structuring research content.
- Utilize and evaluate digital tools for reference management, plagiarism detection, and manuscript preparation, following ethical standards of academic and scientific research writing.
- Create well-structured research documents, such as proposals, literature reviews, and research papers, following academic and publication guidelines.

Syllabus

Unit I **(8 Hours)**

Foundations of Research Writing: Introduction to academic writing; structure of research papers; types of academic documents (proposal, thesis, review article); components of literature review; critical reading and note-making; paraphrasing, summarization, and avoiding plagiarism; citation styles; research ethics and integrity; tools for literature search; reference management tools grammar and readability tools; manuscript templates and formatting guidelines; writing research documents using LaTex.

Unit II **(7 Hours)**

Advanced Techniques & Publication Process: Designing an argument and flow in research writing; coherence, cohesion, and academic tone; presenting results; writing methods, results, and discussion sections; crafting abstracts and titles; responding to peer reviewers; journal selection and indexing; publication ethics (COPE guidelines); plagiarism checking tools; tools

for research visualization; writing using AI responsibly; preparing final manuscript and submission process.

References

1. C.R. Kothari, and G.Garg. Research Methodology: Methods and Techniques. New Age International Publishers, 2019.
2. R. Panneerselvam. Research methodology. PHI Learning Pvt. Ltd., 2014

Reading List

1. Rowe, (2011). Towards a greater diversity in writing styles, argumentative strategies and genre of manuscripts. European Journal of Information Systems, 20, pp. 491–495.
2. Peterson, T.C., Kleppner, S.R., and Botham, C.M. (2018). Ten simple rules for scientists: Improving your writing productivity. PLoS Computational Biology, 14, 10, e1006379.
3. Roig, M. (2010). Plagiarism and paraphrasing criteria of College and University professors. Ethics & Behavior, 11, 3, pp. 307-323.
4. Akin, F., Koray, O. and Tavukçu, K. (2015). How effective is critical reading in the understanding of Scientific Texts? Procedia – Social and Behavioral Sciences, 174, pp. 2444-2451.

List of Practicals (30 Hours)

1. Practice searching literature using Google Scholar, Scopus, and Web of Science. (2 Hours)
2. Create a reference library and cite sources in a sample document using tools introduced in the class. (2 Hours)
3. Prepare a structured mini-literature review based on selected research papers using LaTex, with a suitable title, abstract, and keywords. (6 Hours)
4. Write a research proposal in Latex, including suitable illustrations (tables, figures, with captions) and references, following academic conventions. (10 Hours)
5. Perform paraphrasing and summarization exercises and check originality using plagiarism-checking tools. (6 Hours)
6. Use the available grammar and readability tools to refine the research proposal. (4 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)**

List of DSEs for Semester IV

DSE 401: Edge 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		
DSE 401	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Learning Objectives:

The course enables students to describe key edge-computing concepts and technologies, including infrastructure components, deployment models, services, and real-world applications. Students will explain IoT service structures and devices used in edge-computing environments across various industries and Industrial IoT (IIoT) contexts. They will be able to analyse the differences, relationships, and interoperability between edge computing and other computing strategies.

Course Learning Outcomes:

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

1. describe fundamental concepts, architectures, and technologies of edge computing, including infrastructure components, service models, and real-world applications.
2. analyze an edge ecosystem to identify performance gaps and propose improvements, demonstrating understanding of edge-computing architectures and their major constituents.
3. explain IoT and Industrial IoT (IIoT) service structures, devices, and integration within edge computing environments across diverse industry contexts.
4. compare edge computing with other computing paradigms—such as cloud and fog computing—and evaluate their differences, complementarities, and interoperability.

Syllabus:

Unit I: **(10 Hours)**

Fundamentals of Edge Computing: Concepts of distributed systems and edge computing, edge computing architectures, IoT devices and edge computing in IoT gateways, edge computing interfaces, Edge computing vs. cloud computing vs. fog computing, and Key components of edge computing: Edge devices and sensors, and edge gateways and nodes.

Unit II: **(10 Hours)**

Edge Computing Networks: Network topologies and connectivity, edge servers and other devices, Sealing, Tiered network architecture, data transmissions in edge networks, Edge networks vs cloud networks vs fog networks, edge analytics: data types and types of data analytics, Core tenets of edge analytics, AI, ML, and Deep learning at the edge, Performance evaluations using queuing theory concepts.

Unit III: **(10 Hours)**

Network Advancements and Edge Computing: 5G technologies, mobile edge computing, network slicing, software-defined clouds, edge computing case studies: edge computing in Industry 4.0, IoT service architectures and devices, intelligent IoT applications, privacy and data security in edge computing: data confidentiality, security and privacy, identity management in edge computing, and security practice in edge computing.

Unit IV: **(15 Hours)**

Federating Edge Computing, Future Trends: Service-centric model, Multiple administrative domains, Deployment of services and applications in an edge environment, Edge as a service, Cloud as a service, Edge computing, and interoperability with cloud, Future trends and challenges of edge computing: Blockchain-enabled edge-supported Internet of Vehicles (IoV), Automation and robotics, Computer vision, Supply chain and Edge Computing, Roadmap for Edge Computing, Challenges of Edge Computing.

Readings:

1. Kumari, A., Sadasivam, G. S., Dharani, D., & Niranjanamurthy, M., *Edge computing fundamentals, advances and applications*. CRC Press, 2022.
2. Buyya, R., & Srirama, S. N. (2019). *Fog and edge computing: Principles and paradigms*. Wiley, 2019

List of Practicals:

1. Set up the Arduino IDE for the ESP8266-12 module and program it to configure sensors at the IoT edge setup. **(4 hours)**
2. Set up the communication parameters at the edge layer. **(4 hours)**
3. Implement any two communication protocols. **(2 hours)**
4. Deploy modules to a Windows IoT Edge device. **(2 hours)**
5. Create an IoT hub. **(4 hours)**
6. Register an IoT Edge device to your IoT hub. **(2 hours)**

7. Install and start the IoT Edge for Linux on Windows runtime on your device. **(4 hours)**
8. Remotely deploy a module to an IoT Edge device and send telemetry. **(4 hours)**
9. Deploy a module Manage your Azure IoT Edge device from the cloud to deploy a module that sends telemetry data to IoT Hub. **(4 hours)**

DSE402: Mobile App Development Technologies [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE 402	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

This course introduces the students to modern mobile application development frameworks, tools, and best practices. It focuses on designing, building, testing, and deploying mobile apps for Android and iOS platforms using cross-platform technologies. Students will gain hands-on experience with UI design, state management, API integration, mobile databases, and performance optimisation. Emphasis is placed on real-world mobile development workflows, scalability, and user-centric design.

Course Learning Outcomes

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

1. understand the architecture and components of mobile applications across Android, iOS, and cross-platform ecosystems.
2. design intuitive and responsive user interfaces using modern UI frameworks.
3. develop mobile applications using cross-platform technologies such as Flutter or React Native.
4. integrate REST APIs, authentication, and mobile databases into mobile apps.
5. apply state management, debugging, testing, and performance optimization techniques.
6. deploy mobile applications to the Google Play Store and Apple App Store.

Syllabus

Unit-I **(10 hours)**

Foundations of Mobile Application Development: Introduction to mobile ecosystems: Android, iOS, cross-platform; Native vs hybrid vs cross-platform apps; Mobile app architecture: MVC, MVVM, Clean Architecture; Development environments: Android Studio, Xcode, VS Code; Introduction to Flutter/React Native; UI/UX principles for mobile design; navigation paradigms; material design & human interface guidelines.

Unit-II **(10 hours)**

User Interface Development & State Management: Widgets/components, layouts, and styling; Forms, inputs; Building responsive UIs for multiple screen sizes; State management techniques:

Provider/Bloc/Redux/Stateful Widgets for Flutter; Hooks/Context/Redux for React Native; Handling user interactions; Localization & accessibility.

Unit-III **(12 hours)**

Backend Integration, Data Handling & Device Features: REST APIs, JSON parsing, and API-driven apps; Authentication: token-based, OAuth; Local databases: SQLite, Room, Hive, Shared Preferences; Offline-first applications; Using device sensors: camera, GPS, accelerometer; notifications, background services, permissions; error handling, logging, and security considerations.

Unit-IV **(13 hours)**

Advanced Topics, Testing & Deployment: Performance optimization, memory management, and debugging tools; Automated testing: unit testing, widget testing, integration testing; Continuous Integration/Continuous Deployment (CI/CD) for mobile; Packaging and signing apps; Publishing to Google Play Store and Apple App Store; Comparison of Flutter, React Native, Kotlin Multiplatform, SwiftUI; Industry trends: cross-platform evolution, foldable devices, wearable apps.

Readings

1. Bailey, T., & Biessek, A. *Flutter for beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart*. Packt Publishing Ltd., 2021.
2. Napoli, M. L. *Beginning Flutter: A hands-on guide to app development*. John Wiley & Sons, 2019.
3. Eisenman, B. *Learning React Native: Building native mobile apps with JavaScript*. O'Reilly Media, 2015.
4. Hardy, B., & Phillips, B. *Android programming: The Big Nerd Ranch guide*. Addison-Wesley Professional, 2013.
5. Mathias, M., Gallagher, J., & Ward, M. *Swift programming: The Big Nerd Ranch guide*. Pearson Technology Group, 2020

List of Practicals:

1. Set up the development environment (Android Studio/Xcode/Flutter/React Native). **(2 hours)**
2. Build a basic UI with screens, layouts, and navigation. **(4 hours)**
3. Implement forms, validation, and user input handling. **(2 hours)**
4. Create an app integrating a public REST API with JSON data. **(4 hours)**
5. Implement state management using Provider/Bloc or Context/Redux. **(4 hours)**
6. Implement the Add local storage using SQLite, Room, and Hive. **(4 hours)**
7. Implement the camera, location, and notifications using device features. **(4 hours)**

8. Develop the Mini-project: Develop and demonstrate a fully functional cross-platform mobile app with backend integration. **(6 hours)**

DSE 403: 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		
DSE403	**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)**

DSE404: Incident Response and Threat Hunting [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**	**Knowledge of computer networks and cyber security**

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)**

DSE 405: Systems Modeling and Simulation

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE 405	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Learning Objectives:

The objective of this course is to provide detailed understanding of Simulation concepts and their applications. This course will also help students to design and analyse simulation models with its application on real world problems to design and analyse simulation models with its application on real world problems.

Course Learning Outcomes:

Students completing this course will be able to:

1. Understand the Fundamentals of Simulation and Model Building.
2. Apply different methods of random number generation to generate discrete and continuous random variables.
3. Conduct analysis of simulation models.
4. Understand and apply principles of Monte Carlo simulation.
5. Decode case studies to discover various simulation applications.

Syllabus:

Unit I: (12 hours)

Fundamentals of Simulation and Modeling: Introduction to Simulation: Concept of model and model building, Concept and Terminologies of Simulation, Process of Simulation, Advantages and Limitations of Simulation, Classification of Simulation Models: physical and mathematical models, static and dynamic model, discrete and continuous models, deterministic and stochastic models. Application areas of Simulation.

Unit I1: (11 hours)

Random Number Generation: Properties of random numbers, generation of pseudo random numbers, techniques of generating discrete and continuous random variables: inverse transformation, direct transformation, rejection method, hazard rate method. Test for random numbers.

Unit III: (12 hours)

Design and Analysis of Simulation Models Data collection, identifying distributions with data, Parameter estimation, goodness-of-fit tests, selecting input models without data, Steady-

state simulation, terminating simulation, confidence interval estimation, and output analysis for steady-state simulation. Simulation run statistics, Replication of runs, elimination of initial bias.

Unit IV: **(10 Hours)**

Monte Carlo Methods, Simulation Tools, and Applications: Monte Carlo Simulation. Simulation Tools: General-Purpose Simulation Tools, Case Studies of different types of Simulation, General Sampling Methods, Generating Sample paths: Brownian and Geometric Brownian motions; Gaussian start rate methods; Square root diffusions; process with jumps.

Readings:

1. Ross, S., *Simulation,* Academic Press, 5th Edition 2012.
2. A.M. Law and W.D. Kelton, *Simulation and Modeling and analysis*, 5th Edition, 2015.
3. Evans, J. R., & Olson, D. L. *Introduction to simulation and risk analysis*. Prentice-Hall, Inc. 2nd Edition, 2001.
4. Geoffrey Gordon: *System Simulation*, 2nd Edition, 2002.
5. Frank L. Severance, *System Modeling And Simulation: An Introduction*, Wiley, 1st Edition, 2001.
6. J Banks, J. S. Carson II, B. L. Nelson, D. M. Nicol, 2010, Discrete Events System Simulation, 5th Edition, Prentice Hall.

List of Practicals:

1. Discrete vs continuous models (queue vs population growth) **(2 hours)**
2. Deterministic vs stochastic modeling comparison **(2 hours)**
3. Discrete-event single-server queue simulation (M/M/1) **(2 hours)**
4. Continuous simulation with logistic growth ODEs **(2 hours)**
5. Linear Congruential Generator pseudo-random number generation **(2 hours)**
6. Inverse transform sampling for exponential variables **(2 hours)**
7. Acceptance–rejection sampling for complex distributions **(2 hours)**
8. Hazard rate method for survival-time simulation **(2 hours)**
9. Randomness and distribution fit tests (Chi-square, KS) **(2 hours)**
10. Data collection and distribution fitting from real process **(2 hours)**
11. Parameter estimation and confidence interval analysis **(2 hours)**
12. Terminating vs steady-state simulation comparison **(2 hours)**
13. Replications and elimination of initial bias **(2 hours)**
14. Monte Carlo estimation of π and integrals **(2 hours)**
15. Sample paths of Brownian motion, Geometric Brownian motion, square-root diffusion and jump processes **(2 hours)**

DSE406: Graph Neural 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		
DSE406	**4**	**3**	**0**	**1**	**Graduation**	Deep Learning

Course Objectives:

This course aims to introduce students to graph-structured data and the principles of graph representation learning. It familiarises students with foundational and advanced Graph Neural Network (GNN) architectures, their training procedures, scalability complexities, interpretability techniques, and real-world applications across domains. Students will explore recent trends and challenges in graph-based deep learning.

Course Learning Outcomes

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

1. Understand graph structures and their mathematical representations.
2. Explain the theoretical foundations of graph neural networks.
3. Apply advanced GNN models to real-world applications.
4. Develop and evaluate GNNs on small- and large-scale graph datasets.
5. Apply and explain interpretability techniques to analyse GNN predictions.

Syllabus

Unit-I **(10 hours)**

Introduction to graph-structured data, graph terminology, types of graphs, adjacency matrices, graph features, real-world graph datasets; Classical graph algorithms, BFS, DFS, PageRank, centrality measures, similarity measures; Introduction to graph representation learning, motivation for graph embeddings, limitations of Euclidean assumptions; Shallow graph embedding methods including DeepWalk, Node2Vec, LINE; Inductive vs transductive learning, challenges in graph-based machine learning.

Unit-II **(12 hours)**

Introduction to Graph Neural Networks, message passing framework, spatial and spectral perspectives; Spectral GNNs, graph Fourier transform, spectral convolution, ChebNet, limitations of spectral methods; Spatial GNNs, Graph Convolutional Networks, GraphSAGE, Graph Attention Networks, propagation and aggregation mechanisms; Training GNNs, loss functions for node classification, link prediction and graph classification, over-smoothing and over-squashing, regularization approaches including DropEdge and normalization techniques.

Unit-III **(12 hours)**

Advanced GNN architectures including Graph Isomorphism Networks, relational GNNs, heterogeneous GNNs, recurrent and temporal GNNs; Graph pooling and readout methods, hierarchical pooling techniques, global pooling strategies; GNN applications in recommender systems, natural language processing, knowledge graphs, computer vision tasks such as scene graph modeling, bioinformatics and molecular graph learning; Graph-level prediction, graph classification, molecular property prediction, graph regression tasks.

Unit-IV **(11 hours)**

Scalable GNN training, neighbour sampling, layer sampling, mini-batch GNNs, distributed GNN frameworks; Explainable Graph Neural Networks, GNNExplainer, PGExplainer, subgraph-based interpretability; Evaluation protocols for node and graph tasks, metrics, ablation analysis, robustness and fairness considerations; Recent trends including Graph Transformers, self-supervised GNNs, graph foundation models, domain-specific GNN applications; Limitations of GNNs, oversmoothing, expressivity bottlenecks, reproducibility challenges, emerging research directions.

Readings

1. Hamilton, W. L. *Graph representation learning*. Morgan & Claypool Publishers, 2020.
2. Wu, L., Cui, P., & Pei, J. (Eds.). *Graph neural networks: Foundations, frontiers, and applications*. Springer, 2022.
3. Stamile, C., De Simone, V., & Bustreo, A. *Hands-on graph neural networks using Python*. Packt Publishing, 2021.

List of practicals:

1. Implement graph data structures using adjacency lists and adjacency matrices, and analyze basic graph properties using NetworkX **(3 hours)**
2. Load and explore real-world graph datasets, perform preprocessing, and prepare data for graph-based learning tasks **(3 hours)**
3. Implement shallow graph embedding methods such as DeepWalk or Node2Vec and visualize learned embeddings **(3 hours)**
4. Set up a GNN framework (PyTorch Geometric or DGL) and build a basic graph neural network model **(3 hours)**
5. Implement a Graph Convolutional Network (GCN) for semi-supervised node classification using citation network datasets **(3 hours)**
6. Build an inductive learning model using GraphSAGE with different aggregation functions and evaluate performance **(3 hours)**
7. Implement a Graph Attention Network (GAT) and analyze attention mechanisms in node classification tasks **(3 hours)**

8. Perform graph-level classification using graph pooling techniques such as TopKPool or SAGPool **(3 hours)**

9. Implement a link prediction model using GNNs with edge splitting and negative sampling **(3 hours)**

10. Apply GNN explainability techniques (GNNExplainer or similar) to interpret node and graph predictions **(3 hours)**

DSE 407: CRYPTOGRAPHY [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE 407	**4**	**3**	**0**	**1**	**Graduation**	**NIL**

Course Objectives:

This course aims to provide a comprehensive understanding of number theory, cryptographic primitives, algorithms and cryptographic techniques used in modern security systems. It covers the foundation of number theory, block ciphers, confidentiality and modular arithmetic, public key cryptography and authentication requirements, integrity checks and authentication algorithms.

Course Learning Outcomes:

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

1. implement symmetric and asymmetric encryption algorithms
2. describe the role and implementation of digital signatures.
3. provide security of the data over the network.
4. do research in the emerging areas of cryptography

Syllabus:

Unit-I **(10 hours)**

Introduction to Cryptography and Block Ciphers Introduction to cryptography - Conventional Encryption: Conventional encryption model - classical encryption techniques - substitution ciphers and transposition ciphers – cryptanalysis – steganography - stream and blockciphers - Modern Block Ciphers: Block ciphers principals - Shannon's theory of confusion and diffusion - fiestal structure - data encryption standard(DES) - strength of DES - differential and linear cryptanalysis of DES - block cipher modes of operations - triple DES – AES.

Unit-II **(10 hours)**

Confidentiality and Modular Arithmetic Confidentiality using conventional encryption - traffic confidentiality - key distribution - random number generation - Introduction to graph - ring and field - prime and relative prime numbers - modular arithmetic - Fermat's and Euler's theorem - primality testing - Euclid's Algorithm - Chinese Remainder theorem - discrete algorithms.

Unit-III **(13 hours)**

Public key cryptography and Authentication requirements: Principles of public key crypto systems - RSA algorithm - security of RSA - key management – Diffle-Hellman key exchange algorithm - Elliptic curve cryptography – Elgamel encryption - Message Authentication and Hash Function: Authentication requirements - authentication functions - message authentication code - hash functions - birthday attacks – security of hash functions and MACS.

Unit-IV **(12 hours)**

Integrity checks and Authentication algorithms: MD5 message digest algorithm - Secure hash algorithm (SHA) Digital Signatures: Digital Signatures - authentication protocols - digital signature standards (DSS) - proof of digital signature algorithm - Authentication Applications: Kerberos and X.509 - directory authentication service - electronic mail security-pretty good privacy (PGP) - S/MIME.

Readings:

1. Stallings, W. *Cryptography and network security: Principles and practice*. Pearson/PHI.
2. Trappe, W., & Washington, L. C. *Introduction to cryptography with coding theory*. Pearson.
3. Forouzan, B. A., & Mukhopadhyay, D. *Cryptography and network security* (3rd ed.). McGraw-Hill Education, 2015.
4. Elbirt, A. J. *Understanding and applying cryptography and data security*. CRC Press, Taylor & Francis Group, 2015.

List of Practicals:

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

1. Implement the substitution ciphers and transposition ciphers **(2 hours)**
2. Implement stream and blockciphers **(2 hours)**
3. Implement DES and do cryptanalysis of DES **(2 hours)**
4. Implement Chinese Remainder Theorem (CRT) and demonstrate its applications **(2 hours)**
5. Encryption and decryption of files using python's cryptography library to demonstrate symmetric encryption and asymmetric encryption algorithms discussed in the class **(8 hours)**
6. Computation of SHA-512 Hash and HMAC for file integrity verification using a Shared Secret Key. **(4 hours)**
7. Verification of file integrity and displaying "Integrity Verified" or "Integrity Compromised" Based on various hashing algorithms. **(4 hours)**
8. Implementation RSA based Digital Signature algorithms **(2 hours)**
9. Implementation Elliptic curve cryptography based Digital Signature algorithms **(4 hours)**

DSE 408: Computational Linguistics [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility Criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice		
DSE 408	**4**	**3**	**0**	**1**	**Graduation**	**Basic understanding of probability and statistics and familiarity with Python programming**

Course Objectives:

The course builds foundational and practical understanding of linguistic structures, formal grammars, parsing, corpus analysis, and semantic–pragmatic interpretation. It prepares students to design computational models that can effectively analyze, process, and generate natural language.

Course Learning Outcomes:

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

1. Understand major linguistic structures and their role in computational language processing.
2. Apply formal grammar models and parsing techniques to analyze sentence structure.
3. Apply corpus-based analysis to derive linguistic patterns and support data-driven NLP.
4. Evaluate semantic and pragmatic principles to interpret meaning and context in text.
5. Develop and assess basic computational models for effective language analysis and generation.

Syllabus:

UNIT I **(10 Hours)**

Introduction to Computational Linguistics: Nature and structure of natural language; goals of computational analysis; human–computer communication. Historical foundations: Turing Test, ELIZA, early MT systems, smart vs. solid solutions, verification of linguistic theories. Components of linguistic analysis: morphology, syntax, semantics, pragmatics, and their computational roles. Cognitive foundations of language: perception, recognition, context, reference, signs (symbols, indexicals, icons). Overview of SLIM theory: surface compositionality, time-linear processing, internal matching.

Unit II **(10 Hours)**

Formal Grammar: Introduction and role of parsing in CL, Categorial Grammar, Phrase Structure Grammar, adequacy and empirical motivation. Language hierarchies: generative

capacity, complexity classes, Phrase Structure (PS) vs Link-based Analysis (LA) grammar comparison. Parsing concepts: declarative vs procedural parsing, parser-grammar transparency, bottom-up/top-down parsing, convergence, robust parsing. Left-Associative Grammar (LAG): rule types, derivation order, time-linear analysis, LA hierarchy (A/B/C classes), ambiguity handling, linear-time parsing of natural language. Practical aspects: parser implementation, grammar testing on corpora, machine translation basics.

Unit III: (10 Hours)

Morphology: words, morphemes, word forms, segmentation, derivation, inflection, allomorphy. Automatic word-form recognition: rules, left-associative segmentation. Syntax: valency, agreement, word order; fixed vs free word orders; syntactic analysis of English with LA grammar; complex noun phrases, verb forms, interrogatives, subordinate clauses. Corpus analysis: building corpora, word frequency distributions, annotation, statistical tagging - Parts of Speech (PoS), Grammar implementation: testing grammatical rules against corpora, empirical evaluation, parser output analysis.

Unit IV: (15 Hours)

Semantics: types of semantics-logical, procedural, natural language semantics; meaning, truth, intension/extension, propositional attitudes, ontologies, vagueness. Semantic interpretation in computational systems: homomorphic semantics, time-linear interpretation, complexity of semantic processing. Database semantics: meaning as pattern matching, storing propositions in a word bank, semantic relations of structure, content coding (elementary, phrasal, clausal). Pragmatics: reference, context selection, speech acts, interpretation and production modes, conceptualization, inference, autonomous control in communication models. Integration of morphology-syntax-semantics-pragmatics for natural language understanding and generation.

Readings:

1. Hausser, R. (2001). *Foundations of computational linguistics*. Springer-Verlag Telos.
2. Bolshakov, I. A., & Gelbukh, A. (2006). *Computational linguistics: Models, resources, applications*. Computational Linguistics.
3. Grishman, R. (1986). *Computational linguistics: An introduction*. Cambridge University Press.
4. Mitkov, R. (Ed.). (2022). *The Oxford handbook of computational linguistics*. Oxford University Press.
5. Jurafsky, D. (2000). *Speech and language processing*. Pearson Education

List of Practicals:

1. Use standard Python NLP libraries (NLTK and spaCy) to perform basic text preprocessing operations, including tokenisation, stemming, lemmatisation, and stop-word removal. (**2 hours)**
2. Apply advanced text preprocessing techniques using regular expressions to handle pattern matching, text cleaning, removal, and substitution tasks. (**2 hours)**
3. Construct a phrase-structure grammar in NLTK and generate parse trees for basic English sentences using top-down and bottom-up parsers. (**2 hours)**
4. Perform part-of-speech tagging using NLTK and spaCy, compare outputs, and evaluate differences in tagging accuracy across tools.(**2 hours)**

5. Build a mini text corpus, compute word-frequency distributions, and visualize the results using Python. (**2 hours)**
6. Extract semantic relationships such as synonyms, hypernyms, and hyponyms using WordNet, and compute semantic similarity scores. (**4 hours)**
7. Transform text into Bag-of-Words (BoW) and TF-IDF vector representations, and train classifiers such as Logistic Regression and Naïve Bayes.(**4 hours)**
8. Train a Word2Vec model using Python (Gensim) and evaluate its effectiveness by performing vector-semantics tasks and a text-classification experiment. (**4 hours)**
9. Perform K-means clustering on textual data using BoW or TF-IDF vectors and interpret the resulting document clusters. (**4 hours)**
10. Design an end-to-end pipeline that includes preprocessing, vectorization, model training, and evaluation using a real-world text dataset.(**4 hours)**