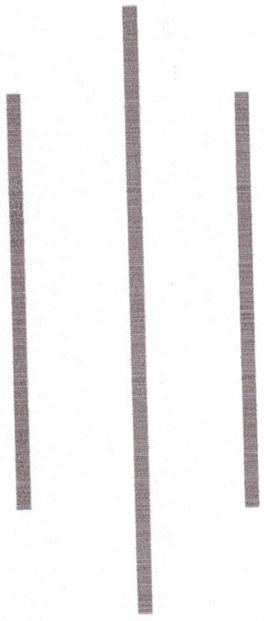
Lumbini Technological University

Institute of Engineering and Information Technology

Office of the Dean

Nepalgunj, Banke

Course Curriculum
Master of Technology in Computer Science
(M. Tech. in CS)



September, 2025

Contents

1. **Name of the program**

   Master of Technology in Computer Science (M. Tech. in CS)

2. **Course description**

   The Master of Technology in Computer Science (M. Tech. in CS) program is a two-year, four-semester academic and professional course that concentrates on computer science and new developments in information technology. This course outline provides a structure for a Master of Technology in Computer Science. It encompasses significant subjects such as assessment methods, areas of specialization, curriculum design, and goals of the program. This is just a general concept that can be adapted to fit specific business, industry, and university needs.

   Beyond the basics usually taught in undergraduate courses, this curriculum places an emphasis on more complex computer science issues. It looks at topics like distributed systems, operating systems, sophisticated data structures, algorithm creation and analysis, and software engineering techniques. Additionally, a lot of programs permit specialization in fields such as databases, cyber security, machine learning, artificial intelligence, and more. Projects will be incorporated into the courses so that students can have hands-on experience on the application of IT industries.

3. **Goal of the program**

   a. To acquire a thorough comprehension of complex computer science ideas.
   b. To gain proficiency in particular computer science fields of interest.
   c. To contribute to the field and carry out independent research.
   d. To expand your options for employment and provide access to more senior roles.
   e. To get graduates ready for leadership positions in academia, research institutions, and the IT sector.

4. **Career prospects**

   This program graduates are in great demand across a range of industries. They can work as systems engineers, software architects, data scientists, research scientists, AI/ML engineers, cybersecurity specialists, and more. A solid basis for pursuing a Ph.D. is also provided by the advanced knowledge and research expertise. A graduate of this program can be the lead in a corporate and an academic sectors as:

   - IT Manager
   - Software Architect
   - Data Scientist
   - Cybersecurity Analyst
   - Cloud Architect
   - Network Engineer

- Consultant
- Teaching Faculties
- Research Assistant/Associates
- Entrepreneur

5. Pre-requisites

The students should meet following criteria:

- Basic knowledge of programming, data structures, algorithms, database management systems
- Basic knowledge of probability, statistics, discrete mathematics and linear algebra

6. Credit and duration of the program

A M. Tech. in CS program is worth 63 credits. The M. Tech. in Computer Science program lasts for two years and four semesters. The 16-week semester requires four working hours per week for instruction and learning, omitting tests and other activities.

7. Eligibility

The eligible criteria for the M. Tech. in Computer Science program, a student must have earned a B. Tech. in Computer Science as their major, a B.Sc. in Computer Science, or Bachelor of Engineering in Computer, Electronics, and Communication and Electrical Engineering, or an equivalent degree approved by the institution.

8. Admission process

The Dean's Office (Institute of Engineering and IT) will administer an entrance exam once a year, in the fall and spring sessions. Successful applicants who meet the requirements will be admitted based on merit. The programs academic year is divided into two semesters. Admissions for the autumn semester commence in September, while those for the spring semester begin in February. The detailed academic schedule will be announced by the Dean's Office.

9. General evaluation scheme

The program evaluation scheme is categorized into following parts:

i. Evaluation practical subjects

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5

	Internal Examination	20	20	10
	Total Internal Marks		**60**	**30**
Semester End Examination			**40**	**20**

ii. Evaluation non-practical subjects

Evaluation	**Components**	**Weightage**	**Full Marks**	**Pass Marks**
Internal evaluation	Class attendance and performance	5	40	20
	Assignment	15		
	Seminar/project/presentation	20		
	Practical/lab examination	-	-	-
	Internal examination	20	20	10
	Total Internal Marks		**60**	**30**
Semester End Examination			**40**	**20**

10. Final examination format

A student is considered to have passed a subject only if they secure a minimum of 50% marks in that subject, with mandatory separate passing in both internal and external assessments of theory and practical examinations.

11. Grading system:

A students performance will be evaluated based on marks obtained in internal assessments, practicals, and semester-end examinations. These marks will be graded on an absolute basis, and the final outcome will be expressed on a four-point grading scale ranging from 0 to 4.

Grading scale:

Grade	**A+**	**A**	**A-**	**B+**	**B**	**B-**	**C+**	**C**	**F**
Marks Range	90-100	80-89	75-79	70-74	65-69	60-64	55-59	50-54	<50
Grade Point	4	3.7	3.3	3	2.7	2.3	2	1.7	0

Performance of a student in a semester is evaluated in terms of the Semester Grade Point

$$\text{SGPA} = \frac{\text{Total honor points earned in a Semester}}{\text{Total number of credits registered in a semester.}}$$

Similarly, performance of a student in a whole program is evaluated in terms of the Cumulative Grade Point Average

$$\text{CGPA} = \frac{\text{Total honor points earned}}{\text{Total number of credits completed.}}$$

Moreover, the students shall receive their semester grades and academic transcript grades only in letter grades and GPA scores. Thus, a student secure Semester Grade Point Average (SGPA) less than 2.7 or less than B Grade (B-, C+, C) in each subject of every semester may apply for grade improvement examination based on university guideline. Furthermore, a student who secures CGPA less than 3 may request for the opportunity to improve the grade in maximum two subjects. The concerned dean office will provide only one chance to upgrade their CGPA. The chance exam of the chosen subjects to improve grade shall be held as per the regular examination process

12. Attendance requirement

Student must attend at least 60% of the classes actually held in each course. It is recommended that the students must attend every lecture, tutorial, and practical classes. Those who will not able to fulfil attendance requirement are not eligible to attend in final examination of that course.

13. Maximum duration to complete course

The allocated time to complete the course is maximum 5 years. If any student will unable to complete the program during this maximum period of time but still he/she wants to complete the program, he/she will have to re-join from beginning following the complete admission procedure.

14. Reattempting the subject

A subject may generally be taken only once for a grade. However, if a student earns less than a B (i.e., B-, C+, or C) or receives an F (Fail), the course may be retaken. Retaking a course with a grade of B-, C+, or C is optional and provides an opportunity to improve performance, whereas a course with an F grade must be retaken and passed to fulfil the program requirements. In the case of a retake, the grade obtained will replace the earlier grade, and evaluation will follow a best-of-two system meaning the higher grade achieved, whether in the original attempt or the retake, will be recorded as the final grade.

15. Dissertation

As per the need and interest of students, a student must submit the research report of dissertation to the office of the dean (Institute of Engineering and IT).

16. Award of degree

Lumbini Technological University (LTU), awards of M. Tech. in CS with specialization degree after completion of all the requirement of this program as prescribed in this curriculum.

17. Course structure

The four semesters course structure is provided in the table below:

SN	Course Code	Semester	Course Name	Number of Credit
1	MCS 601	I	Design and Analysis of Algorithm	3
2	MCS 602	I	Research Methodology	3
3	MCS 603	I	Python Programming	3
4	MCS 604	I	Advance Operating System	3
5	MCS 605	I	Computational Mathematics	3
6	MCS 606	I	Seminar in Emerging Trends	1
7	MCS 651	II	Optimization Technique	3
8	MCS 652	II	Statistics and Probability	3
9	MCS 653	II	Software Project Management	3
10	MCS 654	II	Data Warehouse and Data Mining	3
11	MCS 655	II	Project Development	2
12	Elective I	II	Chosen Elective I Subject	
13	MCS 701	III	Machine Learning	3
14	MCS 702	III	Advance Compiler Design	3
15	Elective II	III	Chosen Elective II Subject	
16	Elective III	III	Chosen Elective III Subject	
17	Elective IV	III	Chosen Elective IV Subject	
18	MCS751	IV	Thesis	15

List of Elective Subjects:

Electives	Course Code	Course Name	Numbers of Credit
Elective I	MCS 656	Advance Computer Architecture	3
	MCS 657	Advanced Database Management System	3
	MCS 658	Human Computer Interaction	3
	MCS 659	Geographical Information System	3
Elective II	MCS703	Distributed and Parallel System	3
	MCS704	Service Oriented Architecture	3
	MCS705	Real Time Monitoring System	3
	MCS706	Remote Sensing	3

Elective III	MCS707	Natural Language Processing	3
	MCS708	Big Data Analytics	3
	MCS709	Wireless Ad. Hoc Network	3
Elective IV	MCS710	Digital Forensics	3
	MCS711	Linux Administration	3
	MCS712	Mobile Application development	3
	MCS713	Information Security Audit	3
	MCS714	Spatial Database Management System	3

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Design and Analysis of Algorithm**

Course Code	MCS 601	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course provides an in-depth exploration of algorithmic design and analysis techniques. Students will study a range of algorithms, understand their complexity, and learn to design efficient algorithms to solve complex computational problems. The course also covers advanced topics in algorithm theory, including approximation algorithms, randomized algorithms, and advanced data structures.

2. Course Objectives

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

- Analyze the efficiency of algorithms using Big O, Big Θ, and Big Ω notations.
- Design and implement algorithms for various computational problems.
- Evaluate and apply advanced data structures to optimize algorithm performance.
- Understand and apply concepts of NP-completeness and computational intractability.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Justify the correctness of algorithms through inductive proofs.
- Perform various analysis of algorithms using asymptotic notations.
- Understand and apply divide-and-conquer algorithm, and develop algorithm using this approach.
- Apply the dynamic programming approach, identify suitable scenarios for its use, and create and evaluate dynamic programming algorithm.
- Comprehend greedy algorithm approach, determine when to use it, develop algorithms based on this strategy, and conduct their analysis.
- Understand major graph algorithms, and apply graphs to solve engineering challenges.
- Describe methods for analyzing randomized algorithms, including their expected running times and error probabilities.
- Define approximation algorithms and their advantages, be familiar with various such algorithms.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction **(6 hrs)**

- Introduction to Algorithm
- Importance of Algorithm
- How to measure performance of an algorithm?
- Algorithm analysis including Best-, worst-, and average-case performance.
- Asymptotic notation: big-O, big-Ω, and big-Θ; little-o, and little-ω.

Unit 2: Divide and Conquer **(8 hrs)**

- Structure of Divide and Conquer algorithms
- Binary search. Fast integer multiplication
- Sorting algorithms – Merge sort and Quick Sort
- Master Theorem

Unit 3: Dynamic Programming **(8 hrs)**

- Matrix-Chain Multiplication
- Elements of dynamic programming
- Longest Common Subsequence
- 0/1 Knapsack problem
- Traveling Salesman problem

Unit 4: Greedy Algorithms **(6 hrs)**

- An activity-selection problem
- Elements of the greedy strategy
- Huffman codes

Unit 5: Graph Algorithms **(8 hrs)**

- Representation of graphs
- Breadth-first search
- Depth-first search
- Krushkal's and Prim's MST algorithms
- Dijkstra's Single source shortest path algorithm

Unit 6: Randomized and Approximation Algorithms **(6 hrs)**

- Randomized Quicksort
- Traveling Salesman Problem

Unit 7: NP-Completeness and Beyond **(6 hrs)**

- P vs. NP Problem
- Cook's Theorem
- NP-Complete Problems and Reductions

4.2 Laboratory Work (16 hrs)

Lab	Practical Title	Hours
1	Implement Linear and Binary Search. Compare their performance on different input sizes.	2 hrs
2	Implement Bubble, Insertion, and Merge Sort algorithms. Analyze their time complexity.	2 hrs
3	Implement Quick Sort (including randomized version). Compare both versions.	2 hrs
4	Implement Karatsuba Algorithm for fast integer multiplication. Compare with standard method.	2 hrs
5	Implement Matrix Chain Multiplication using Dynamic Programming.	2 hrs
6	Construct Huffman Tree for a given set of character frequencies. Generate Huffman Codes.	2 hrs
7	Implement Kruskal's and Prim's Algorithms to find Minimum Spanning Tree (MST).	2 hrs
8	Implement Dijkstra's Algorithm for shortest path in a graph with non-negative weights.	2 hrs

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

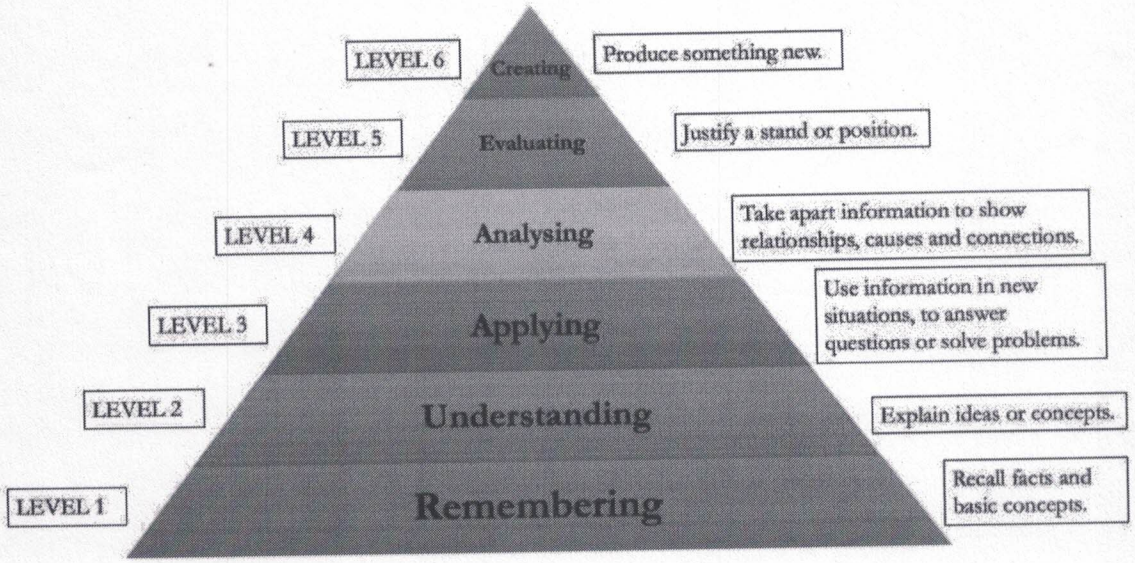
6 Books (4-10 books):

- "Introduction to Algorithms", 3rd Edition, by Cormen, Leiserson, Rivest, and Stein.
- "Fundamentals of Computer Algorithms", 2nd Edition by Sartaj Sahni and Sanguthevar Rajasekaran, and Ellis Horowitz.
- "Algorithms", 4th Edition, by Robert Sedgewick and Kevin Wayne.
- "The Algorithm Design Manual", 2nd Edition, by Steven S. Skiena.

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson

plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Research Methodology**

Course Code	MCS 602	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

Research Methodology in Computer Science/IT is a foundational course designed for first-year, first-semester Master's students pursuing a degree in Computer Science/ IT. This course provides students with a comprehensive understanding of research methods, techniques, and best practices relevant to data science research. Through a combination of theoretical lectures, practical exercises, and hands-on projects, students will learn how to formulate research questions, design research studies, collect and analyze data, and communicate research findings effectively.

2. Course Objectives

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

- To introduce students to the principles and concepts of research methodology in the context of data science.
- To develop students' skills in formulating research questions and hypotheses.
- To familiarize students with various research study designs and data collection techniques used in data science research.
- To equip students with the knowledge and tools necessary to analyze and interpret data using statistical methods and data visualization techniques.
- To enable students to effectively communicate research findings through written reports and presentations.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- **Knowledge (Remembering)**
  - Define key terms and concepts related to research methodology in data science. o Recall the principles of ethical conduct in research and data handling. o Identify the different types of research designs and sampling techniques.
- **Comprehension (Understanding)** o Summarize the importance of research in advancing knowledge and solving real-world problems in the context of data science.
  - Explain the process of formulating research questions and hypotheses.

- Interpret various types of research designs and their applicability to different research scenarios.

- **Application (Applying)** o Apply appropriate data collection methods and sampling techniques to design research studies in data science.
  - Utilize statistical techniques to analyze and interpret data collected from research studies.
  - Implement data visualization techniques to communicate research findings effectively.
- **Analysis (Analysing)** o Analyze the strengths and limitations of different research designs and sampling methods in data science research.
  - Evaluate the validity and reliability of the research findings based on statistical analysis.
  - Compare and contrast different data visualization techniques and their effectiveness in conveying insights from data.
- **Synthesis (Creating)** o Design research proposals and project plans for data science research studies, including research questions, objectives, and research methodologies.
  - Develop research reports and academic papers that adhere to standard formatting and citation guidelines.
  - Synthesize research findings into coherent and visually appealing presentations for diverse audiences.

- **Evaluation (Evaluating)**
  - Critically assess the ethical implications of data collection, analysis, and reporting in data science research.
  - Evaluate the appropriateness of research designs and methodologies based on research objectives and constraints.
  - Provide constructive feedback on peer-reviewed research proposals, reports, and presentations, focusing on clarity, rigor and validity.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction to Research Methodology (4 hrs)

- Overview of Research Methodology in Data Science.
- Importance of Research in advancing knowledge and solving real-world problems.
- Ethical considerations in data science research.
- Formulating research questions and hypotheses.

Unit 2: Research Design and Sampling (8 hrs)

- Types of Research Design (e.g., experimental, observational, survey).
- Sampling techniques and strategies for data collection.
- Sample size determination and power analysis. - Experimental design and control variables.

Unit 3: Data Collection Methods **(10 hrs)**

- Primary data collection methods (e.g., surveys, interviews, observations).
- Secondary data sources (e.g. public datasets, data repositories).
- Data preprocessing and cleaning techniques.
- Data privacy and confidentiality considerations.

Unit 4: Data Analysis Techniques **(12 hrs)**

- Descriptive statistics and exploratory data analysis (EDA).
- Inferential statistics and hypothesis testing.
- Regression analysis and predictive modelling.
- Machine learning algorithms for data analysis (e.g., classification, clustering, regression).

Unit 5: Data Visualization and Interpretation **(6 hrs)**

- Principles of data visualization and graphical representation.
- Tools and libraries for data visualization (e.g., Matplotlib, Seaborn, ggplot2).
- Interpretation of visualization and communicating insights effectively.
- Interactive visualization techniques for exploratory analysis.

Unit 6: Research Communication and Reporting **(8 hrs)**

- Writing research proposals and project plans.
- Structuring research reports and academic papers.
- Presenting research findings orally and visually. - Peer review process and academic publishing.

4.2 Laboratory Work **(16 hrs)**

- Formulating research questions based on provided datasets.
- Collecting data from online sources or using survey tools.
- Preprocessing collected data to handle missing values and outliers.
- Performing EDA on collected datasets using Python/R libraries.
- Creating visualizations to identify patterns and insights in data.
- Implementing random sampling and stratified sampling techniques on provided datasets.
- Conducting hypothesis tests (e.g., t-tests, chi-square tests) on sample data.
- Interpreting test results and concluding.
- Building regression models to predict outcomes based on predictor variables.
- Assessing model performance and interpreting regression coefficients.
- Implementing machine learning algorithms (e.g., classification, clustering) on research datasets.
- Interpreting machine learning model outputs and evaluating model performances.

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

i) Textbooks:

- "Research Methods for Data Science" by John W. Creswell and J. David Creswell.
- "Practical Statistics for Data Scientists" by Andrew Bruce and Peter Bruce.

ii) Online resources:

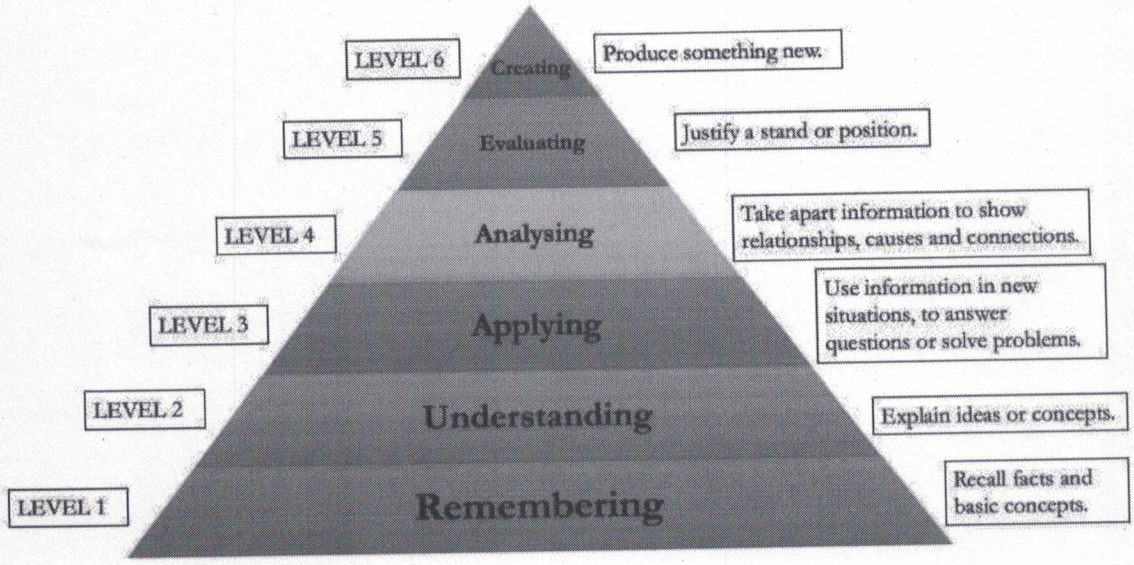
Research articles, tutorials, and documentation on research methodology and data science techniques.

iii) Software tools:

Python/R programming environments, Jupyter Notebooks, statistical analysis software (e.g., SPSS, SAS).

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue Assess Critique Defend Evaluate Judge Justify

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose Construct Create Devise Generate Organise Plan Produce

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject:Python programming**

Course Code	MCS 603	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course builds on a fundamental concept on programming in Python and leads to advanced programming techniques, covering object-oriented and high-level program design. It explores basic programming structures such as conditionals, iterations, recursions, functions, modules, files and object-oriented techniques such as classes and objects, and inheritance. The syllabus is divided into 7 units, where each chapter includes hands-on practical and tutorial sessions.

2. Course Objectives

The general objectives of this course is:

- Develop advanced-level programming skills in Python.
- Enable students to write efficient, readable, and maintainable Python code.
- Familiarize students with Python's extensive standard and third-party libraries.
- Equip students to perform data analysis, visualization, and modeling using Python tools.
- Encourage application of programming skills to solve real-world IT problems.
- Strengthen problem-solving through the use of structured, modular, and object-oriented programming.
- Prepare students for professional tasks in software development, data science, and automation.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Write a clear, effective and efficient computer program in Python
- Develop, debug and test Python program
- Use Python standard libraries and third-party modules
- Perform data analysis and visualisation
- Develop a complete application on python

4. Course Details

4.1 Theory **(48 hrs)**

Unit-1 Introduction **(6 hrs)**

- Computer program, programming language, software, compilers, interpreters
- Formal and natural languages, program debugging, syntax errors, runtime errors, experimental debugging, documentation, and comments.
- Program development life cycle, flowcharts, algorithms, and pseudocodes
- Python programming, variables, expressions and statements, operators, and keywords
- Variable and data types, variable naming, keywords, statements
- Evaluating expression, operators and operands, converter functions

Unit 2 Conditionals, Iterations and Functions **(8 hrs)**

- Boolean values and expressions, logical operators, conditional execution, nested conditionals,
- Iterations, assignments, updating variables, for loop, while statement, break statements, other flavours of loops, continue statements, more generalisations, nested loops for nested data.
- Basics of functions, function definitions, call, flow of executions, function arguments, return values, local variables and parameters

Unit 3: Strings, tuples, lists and dictionaries **(8 hrs)**

- Working with strings, lengths, slices, string comparison, immutable, find function, looping and counting, split method, string format method.
- Working with tuples and lists, tuple assignment, and tuple as return values
- Lists, accessing elements, membership, list operations, slices, deletion, aliasing, cloning, list methods, pure functions and modifiers, string and lists, list and range, nested lists
- Creating dictionaries, assessing values, methods, aliasing and copying, memoizations, sparse matrices
- Interpreting through dictionaries, comprehensions,

Unit 4: Modules and files **(6 hrs)**

- Concept of modules, random numbers, popular modules such as the time module, the math module, and creating your own modules.
- Namespaces, scope and lookup rules, attributes and dot operators.
- Import statements and its variants unit test and modules.
- Working with files, writing your first files, reading from file, working with binary files, directories, and fetching from the web.

Unit 5: Classes, objects and inheritance **(8 hrs)**

- Object-oriented programming, user-defined compound data types, attributes, initialisers, adding methods to a class, instance as arguments and parameters, return values
- Working with objects, rectangles, and mutable objects, sameness, and copying.
- Pure functions, modifiers, generalisations, operator overloading and polymorphism.

- Inheritance, type of inheritance, methods, self, __init_

Unit 6: Python libraries for computations **(6 hrs)**

- Working with existing libraries, os, sys, datetime, etc.
- Working with command line arguments
- NumPy arrays, slicing, operations, creating arrays with different operations, reshaping, flattening , transposing, indexing, Boolean indexing, broadcasting.
- Numpy for statistics, aggregate functions, axis operations, vectorised operations, linear algebra operations such as matmul, dot , determinant, rank, inverse, transpose, linlang etc.
- Pandas, reading csv, json, excel, data exploration, data selections, filtering, cleaning, transformation, grouping, aggregation, sorting and ranking, merging, joining, and time series data with pandas.

Unit 7: Data Visualisation **(6 hrs)**

- Why data visualisations, effective visualisation principles, Exploratory data analysis,
- Types of visualisation charts, graphs, and data importing.
- Matplotlib basics, axes, plots and customisations, lifecycle of plots, pairwise data, gridded data, statistical distribution, boxplots, 3D plots
- Seaborn basics, themes and aesthetics, categorical vs numerical plots, styling, pair plots, heatmaps, facet Grids, saving and exporting plots.

4.2 Laboratory Work **(16 hrs)**

Lab	Practical Title	Hours
1	Install Python, write a simple script, run and debug it.	2 hrs
2	Develop a program using conditionals, loops (for, while), and functions with arguments and return values.	2 hrs
3	Develop a program using strings, tuples, lists, and dictionaries demonstrating slicing, methods, and comprehensions.	2 hrs
4	Develop a program that imports modules, works with files (read/write), and fetches data from the web.	2 hrs
5	Write a program demonstrating classes, objects, methods, and inheritance (e.g., card game example).	2 hrs
6	Use Python libraries (os, sys, datetime), work with command-line arguments, NumPy arrays, and Pandas data manipulation.	2 hrs
7	Develop a data visualization project using Matplotlib and Seaborn for different types of plots and charts.	4 hrs

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

i) Textbooks:

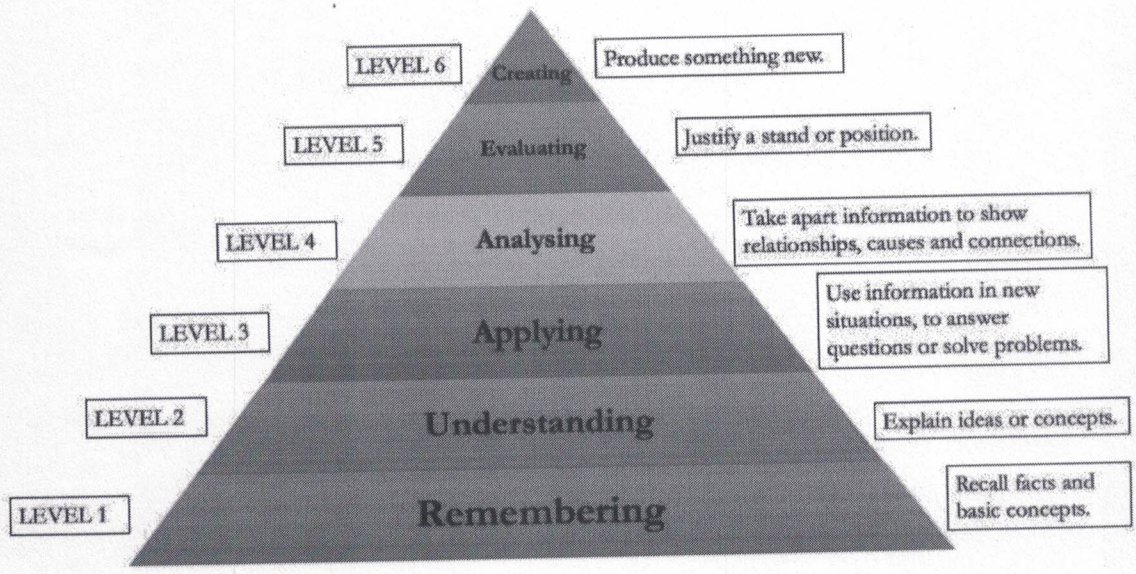
- How to Think Like a Computer Scientist: Learning with Python 3, Peter Wentworth, Jeffrey Elkner, Allen B. Downey and Chris Meyers (3rd Edition)

ii) References:

- How to Think Like a Computer Scientist: Learning with Python 3rd editions
- Downey, A. (2012). Think python. " O'Reilly Media, Inc.".
- McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.".
- Vasiliev, Y. (2022). Python for Data Science: A Hands-on Introduction. No Starch Press.

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue Assess Critique Defend Evaluate Judge Justify

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Advanced Operating System**

Course Code	MCS 604	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course explores the advanced concepts, principles, and methodologies used in the design and operation of modern operating systems. Building on foundational knowledge of operating systems, the course emphasizes the mechanisms and policies that enable multitasking, resource management, concurrency, and system security in distributed, real-time, and virtualized environments

2. Course Objectives

The general objectives of this course is:

- To provide students with a deep understanding of the principles and technologies that underlie modern operating systems.
- To enable students to analyse and design advanced operating systems and their components.
- To prepare students to work with and develop emerging technologies in the field of operating systems, such as distributed systems, cloud computing, and mobile devices.
- To expose students to real-world case studies and examples of advanced operating systems in action.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Analyze advanced operating system structures and architectures, including kernel-level and user-level design choices in modern OS implementations.
- Explain and evaluate process synchronization, deadlock handling, and concurrency control mechanisms used in multiprocessor and distributed environments.
- Apply resource management policies for efficient CPU scheduling, memory allocation, and I/O handling in real-time and virtualized systems.
- Understand and analyze distributed system concepts, including communication, coordination, clock synchronization, and fault tolerance.
- Demonstrate the implementation and usage of virtual memory, paging, and caching mechanisms in contemporary operating systems.
- Critically examine system-level security measures such as access control, authentication, intrusion detection, and isolation in shared environments.

- Implement and simulate advanced OS features (e.g., threads, file systems, inter-process communication) using modern programming tools and techniques.
- Assess the role of virtualization and containerization technologies in system design and cloud-based deployment scenarios.
- Design and analyze algorithms for scheduling, memory management, and synchronization using real-world case studies and open-source operating systems.
- Engage in research and collaborative projects to explore cutting-edge developments in distributed, real-time, embedded, and cloud operating systems.

4. Course Details

4.1 Theory (48 hrs)

Unit-1 Introduction (3 hrs)

- Operating System Concepts
- System Calls
- Operating System Structure
- Introduction to key concepts and terminologies used in advanced operating systems, such as concurrency, virtualization, and distributed systems.

Unit-2 Processes, Threads, Deadlocks (10 hrs)

- Processes (Process model, Process Creation, State diagram, Implementation).
- Threads (Thread Model, POSIX Threads, Multi-Threading models, Thread Implementation).
- Inter-process Communication (Introduction to Critical Section Problem, Mutual Exclusion by Busy Waiting, Sleep and Wakeup, Semaphores, Mutexes, Monitors).
- Process Scheduling (Process scheduling and its categories).
- Deadlocks (Deadlock Characterization, Handling Deadlocks: Prevention, Avoidance, Detection, Recovery)

Unit-3 Memory Management (8 hrs)

- A memory abstraction
- Virtual memory
- Page replacement algorithms
- Design and implementation of issues of paging system
- Segmentation
- Advanced topics in memory management, such as persistent memory and support for high-performance computing and big data applications

Unit-4 Storage and I/O Systems (5 hrs)

- Files and Directories
- Implementing file systems

- Disk Scheduling
- RAID structure
- I/O Systems

Unit-5 Protection and Security **(3 hrs)**

- System Protection Mechanism
- Operating system security challenges
- System Security Mechanism

Unit-6 Virtualization and the cloud **(10 hrs)**

- Requirement of virtualization
- Hypervisors
- Techniques of virtualization
- Memory virtualization
- I/O virtualization
- Virtual Machines on Multicore CPUs
- Network, Distributed, Cloud file system

Unit-7 Multiple Processor Systems **(9 hrs)**

- Multiprocessors (Hardware, Operating System Types, Synchronization, Scheduling)
- Multicomputer (Hardware, Software, Remote Procedure Call, Distributed Shared Memory, Multicomputer Scheduling, Load Balancing)
- Distributed Systems (Introduction, Distributed Synchronization, Security)

4.2 Laboratory Work **(16 hrs)**

Lab	Practical Title	Hours
1	Demonstrate use of system calls like fork(), exec(), getpid(), wait() in Linux.	2 hrs
2	Implement process creation and thread management using POSIX threads and compare threading models.	2 hrs
3	Implement Inter-Process Communication (IPC) using pipes, shared memory, and semaphores.	2 hrs
4	Simulate Deadlock Detection and Avoidance using Banker's algorithm or resource allocation graph.	2 hrs
5	Simulate page replacement algorithms like FIFO, LRU, and Optimal using Python/C.	2 hrs
6	Simulate Disk Scheduling algorithms such as FCFS, SSTF, and SCAN.	2 hrs
7	Demonstrate Virtualization using tools like VirtualBox or KVM and explain memory/I/O virtualization.	2 hrs
8	Implement a basic Distributed Clock Synchronization algorithm (e.g., Lamport's algorithm).	2 hrs

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

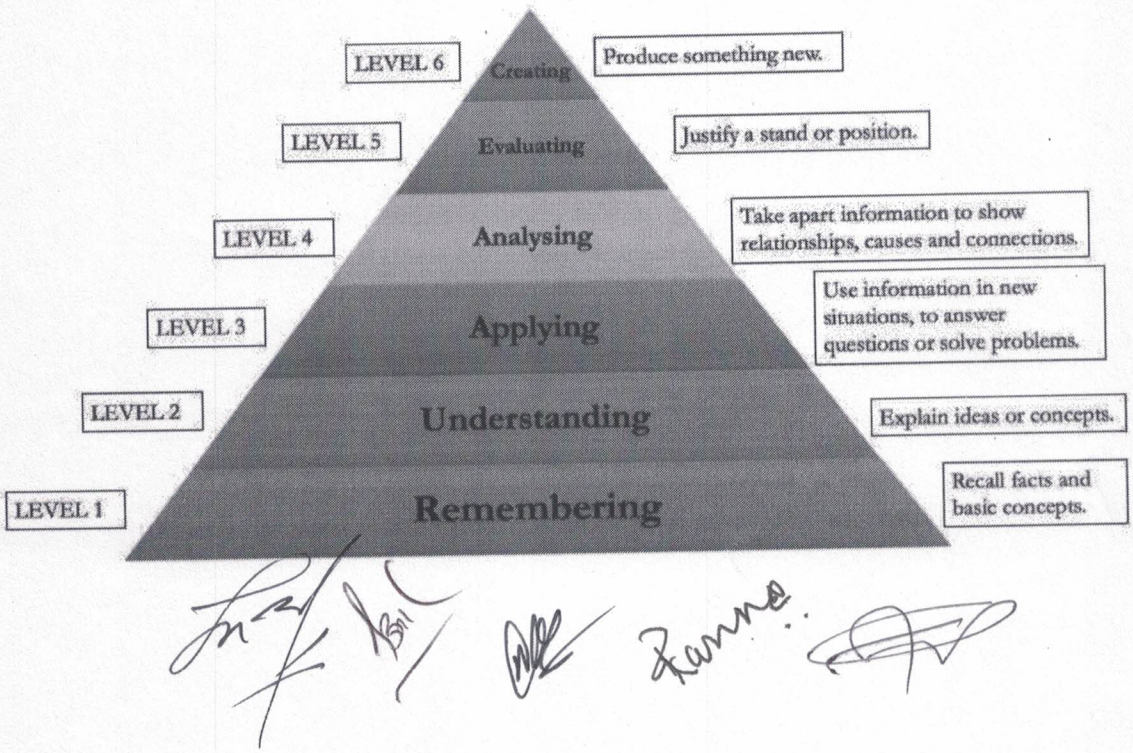
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

- Tanenbaum, A. S., & Bos, H. Modern Operating Systems, 5/e. Pearson.
- Silberschatz, A., Gagne, G., & Galvin, P. B. Operating System Concepts, 10/e. Wiley.
- Stallings, W. Operating Systems: Internals and Design Principles, 9/e. Pearson

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue Assess Critique Defend Evaluate Judge Justify

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose Construct Create Devise Generate Organise Plan Produce

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Computational Mathematics**

Course Code	MCS 605	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course provides a foundation in mathematics essential for data science applications. It focuses on key concepts in scientific computing, linear algebra, and differential equations, enabling students to apply mathematical methodologies to solve complex data science problems. Students will gain hands-on experience through practical exercises and projects, enabling them to apply mathematical techniques to real-world data science problems.

2. Course Objectives

The general objectives of this course is:

- Develop a strong foundation in mathematical tools and techniques used extensively in data science.
- Enhance problem-solving skills by applying mathematical concepts to practical data science scenarios.
- Foster critical thinking and analytical abilities to effectively interpret and utilize mathematical results in data analysis.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- **Scientific Computing:**
  - Explain different types of errors encountered in scientific computing
  - Implement numerical algorithms to solve problems encountered in data science
  - Utilize interpolation and curve fitting techniques to solve practical problems in data science
- **Linear Algebra:**
  - Apply matrix operations and vector manipulations to represent and analyze data.
  - Solve systems of linear equations arising from data science applications.
  - Utilize concepts of eigenvalues and eigenvectors for dimensionality reduction and data analysis.
- **Differential Equations (ODEs):**
  - Model real-world phenomena relevant to data science using ordinary differential equations.

- Employ numerical techniques to solve ODEs and interpret the solutions in the context of data analysis.
- Understand the limitations and applications of ODEs in data science modeling.

4. Course Details

4.1 Theory (48 hrs)

Unit 1 Scientific computing in General (8 hrs)

- Errors : Absolute, Relative, round off, truncation
- Significant digits and Error propagation
- Solution to single non linear equation : Newton's method, Iteration method
- Solution to system non linear equations : Newton's Method, Iteration method

Unit 2 Interpolation and curve fitting (8 hrs)

- Curve Fitting : Linear, exponential, nth degree
- Interpolation : Lagrange and Newton
- Spline interpolation : Linear, Cubic
- Bspline interpolation : Linear, Cubic

Unit 3 Matrices and Linear system of equations (6 hrs)

- Vectors : Scalar Product, Norm, Angle, Distance
- Matrices : Symmetric, Skew-Symmetric, Hermitian, Orthogonal, Unitary Matrices ? The Gram-Schmidt process
- Solution to systems :
  - Direct methods :Gaussian elimination, LU decomposition, tridiagonal systems
  - Iterative methods : Gauss-Seidel, SOR, conjugate gradient

Unit 4 Eigenvalue problems (8 hrs)

- Diagonalization
- Power and inverse power method
- Householder's method
- The QR method
- Singular value decomposition (SVD)
- Principal component analysis (PCA)

Unit 5 Introduction to differential equations (6 hrs)

- Motivation with applications in data science
- Definition and terminology
- Classification of differential equations
- Geometrical meaning of first order differential equation
- Initial and boundary value problems

Unit 6 Solution of differential equations **(12 hrs)**

- Initial value problems (IVPs) : Eulers method, Runge-Kutta methods
- System and higher order ODEs : Eulers method, Runge-Kutta methods
- Boundary value problems (BVPs) : Finite difference method
- Partial differential equations (PDEs) : Laplace and Poisson equations

4.2 Laboratory Work **(16 hrs)**

Lab	Practical Title	Hours
1	Write a program to compute absolute, relative, round-off, and truncation errors.	2 hrs
2	Implement Newton-Raphson and Fixed-Point Iteration methods to solve a nonlinear equation.	2 hrs
3	Solve a system of nonlinear equations using Newton's method and iterative method.	2 hrs
4	Fit data using linear, exponential, and polynomial (nth degree) regression models.	2 hrs
5	Implement Lagrange and Newton interpolation for a given set of data points.	2 hrs
6	Perform cubic spline and B-spline interpolation on sample data.	2 hrs
7	Solve a system of linear equations using Gaussian elimination and LU decomposition.	2 hrs
8	Solve initial value problems using Euler's and Runge-Kutta methods.	2 hrs

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

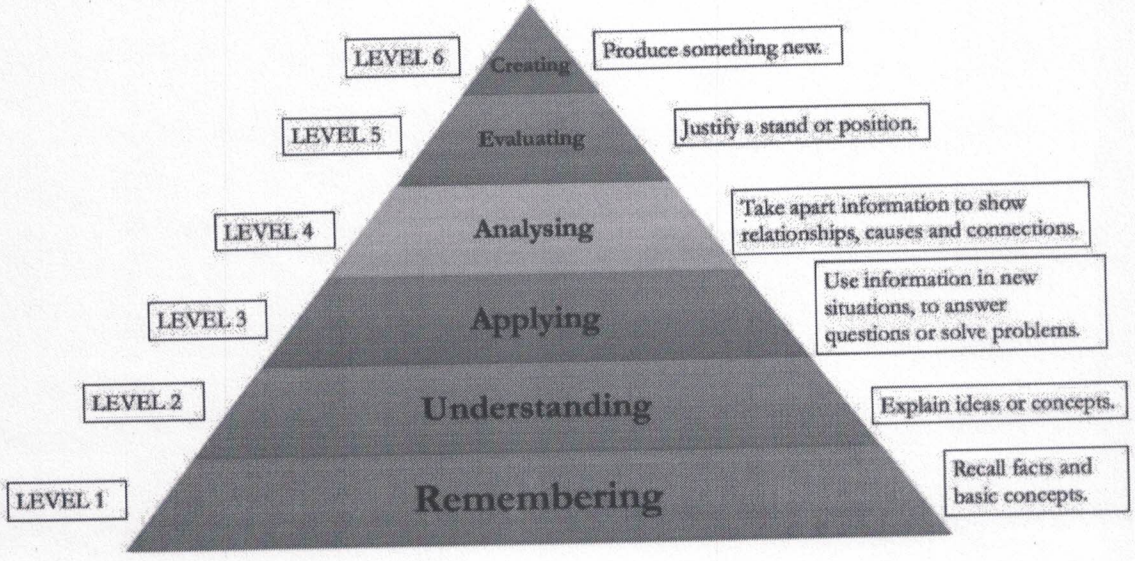
6 Books (4-10 books):

- Modelling with Differential Equations by D. U. Burghers, and M.S Borrie, Ellis Horwood Limited.
- Differential Equations and Their Applications by Zafar Ahsan, PHI Learning Pvt. Ltd.
- Numerical Methods and applications by E. Ward Cheney, and David R. Kinacaid, Cengage Learning
- Applied Numerical Analysis, C.F. Gerald and P.O. Wheately, Addison Wesely
- Linear Algebra and applications by Gilbert Strang, Cengage Learning

- Introduction to Computational linear algebra by Nabil Nassif, Jocelyne Erhel and Bernard Philippe, CRC press

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply *Calculate* *Demonstrate* *Interpret* *Show* *Solve* *Suggest*

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* *Investigate*

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Optimization Theory**

Course Code	MCS 651	Year/Semester	I/II
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course is subjected to provide the fundamental concept of Optimization Theory necessary for data science. Students will learn the concept of Linear Programming, Integer Programming, Network Optimization and Non-Linear Programming together with the idea of Calculus in Optimization techniques.

2. Course Objectives

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

- Solve optimization problems relevant to data science using appropriate knowledge of optimization theory.
- Apply computational tools and optimization algorithms to practical problem-solving.
- Analyze, design, and implement optimization techniques for real-world applications.
- Evaluate the efficiency and effectiveness of different optimization methods in computational contexts.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Solve optimization problems relevant to data science by applying appropriate optimization theory.
- Utilize computational tools effectively and implement optimization algorithms for practical problem-solving.
- Analyze optimization techniques, design suitable approaches, and implement them for real-world applications.
- Apply dynamic optimization methods, identify suitable problem scenarios, and evaluate their effectiveness.
- Understand greedy and heuristic optimization strategies, determine appropriate use cases, and analyze their performance.
- Apply graph-based optimization algorithms to address engineering and computational challenges.
- Evaluate randomized optimization algorithms by analyzing expected performance and error probabilities.
- Define approximation methods for optimization, explain their advantages, and apply them to solve complex problems.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction (4 hrs)

- Introduction to Optimization
- Types of Optimization: Linear, Non-Linear, Convex, Non-Convex
- Application of Optimization in Data Science

Unit 2: Optimization Using Calculus (6 hrs)

- Introduction to Local Extrema and Global Extrema
- Optimization Techniques for Univariate Functions
- Optimization Techniques for Multi-Variate Functions

Unit 3: Linear Programming (10 hrs)

- Concept of Linear Programming
- Simplex Method
- Duality Theorem

Unit 4: Integer Programming (6 hrs)

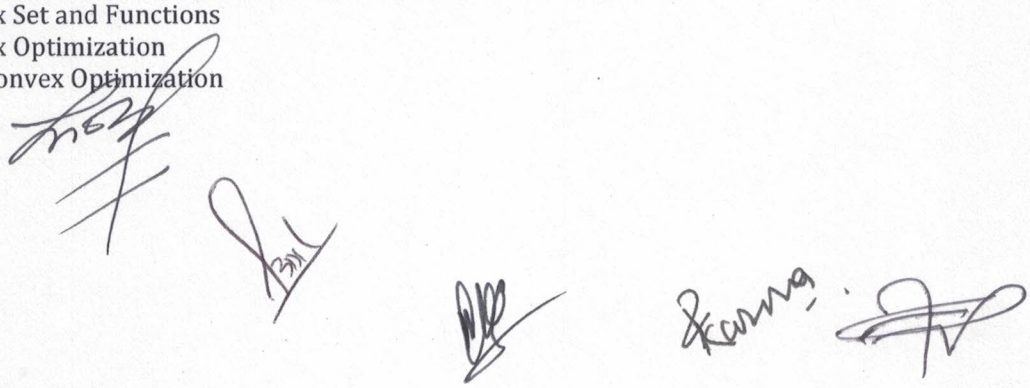
- Introduction to Integer Programming
- Types of Integer Programming Problem
- Gomory's All Integer Cutting Plane Method
- Branch and Bound Method

Unit 5: Network Optimization (10 hrs)

- Introduction
- Transportation Problem
- Assignment Problem
- Travelling Sales-Man Problem

Unit 6: Non-Linear Optimization (12 hrs)

- Introduction to Non-Linear Optimization
- Unconstrained Optimization (One variable and Multi-Variable)
- Convex Set and Functions
- Convex Optimization
- Non-Convex Optimization



4.2 Laboratory Work (16 hrs)

- Use of TORA Software in Optimization
- Use of Python Libraries in Optimization

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

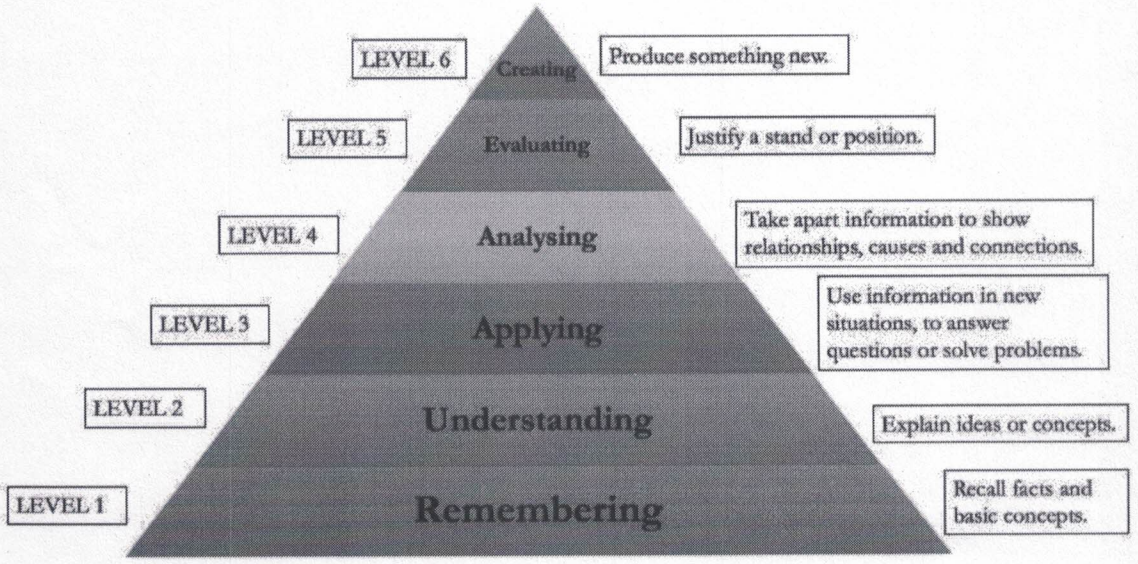
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

- Frederick S. Hillier and Gerald J. Lieberman, Introduction to Operations Research, McGraw-Hill.
- J. K. Sharma, Operations Research (Theory and Applications), Macmillan India, New Delhi.
- Michel Bierlaire, Optimization: Principles and Algorithms, EPFL Press.
- Donald A. Pierre, Optimization Theory with Applications, Dover Publications. Inc. New York
- Jorge Nocedal, Stephen Wright, Numerical Optimization, Springer.
- G. Hadley, Linear Programming, Addison Wesley Publishing Company.
- G. B. Thomas, J. Hass, C. Heil, M. D. Weir, Thomas' Calculus, Pearson.

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue Assess Critique Defend Evaluate Judge Justify

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose Construct Create Devise Generate Organise Plan Produce

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Statistics and Probability**

Course Code	MCS 652	**Year/Semester**	I/II
Credit Weightage	3	**Lecture**	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course introduces students to basic to advanced concepts and techniques applied in Statistics and Data Science domain. Students will gain theoretical as well as practical knowledge on various topics of Descriptive Statistics, Inferential Statistics, Statistical Modeling. This course will also familiarize students with recent data science tools for solving practical problems.

2. Course Objectives

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

- Familiarize students with random events and their probabilities
- Familiarize students with some advance knowledge of Statistical inference.
- Provide students with theoretical and practical knowledge statistical modeling.
- Allow students to explore state-of-the-art data science tools for solving problems.
- Let students gain experience of doing independent study and research.
- Prepare students for both academic and industrial career in data science domain.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Explain different types of data.
- Calculate different measures of data.
- Represent data with some graphs.
- Measure probabilities with different forms of random events.
- Describe some standard form of statistical distributions.
- Perform parameter estimation and hypothesis testing mean and variance.
- Familiarize with bi-variate probability distribution.
- Measure association and dependency of two or more variables.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Descriptive Statistics (6 hrs)

- Nominal, Ordinal, Ratio and Scale Data
- Measures of Central Values, Variation, Skewness, Kurtosis

- Graphical Representation of Scale Data using Stem-and-Leaf Plot, Frequency Curve, Histogram, Boxplot, Scatter Plot

Unit 2: Probability **(8 hrs)**

- Random Experiment, Sample Space, Events, Trials
- Classical, Empirical, Subjective and Axiomatic Approaches of Probability
- Mutual Exclusiveness of Events, Independence of Events, Addition Rule
- Conditional Probability, Multiplication Rule
- Partition of Sample Space, Total Probability Rule
- Bayes' Rule

Unit 3: Distribution of Random Variables **(10 hrs)**

- Random Variables – Discrete and Continuous
- Probability Functions of Random Variables
- Expectation and Variance of Random Variables
- Moments of Random Variables and Moment Generating Function
- Discrete Probability Distributions – Uniform, Binomial, Poisson, Geometric, Hypergeometric Distributions
- Continuous Probability Distributions – Uniform, Normal, Exponential, Gamma Distributions

Unit 4: Inferential Statistics **(12 hrs)**

- Sampling Distributions – χ^2, t and F Distributions
- Parameter Estimation – Method of Moments, Method of Maximum Likelihood
- Test of Hypothesis – Neyman-Pearson Lemma, Likelihood Ratio Test
- Interval Estimation and Test of Hypothesis Concerning Mean of a Normal Population
- Interval Estimation and Test of Hypothesis Concerning Variance of a Normal Population
- Interval Estimation and Test of Hypothesis Concerning Difference in Means of Two Normal Populations (Both Independent and Dependent Samples)
- Interval Estimation and Test of Hypothesis Concerning Ratio of Variances of Two Normal Populations
- ANOVA
- p-value of a Test

Unit 5: Bivariate Probability Distributions **(6 hrs)**

- Joint Probability Mass Function, Marginal Probability Mass Function, Conditional Probability Mass Function
- Joint Probability Density Function, Marginal Probability Density Function, Conditional Probability Density Function
- Expectation, Variance and Covariance of Bivariate Random Variables

Unit 6: Correlation and Regression **(6 hrs)**

- Measure of Correlation of Bivariate Random Variables – Pearson's Correlation Coefficient, Spearman's Rank Correlation, Test of Significance of Correlation
- Simple Linear / Non-Linear Regression

- Multiple Linear Regression
- Test of Significance of Regression Coefficients
- Test of Significance of Regression Model
- Measure of Accuracy of Regression Models – MSE, MAE, MAPE, R-Square, Adjusted R-Square

4.2 Laboratory Work (16 hrs)

Use MS-Excel/ STATA/ R/ SPSS to-

- Obtain statistical values of numerical data
- Obtain frequency distribution/ relative frequency/ cumulative-frequency distribution of categorical data
- Bin continuous data and obtain frequency distribution
- Graphing categorical data- bar plot, pie chart, line graph
- Graphing continuous data- histogram, stem-and-leaf plot, boxplot
- Develop probability distribution of discrete random variable
- Obtain probabilities related to binomial, Poisson and normal distributions
- Fitting of binomial, Poisson and normal distributions
- Test of hypothesis on mean of a normal population
- Test of hypothesis on difference in means of two normal populations (dependent and independent samples)
- Test of hypothesis on variance of a normal population
- Test of hypothesis on ratio of variance of two normal populations.
- Develop joint probability distribution and marginal distributions of two-dimensional discrete random variables.
- Obtain correlation between two samples
- Carry simple linear/ non-linear regression
- Carry multiple linear regression

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

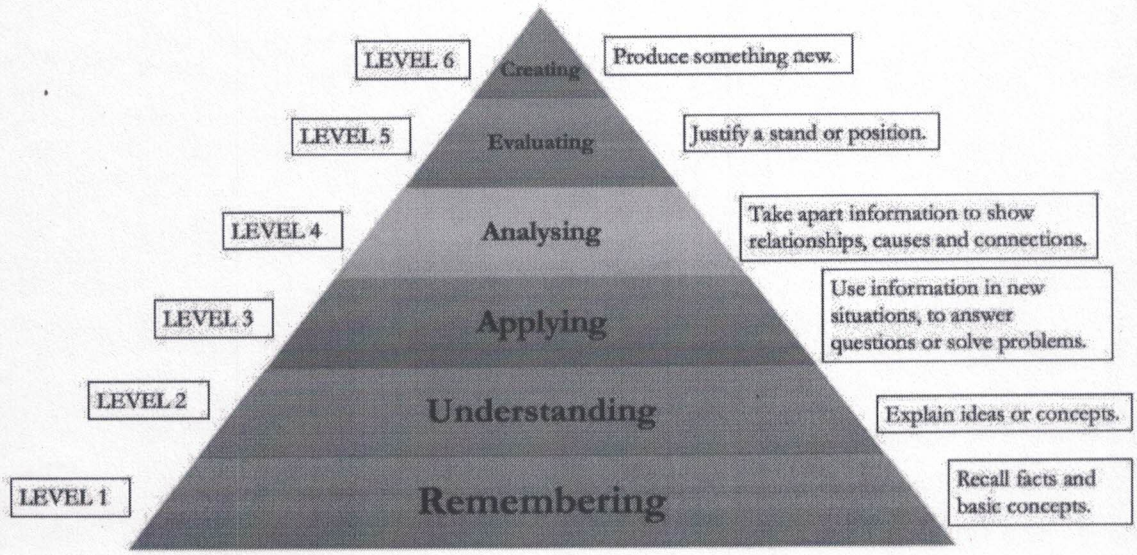
6 Books (4-10 books):

- Michael Baron Probability and Statistics for Computer Scientists,2019. CRC Press Taylor & Francis Group

- Biswas, S. (1991): Topics in Statistical Methodology, Wiley Eastern, India
- Montgomery, Douglas C.; Runger, George C.: Applied Statistics and Probability for Engineers 7th Edition, Wiley
- Bhat, B.R. (1999): Modern Probability Theory - An Introductory Textbook, New Age International, New Delhi
- Walpole, Ronald; Myers, Raymond; Myers Sharon; Ye, Keying- Probability and Statiscits for Engineers and Scientists Pearson Publication

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply *Calculate* *Demonstrate* *Interpret* *Show* *Solve* *Suggest*

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* *Investigate*

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Software Project Management**

Course Code	MCS 653	Year/Semester	I/II
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course, designed for M.Tech students in Information Technology and Computer Science, provides a comprehensive understanding of Software Project Management principles and practices. It covers project planning, effort estimation, risk management, human factors, leadership, and software quality assurance. Students will explore theoretical concepts and practical techniques to effectively manage software projects, addressing challenges such as project constraints, risks, and team dynamics. Through hands-on exercises and case studies, students will apply tools like Gantt charts, PERT/CPM, and COCOMO models to real-world scenarios, preparing them to lead successful software projects.

2. Course Objectives

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

- Understand the characteristics, constraints, and challenges of software projects and their management.
- Develop skills in project planning, including work breakdown structures, activity sequencing, and resource allocation.
- Master software effort estimation techniques, such as LOC, Function Point, and COCOMO models.
- Identify, analyze, and mitigate risks in software projects to ensure successful outcomes.
- Explore human factors, leadership, and team organization to foster effective collaboration and conflict resolution.
- Apply software quality assurance techniques, including Formal Technical Reviews (FTR) and Cleanroom Methodology.
- Gain practical experience in using project management tools and methodologies to monitor and control software projects.

3. Learning Outcomes

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

- Explain the key characteristics, constraints, and reasons for software project failures.
- Create comprehensive project plans, including Work Breakdown Structures (WBS), Gantt charts, and PERT/CPM diagrams.

- Apply software effort estimation techniques to predict project costs and timelines accurately.
- Identify, prioritize, and develop response strategies for software project risks.
- Demonstrate effective leadership, communication, and conflict resolution skills in managing project teams.
- Implement software quality assurance processes, including reviews and reliability measures.
- Use project management tools to monitor progress, control resources, and evaluate project performance using Earned Value Analysis.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction of software Project Management (8 hrs)

- Projects and Project Characteristics
- Project Constraints
- Problems with Software Projects
- Software Project Failures & Major Reasons
- Software Project Management
- Project Management Framework Project
- Stakeholders
- Project Organization Types
- Project Charter.

Unit 2: Project Planning (8 hrs)

- Definition Planning
- Planning Tasks
- Work Breakdown Structure (WBS),
- Activity Planning
- Activity Sequencing
- Time Scheduling
- Gantt Chart
- PERT/CPM
- SQA and Test Plan
- Resource Plan
- Communication Plan
- Project Monitoring and Control
- Earned Value Analysis

Unit 3: Software Effort Estimation (8 hrs)

- Software Effort Estimation
- Need for Software Estimation
- Software Estimation Process
- Software Estimation Techniques
- Expert Judgment based Estimation
- LOC, Function Point and Object point Analysis

- COCOMO cost estimation method

Unit 4: Project Risk Management **(8 hrs)**

- Risk Identification,
- Top 10 Software Project Risks
- Risk Analysis and Prioritization
- Risk Response Planning
- Risk Resolution
- Risk Tracking and Control

Unit 5: Human Factors and Leadership Team Organization, Contract Management **(8 hrs)**

- Motivation
- Communication,
- Handling Difficult People
- Leadership and health safety
- Conflict resolution
- Introduction to contract management and types of contracts
- Stages on contract, placement, typical terms of a contract, contract management, acceptance

UNIT 6: SOFTWARE QUALITY **(8 hrs)**

- Software Quality - Quality Measures - FURPS - Software Quality Assurance - Software Reviews - Format Technical Review (FTR)
- Formal Approaches to SQA - Software Reliability
- Introduction to SQA - The Software Quality Assurance Plan - Formal approaches to SQA - Clean room Methodology

4.2 Laboratory Work **(16 hrs)**

Lab	Practical Title
1	Review of 3 fictions proposal, review of Software requirement specification (SRS). Students are asked to develop the proposal and SRS.
2	Compare and contrast the different types of SDLC
3	Develop and design the project planning.
4	Develop detailed activity networks and schedules (CPM/ PERT)
5	Testing tools.

5. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

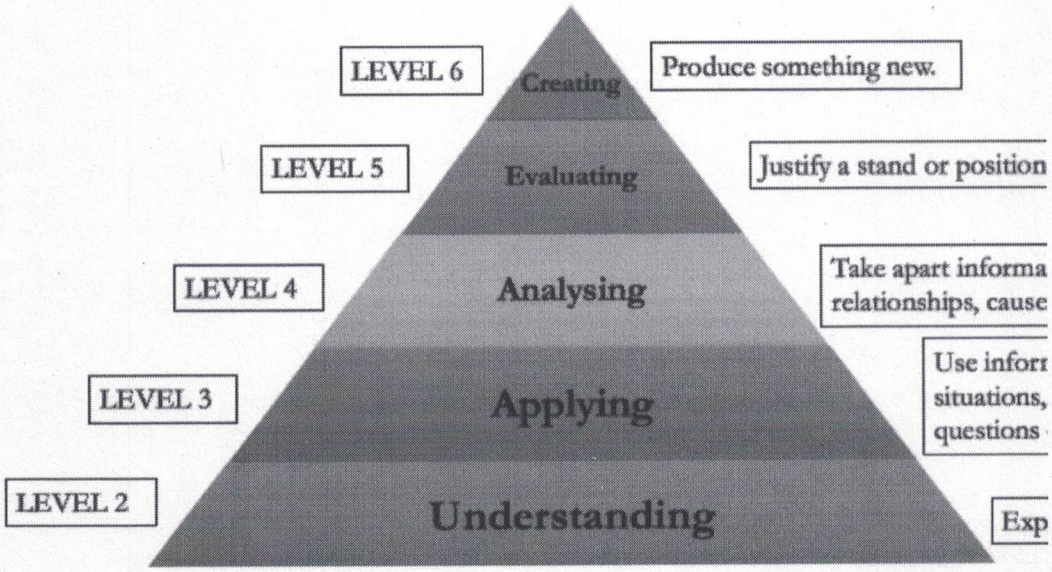
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6. Books(4-10 books):

- Bob Hughes and Mike Cotterell: Software Project, Management latest edition, McGraw-Hill,
- Bruegge and Allen H. Dutoit (2010) Object Oriented Software Engineering – using UML, Third Edition, Prentice Hall,
- Effective Project Management: Traditional, Agile, Extreme by Robert K. Wysocki (Wiley)
- Software Quality Assurance: From Theory to Implementation by Daniel Galin (Pearson)
- Agile Project Management with Scrum by Ken Schwaber (Microsoft Press)
- Project Management: A Systems Approach to Planning, Scheduling, and Controlling by Harold Kerzner (Wiley)

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective

solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Data Mining and Warehouse**

Course Code	MCS 654	**Year/Semester**	I/II
Credit Weightage	3	**Lecture**	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course, tailored for M.Tech students in Information Technology and Computer Science, provides an in-depth study of Data Mining and Data Warehousing. It covers theoretical concepts, practical methodologies, and applications for extracting meaningful patterns from large datasets. Students will explore data preprocessing, association rule mining, classification, clustering, and advanced mining of complex data types. The course also addresses data warehousing, including multidimensional data models, OLAP technology, and data cube development. Through practical lab exercises, students will apply these concepts using tools like Python, SQL, and data mining software, preparing them to tackle real-world data challenges.

2. Course Objectives

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

- Understand the principles, architectures, and functionalities of data mining and data warehousing systems.
- Master data preprocessing techniques, including cleaning, transformation, and reduction, to prepare datasets for analysis.
- Develop proficiency in data mining techniques such as association rule mining, classification, and clustering.
- Explore advanced data mining methods for spatial, multimedia, temporal, text, and web data.
- Design and implement data warehouse systems, including multidimensional models and OLAP operations.
- Apply visualization techniques to represent input data and mined knowledge effectively.
- Evaluate the social, ethical, and technical challenges associated with data mining applications.
- Gain hands-on experience with industry-standard tools and programming languages for data mining and warehousing.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:s

- Articulate the importance, functionalities, and architectures of data mining and data warehousing systems.
- Perform data preprocessing tasks (cleaning, integration, transformation, reduction) to prepare datasets for mining.
- Implement and analyze association rules, classification models, and clustering algorithms using appropriate algorithms.
- Design and query data warehouses using multidimensional models and OLAP operations.
- Apply specialized algorithms to mine complex data types, including spatial, temporal, and text data.
- Use visualization techniques to represent data mining results and knowledge effectively.
- Evaluate the performance of data mining models using metrics like accuracy, precision, and recall.
- Develop practical solutions to real-world data problems using tools like Python, R, or SQL.
- Assess the social and ethical implications of data mining applications

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction **(6 hrs)**

- Introduction to Data Mining
- Importance of Data Mining
- Data Mining functionalities
- Classification of Data Mining Systems
- Data mining architecture
- Major Issues in Data Mining
- Applications of Data Mining
- Social Impacts of Data Mining

Unit 2: Data warehouse **(6 hrs)**

- Introduction to Data Warehouse and OLAP Technology for Data Mining
- Multidimensional data Model
- Data Warehouse Data Model
- Data Warehouse Architecture
- Data Warehouse Implementation
- Development of Data Cube Technology
- From Data warehousing to Data Mining

Unit 3: Data Cleaning **(9 hrs)**

- Matrix-Chain Multiplication
- Elements of Dynamic Programming
- Longest Common Subsequence
- 0/1 Knapsack problem
- Traveling Salesman problem
- Data Preprocessing
- Data cleaning
- Data Integration and Transformation: standardization, normalisation, smoothing, aggregation, generalization

- Data reduction: Data Cube Aggregation, Dimensionality reduction, Numerosity Reduction, Discretization, and Concept Hierarchy Generation
- Data Mining primitives
- Languages and System Architectures
- Concept Description : Characterization and Comparison, Analytical Characterization, Mining Class Comparison
- Representing input data and output knowledge: Visualization techniques
- Guidelines for Successful Data Mining

Unit 4: Association rule in data mining **(8 hrs)**

- Association Rule Mining
- Mining of single-dimensional Boolean association rules
- Multilevel association rules and Multidimensional association rules
- Correlation Analysis
- Constraint-based association Mining.

Unit 5: Classification and Predication **(7 hrs)**

- Basic issues regarding classification and predication
- Classification by Decision Tree
- Bayesian classification
- Classification by back propagation
- Associative classification
- Prediction
- Classifier accuracy

Unit 6: Cluster Analysis **(7 hrs)**

- Cluster Analysis
- Basic issues
- Clustering using partitioning methods
- Hierarchical methods,
- Density-based methods
- Grid-based methods and model-based methods,
- Algorithms for outlier analysis.

Unit 7: Mining Complex Types of Data **(5 hrs)**

- Multidimensional analysis and descriptive mining of complex data objects
- Introduction to spatial mining, multimedia mining, temporal mining, text mining and web mining with related algorithms.

4.2 Laboratory Work **(16 hrs)**

- Install and explore data mining tools (e.g., Weka, RapidMiner, or Python libraries like scikit-learn). Load a sample dataset (e.g., Iris or Titanic) and generate basic summary statistics

- Design a star schema for a retail sales data warehouse. Implement it using SQL and perform OLAP queries (e.g., roll-up, drill-down) using a database tool like PostgreSQL or MySQL.
- Preprocess a noisy dataset (e.g., Kaggle's House Prices dataset) by handling missing values, normalizing data, and discretizing continuous attributes. Use Python libraries for implementation
- Implement the Apriori algorithm in Python to mine association rules from a market basket dataset (e.g., Kaggle's Grocery dataset).
- Develop a decision tree and Bayesian classifier for a dataset (e.g., UCI's Breast Cancer dataset) using scikit-learn. Compare their performance using cross-validation
- Implement K-means and DBSCAN clustering on a dataset (e.g., UCI's Mall Customers dataset). Identify outliers using Isolation Forest. Visualize clusters using Python.
- Perform text mining on a dataset (e.g., Twitter sentiment dataset) using Python's NLTK or spaCy. Extract topics using LDA and visualize word clouds. Optionally, explore web scraping for data collection

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

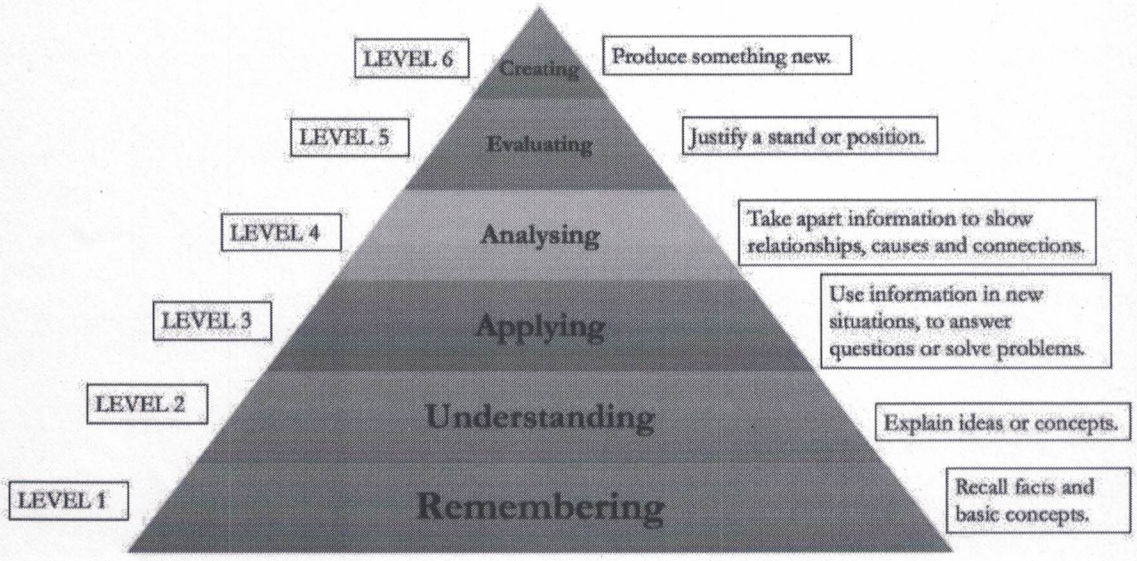
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

- Data Mining Concepts and Techniques by Jiawei Han, Micheline Kamber– Elsevier
- Data Mining by Arun K. Pujari – University Press
- Modern Data Warehousing, Data Mining and Visualization by George M. Marakas –Pearson
- Data Mining by Vikram Puri And P.Radha Krishana –Oxford Press
- Data Warehousing by Reema Theraja –Oxford Press

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue Assess Critique Defend Evaluate Judge Justify

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Project Development**

Course Code	MCS 655	Year/Semester	I/II
Credit Weightage	2	Lecture	N/A
		Tutorial	N/A
		Practical	3 hrs/wk
		Total	48 hrs

1. Course Description

This course provides students with hands-on experience in designing, developing, and managing a complete software or research project. Students will learn project planning, requirement analysis, system design, implementation, testing, and documentation. The course emphasizes teamwork, professional ethics, and the application of modern tools and methodologies to solve real-world problems.

2. Course Objectives

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

- Plan, design, and execute a complete software or research project.
- Apply project management and software engineering principles to organize work effectively.
- Utilize modern development tools and technologies to develop scalable and maintainable solutions.
- Collaborate effectively in teams and communicate project outcomes professionally.
- Document, present, and defend the project results comprehensively.

3. Learning Outcomes

Upon completing this course, Students will be able to:

- Define project objectives, scope, and deliverables clearly.
- Conduct requirement analysis and feasibility studies.
- Design system architecture and select appropriate technologies.
- Develop working prototypes and full-scale project solutions.
- Implement testing strategies, including unit testing, integration testing, and user acceptance testing.
- Prepare professional project reports, presentations, and documentation.
- Apply ethical standards and professional practices in project development

4. Project Implementation Guidelines

4.1 Planning and Scope

- Clearly define project objectives, scope, and expected outcomes.
- Prepare a project proposal including problem statement, motivation, literature survey, methodology, and expected deliverables.
- Break the project into phases: requirement analysis, design, development, testing, and documentation.

4.2 Technology and Tools

- Select appropriate programming languages, frameworks, and platforms.
- Use database management systems if required (SQL/NoSQL).
- Incorporate version control (e.g., Git/GitHub) for collaborative development.
- Consider cloud services or virtualization tools if relevant for deployment or testing.

4.3 Design & Architecture

- Use UML diagrams (Class, Sequence, Activity, Use Case) to describe system design.
- Prepare data flow diagrams (DFD), ER diagrams, and database schema if applicable.
- Clearly define module responsibilities and interfaces.
- Include considerations for scalability, maintainability, and performance optimization.

4.4 Development & Coding Standards

- Follow modular coding practices with proper naming conventions.
- Comment the code adequately for clarity and maintainability.
- Implement error handling and logging mechanisms.
- Follow best practices in security, data handling, and resource management.

4.5 Testing & Quality Assurance

- Perform unit testing, integration testing, system testing, and user acceptance testing (UAT).
- Maintain a test plan with test cases, expected results, and actual results.
- Include performance testing and optimization where relevant.
- Document bugs, fixes, and improvements in a test log.

4.6 Project Deliverables

- Working software or research prototype.
- Source code, well-commented and version-controlled.
- User manual or installation guide if applicable.
- Final report and presentation slides.

5. Project Report Guidelines

The report should be professional, clear, and well-structured. Recommended sections are:

5.1 Front Matter

- Title page with project title, student names, roll numbers, supervisors, department, and university.
- Acknowledgements (optional).
- Abstract (150–250 words summarizing the project objectives, methods, results, and conclusions).

5.2 Table of Contents

- Include list of figures, tables, and abbreviations.

5.3 Introduction

- Background and context of the project.
- Motivation and significance of the problem.
- Objectives and scope of the project.

5.4 Literature Review

- Discuss previous work and related research.
- Highlight gaps or limitations that your project addresses.
- Include citations in standard formats (APA, IEEE, or university guideline).

5.5 Requirement Analysis

- Functional and non-functional requirements.
- System specifications and constraints.
- Use Use Case Diagrams or Requirement Tables where necessary.

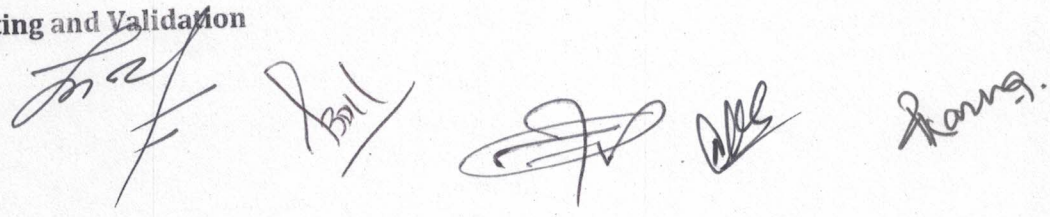
5.6 System Design

- High-level and detailed design of the system.
- Architectural diagrams (UML, DFD, ER diagrams).
- Database design, schemas, and data flow.

5.7 Implementation

- Programming languages, tools, and libraries used.
- Description of modules and workflow.
- Key algorithms, data structures, or methodologies implemented.
- Screenshots, flowcharts, or pseudo-code for clarity.

5.8 Testing and Validation



- Test plan and test cases.
- Testing methods: unit, integration, system, acceptance.
- Observations, bug fixes, and validation results.
- Performance metrics (if applicable).

5.9 Results and Discussion

- Present outcomes of the project.
- Discuss performance, accuracy, efficiency, or usability.
- Compare with baseline or expected outcomes.
- Include charts, graphs, tables, or screenshots for evidence.

5.10 Conclusion and Future Work

- Summarize achievements and lessons learned.
- Limitations of the current work.
- Recommendations and scope for future enhancements.

5.11 References

- List all cited books, papers, websites, and other resources in IEEE/APA format.

5.12 Appendices

- Source code (optional if too large), configuration files, additional diagrams.
- User manual or deployment instructions.

6. General Guidelines

- Page formatting: A4, 1.5 line spacing, 12-pt font, Times New Roman or Arial.
- Figures & Tables: Numbered and properly referenced in the text.
- Plagiarism: Ensure originality; follow university anti-plagiarism policies.
- Presentation: Prepare slides for final defense (10–15 min). Include objective, methodology, results, and conclusion.
- Supervision: Regular meetings with supervisor are mandatory for feedback and guidance.

7. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Project Proposal & Requirement Analysis	10	10	5
	Progress/Milestone Reports	10	10	5
	Implementation/Coding/ Prototype	15	15	7.5
	Practical Demonstration & Testing	10	10	5
	Project Report & Documentation	10	10	5
	Presentation/Seminar/Defense	5	5	2.5
	Total Internal Marks		60	30
Semester-End Examination	Final Project Evaluation	40	40	20

Notes/Guidelines for Evaluation:

- **Project Proposal & Requirement Analysis:** Assessment of clarity of objectives, feasibility, and completeness of requirements.
- **Progress/Milestone Reports:** Regular progress submissions, adherence to timelines, and supervisor feedback.
- **Implementation/Coding/Prototype:** Quality, modularity, correctness, and functionality of code or system.
- **Practical Demonstration & Testing:** Successful demonstration of working system, testing coverage, and handling of edge cases.
- **Project Report & Documentation:** Completeness, clarity, formatting, references, and alignment with implementation.
- **Presentation/Seminar/Defense:** Ability to communicate project objectives, methodology, results, and answer questions confidently.
- **Semester-End Evaluation:** Combined assessment by internal and external examiners of final deliverables and presentation.

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Advanced Computer Architecture**

Course Code	MCS 656	Year/Semester	I/II
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course explores advanced computer architecture, emphasizing parallelism across instruction, data, and thread levels. It introduces methods for efficient parallel program development, explores parallel programming environments, and investigates emerging technologies.

2. Course Objectives

- To introduce foundational concepts of parallel computing and its relevance in modern computer architecture, and performance issues associated with parallel systems
- To explain instruction-level parallelism (ILP) by analyzing techniques such as pipelining, out-of-order execution, and speculative execution to enhance processor performance
- To explore data-level parallelism (DLP) by evaluating vector processing, SIMD architectures, and GPU computing to process large datasets efficiently
- To analyze thread-level parallelism (TLP) and multi-threaded programming models to effectively utilize multi-core and many-core processors
- To evaluate emerging computing trends and their potential impact on future architectures

3. Learning Outcomes

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

- Define key concepts and terminologies related to parallel computing.
- Classify different types of parallel architectures and computational models.
- Illustrate how ILP enhances performance through pipelining and other hardware-level techniques
- Evaluate ILP strategies such as superscalar execution, branch prediction, and dynamic scheduling
- Identify and explain common SIMD (Single Instruction, Multiple Data) instruction set extensions
- Compare DLP techniques across CPUs and GPUs for high-performance computing tasks.
- Explain the principles of multithreading and concurrency in modern processors
- Debug, optimize, and assess the performance of parallel programs in different environments.
- Reflect on how emerging technologies influence future computing system designs and applications

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction (6 hrs)

- Computer Architecture and Organization
- Basic Parallel Processing Architecture, Taxonomy- SISD. MISD, SIMD, MIMD structures
- Serial, Parallel & Concurrent Computation
- Computer System Attributes to Performance
- Ahmdahal's Law and Little's Law

Unit 2: Instruction Level Parallelism (12 hrs)

- Concept and Examples of Data Dependence
- Challenges in Instruction level parallelism realization
- Pipeline hazards and their solutions
- Data forwarding
- Register renaming
- Reordering of instructions
- Out of order execution
- Branch prediction
- Dynamic scheduling
- Limitations of scalar pipelines
- VLIW and superscalar processors

Unit 3: Data Level Parallelism (5 hrs)

- Introduction to Vector architecture
- SIMD instruction set extensions
- Graphics Processing Units architecture.

Unit 4: Thread Level Parallelism (6 hrs)

- Motivation for multicore and many core systems
- Amdahl's law under power constraint
- Challenges in efficient multicore system design
- Cache coherence problem

Unit 5: Parallel Program Development and Environment (12 hrs)

- Parallel Programming Environments; Software Tools and Environments; Y-MP Paragon and CM-5 Environments; Visualization and Performance Tuning
- Synchronization and Multiprocessing Modes; Principles of Synchronization; Multiprocessor Execution Modes; Multitasking on Cray Multiprocessors
- Shared-Variable Program Structures; Locks for Protected Access; Semaphores and Applications; Monitors and Applications
- Message-Passing Program Development; Distributing the Computation; Synchronous Message Passing; Asynchronous Message Passing
- Mapping Programs onto Multicomputer; Domain Decomposition Techniques; Control Decomposition Techniques; Heterogeneous Processing

Unit 6: Emerging Computing Trends (7 hrs)

- Application-specific architectures : ASIP. FPGA and ASIC
- Heterogeneous multicore Platform
- Introduction to Domain specific computing
- Comparison of performances and power consumption of general-purpose processors, DSP, GPU, FPGA, and ASIC

4.2 Laboratory work **(16 hrs)**

The practical portion of this course will consist of numerical exercises from each chapter, enabling students to strengthen their comprehension of theoretical material through organized problem-solving sessions

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

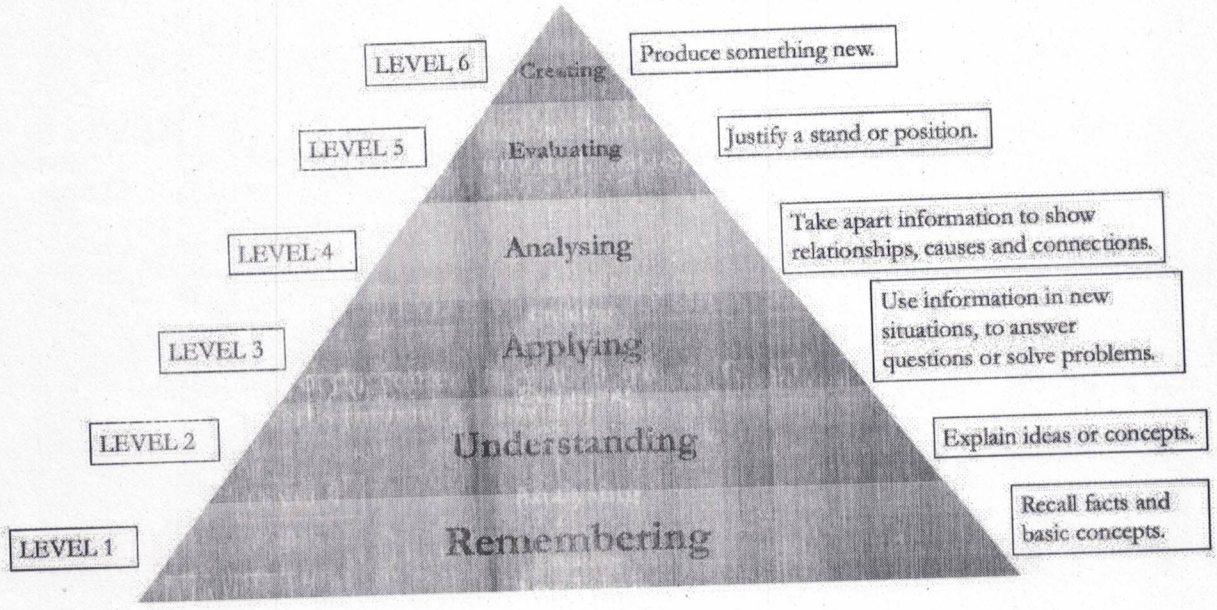
6 Books (4-10 books):

- Kai Hwang and Naresh Jotwani, "Advanced Computer Architecture - Parallelism, Scalability, Programmability", McGraw-Hill International Edition, 2011
- John L. Hennessey and David A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann/ Elsevier, Fifth Edition, 2012
- Dezso Sima, Terence Fountain, Peter Karsuk, "Advanced Computer Architectures: A design space approach", Addison Wesley, 1997
- Kai Hwang and Faye Briggs, "Computer Architecture and Parallel Processing", McGraw-Hill International Edition, 2000
- William Stallings, "Computer Organization and Architecture: Designing for Performance" Tenth Edition, Pearson Education, 2016

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an

improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange *Define* *Describe* *List* *Match* *Name* *Order* *Recall* *Reproduce*

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify *Discuss* *Explain* *Identify* *Report* *Summarise*

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply *Calculate* *Demonstrate* *Interpret* *Show* *Solve* *Suggest*

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* *Investigate*

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Advanced Database Management System**

Course Code	MCS 657	Year/Semester	I/II
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course provides in-depth concepts of advanced concepts of database management system, including enhanced entity-relationship model, object and object-relational databases, file structures, indexing, query processing and optimization, distributed databases, NoSQL databases, Big Data, information retrieval and web search.

2. Course Objectives

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

- To provide students with hands-on experience to design databases using enhanced entity-relationship model
- To familiarize students with object-oriented and object-relational databases
- To provide students knowledge of file structures, indexing, query processing and query optimization
- To familiarize students with different concepts of distributed databases, NoSQL databases, and BigData technologies
- To familiarize students with different concepts of information retrieval and web search

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- design databases using both ER and EER models
- learn and implement object-oriented and object-relational database concepts
- know different storage structures and concepts of indexing and hashing
- learn steps in query processing and importance of query optimization
- implement distributed databases, NoSQL databases, and BigData technologies
- understand concepts of information retrieval and web search

4. Course Details

(48 hrs)

4.1 Theory

(5 hrs)

Unit 1: Enhanced Entity-Relationship (EER) Model

- Entity-Relationship Model Revised
- Subclasses, Superclasses, and Inheritance

- Specialization and Generalization
- Constraints and Characteristics of Specialization and Generalization Hierarchies
- Representing Specialization and Generalization in UML Class Diagrams
- Data Abstraction, Knowledge Representation, and Ontology Concepts

Unit 2: Object and Object-Relational Databases **(6 hrs)**

- Overview of Object Database Concepts
- Object Database Extensions to SQL
- The ODMG Object Model and the Object Definition Language ODL
- Object Database Conceptual Design
- Object Query Language OQL
- Language Binding

Unit 3: File Structures and Indexing **(8 hrs)**

- Heap Files
- Sorted Files
- Hashing Techniques
- Types of Single-Level Ordered Indexes
- Multilevel Indexes
- Dynamic Multilevel Indexes Using B-Trees and B+-Trees
- Indexes on Multiple Keys
- Other Types of Indexes
- Some General Issues Concerning Indexing
- Physical Database Design in Relational Databases

Unit 4: Query Processing and Optimization **(6 hrs)**

- Translating SQL Queries into Relational Algebra and Other Operators
- Query Trees and Heuristics for Query Optimization
- Cost-Based Query Optimization

Unit 5: Distributed Database Concepts **(6 hrs)**

- Distributed Database Concepts
- Data Fragmentation, Replication, and Allocation Techniques for Distributed Database Design
- Types of Distributed Database Systems
- Distributed Database Architectures
- Distributed Catalog Management

Unit 6: NoSQL Databases and BigData **(10 hrs)**

- Introduction to NOSQL Systems
- The CAP Theorem
- Document-Based NOSQL Systems
- NOSQL Key-Value Stores
- Column-Based or Wide Column NOSQL Systems
- NOSQL Graph Databases
- What Is Big Data?
- Introduction to MapReduce

Unit 7: Information Retrieval and Web Search **(7 hrs)**

- Information Retrieval (IR) Concepts
- Retrieval Models
- Types of Queries in IR Systems
- Text Preprocessing
- Inverted Indexing
- Evaluation Measures of Search Relevance
- Web Search and Analysis
- Trends in Information Retrieval

4.2 Laboratory Work **(16 hrs)**

Lab	Practical Title
1	Learn to use drawing tool to draw ER and EER models
2	Implement object-oriented and object-relational databases
3	Learn to create indexes in databases
4	Implement distributed database
5	Learn to use NoSQL database such as MangoDB and big data system such as Hadoop
6	Implement information retrieval and web search concepts

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

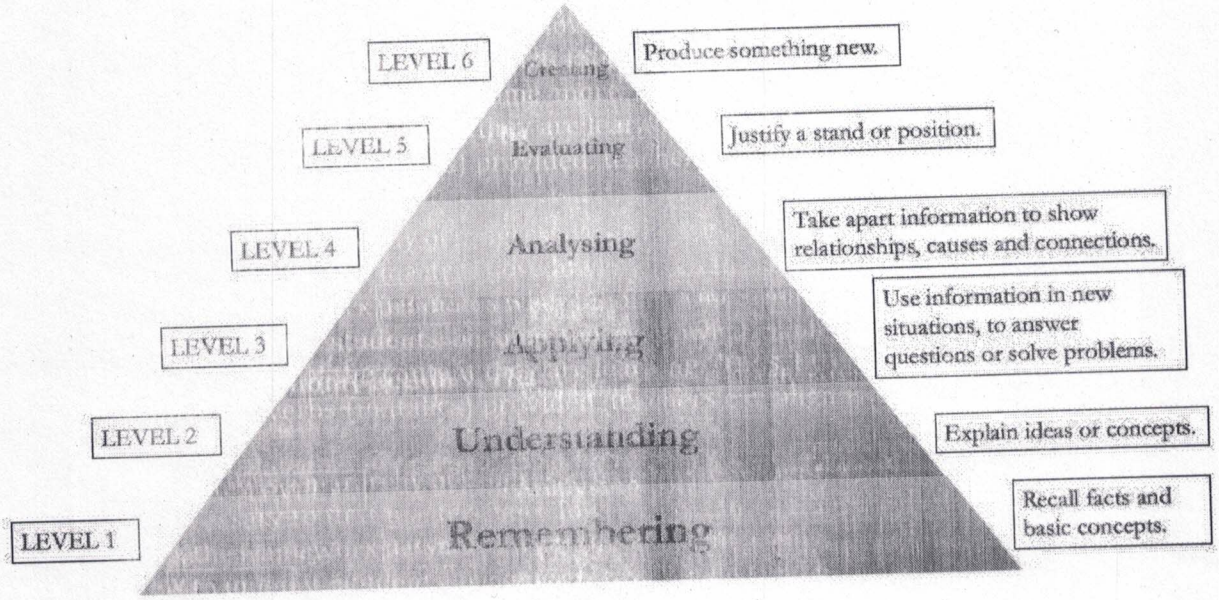
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

- Elmasri and Navathe, Fundamentals of Database Systems, Pearson Education, 7th Edition, 2016.
- Korth, Silberchatz, Sudarshan, Database System Concepts, McGraw-Hill, 7th Edition.
- Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGraw-Hill.
- Peter Rob and Coronel, Database Systems, Design, Implementation and Management,Thomson Learning.
- C.J. Date & Longman, Introduction to Database Systems, Pearson Education.

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* *Investigate*

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Human-Computer Interaction**

Course Code	MCS 658	Year/Semester	I/II
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course emphasizes the growing importance of understanding users and their environments, especially as computer technology becomes ever more pervasive and consumer-centered. At its core, Human-Computer Interaction (HCI) focuses on designing technologies that are secure, easy to use, and efficient—ensuring a meaningful and satisfying experience for users.

2. Course Objectives

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

- To analyze and apply the core components of interactive technology design—including usability, accessibility, and user experience (UX) principles—to create effective digital solutions.
- To evaluate the theoretical frameworks and methodological foundations (e.g., cognitive psychology, design thinking, and iterative prototyping) that guide interface design and user interaction.
- To conduct user-centered research and usability testing to assess interactive artifacts, employing both qualitative and quantitative evaluation techniques.
- To interpret usability data and feedback to diagnose design flaws and propose evidence-based improvements for enhanced user satisfaction and performance.
- To advocate for ethical and equitable design practices, ensuring technologies cater to diverse user needs while addressing societal and cultural impacts.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

- Upon completing this course, Students will be able to:
- understand core HCI principles
- apply user-centered design methods
- analyze user behavior and cognition
- design and prototype interactive systems
- conduct and apply usability evaluations

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction **(6 hrs)**

- Ergonomics and Human Factors
- Foundational Concepts of Interaction Design; Good and poor design; Usability and user experience goals
- Computer-Supported Cooperative Work (CSCW)
- Collaborative Technologies (e.g., Google Docs, Zoom)

Unit 2: Design Lifecycle **(8 hrs)**

- User-Centered Design Concepts
- Participatory Design
- Usability Engineering ; Identifying users; Understanding user requirements, goals, skills, and environment

Unit 3: Psychology of HCI **(8 hrs)**

- Human Perception, Cognition, and Emotion; Attention, memory, emotion,
- User Behavior, Accessibility, and Diversity
- Applying Physical World Knowledge to Digital Interfaces

Unit 4: Methods for Studying Users in Context **(6 hrs)**

- Observation Techniques; Data collection and analysis
- Interviews and Questionnaires ; Types of interviews (structured, unstructured, ...) ; Survey design and analysis

Unit 5: Principles for Usability in UI Design **(6 hrs)**

- Usability Guidelines ; Navigation, display organization, data entry
- Design Principles ; Interaction styles, preventing errors, automation
- Theories of Interface Design ; GOMS, stages-of-action, widget-level theories

Unit 6: Prototyping Methods **(6 hrs)**

- Low- and High-Fidelity Prototyping
- Conceptual Design Techniques; Scenarios, models, and iterations
- From Design to Implementation

Unit 7: Evaluation Methods **(6 hrs)**

- Expert Reviews; Heuristic evaluation, cognitive walkthroughs, ...
- Usability Testing
- Evaluation During Active Use; Surveys, focus groups, data logging, online feedback

Unit 8: Mini-Project Development and Integration **(2 hrs)**

- Application of Full HCI Lifecycle; From design to prototype to evaluation

4.2 Laboratory Work **(16 hrs)**

Students are required to conceptualize and develop a project that applies Human-Computer Interaction principles and techniques covered throughout the course. The evaluation will be based

on the relevance and originality of the project topic, its practical significance, design, and technical quality, depth of HCI integration, user-centered approach, report clarity, and effectiveness of the final presentation.

5 Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

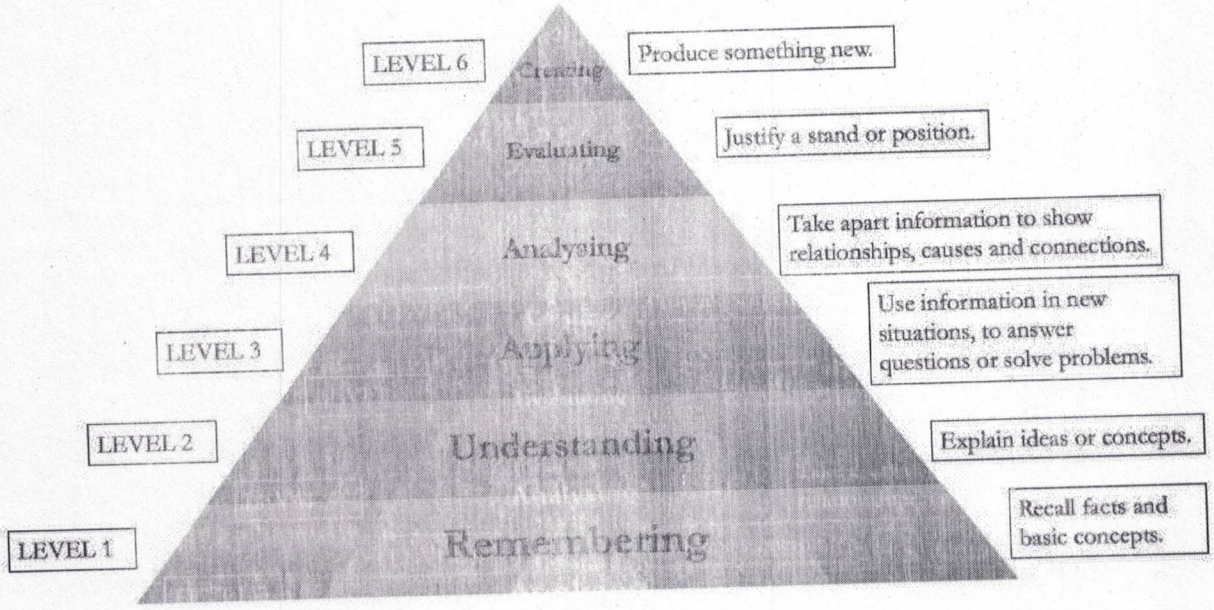
Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6 Books (4-10 books):

- Helen Sharp, Jenny Preece, and Yvonne Rogers. Interaction Design: Beyond Human-Computer Interaction. "In John Wiley", 2011.
- Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs. Designing the User Interface: Strategies for Effective Human-computer Interaction, "Pearson Education", 2017.
- Donald A. Norman. The Design of Everyday Things. "Basic Books", 2013.

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange *Define* *Describe* *List* *Match* *Name* *Order* *Recall* ***Reproduce***

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify *Discuss* *Explain* *Identify* *Report* *Summarise*

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply *Calculate* *Demonstrate* *Interpret* *Show* *Solve* ***Suggest***

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* ***Investigate***

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* ***Justify***

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Machine Learning**

Course Code	MCS 701	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

The course provides a comprehensive overview of Machine Learning, aiding graduate students in grasping fundamental concepts. It starts with foundational knowledge essential for understanding various machine learning algorithms, including Supervised and Unsupervised methods, the Bias-Variance Trade-Off, Overfitting, and Underfitting. The course delivers an example-based approach, offering insight into the practical application of machine learning techniques to real-world problems through case studies and related projects.

2. Course Objectives

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

- To introduce fundamental concepts and techniques for prediction and classification.
- To enhance understanding of different problem-solving techniques with appropriate machine learning algorithms.
- To provide insight into the practical application of machine learning algorithms in real-world problem-solving scenarios.

3. Learning Outcomes

Upon completing this course, Students will be able to:

- Students will acquire foundational knowledge, understanding key principles such as the BiasVariance Trade-Off and the concepts of overfitting and underfitting.
- Students will apply their knowledge to solve complex problems, using case studies and projects as vehicles for practical application.
- Students will evaluate the effectiveness of various machine-learning techniques, critically assessing their performance and suitability for different tasks.
- Students will develop a comprehensive understanding of statistical and machine learning principles and the ability to apply them proficiently in diverse real-world contexts.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction (8 hours)

- Introduction to Machine Learning
- Machine Learning Methods
- The Bias-Variance Trade-Off
- Overfitting and Underfitting

Unit 2: Linear Regression (10 hours)

- Linear Regression
- Multiple Linear Regression
- Qualitative Predictors
- Gradient Descent

Unit 3: Resampling and Linear Model Selection (10 hours)

- The validation Set Approach
- Leave-One-Out Cross Validation
- K-Fold Cross Validation
- Bias-Variance Trade-off for k-Fold Cross Validation

Unit 4: Machine Learning Algorithms (10 hours)

- Naïve Bayes
- Random Forest
- Support Vector Machines
- K-means and Fuzzy C-means

Unit 5: Machine Learning Applications and Case Studies (10 Hours)

- Machine learning Applications and Case studies
- Project Works

4.2 Laboratory Work (16 hrs)

- Real-life applications for statistical learning.

  (a) Describe three real-life applications in which *classification* might be useful. Describe the response, as well as the predictors.
  (b) Describe three real-life applications in which *regression* might be useful. Describe the response, as well as the predictors.
  (c) Describe three real-life applications in which *cluster analysis* might be useful.

- Download the Auto dataset from the textbook website, as this question involves the use of simple linear regression on the Auto dataset. .

  a. Use the sm.OLS() function to perform a simple linear regression with mpg as the response and horsepower as the predictor. Use the summarize () function to print the results. Comment on the
  b. output. For example:

     i. Is there a relationship between the predictor and the response?
     ii. How strong is the relationship between the predictor and the response?
     iii. Is the relationship between the predictor and the response positive or negative?
     iv. What is the predicted mpg associated with a horsepower of 98?
     v. What are the associated 95 % confidence and prediction intervals?

  c. Plot the response and the predictor in a new set of axes ax. Use the ax.axline() method or the abline () function defined in the lab to display the least squares regression line.
  d. Produce some of diagnostic plots of the least squares regression fit as described in the lab. Comment on any problems you see with the fit.

- Explore Cross-Validation Techniques Using Machine Learning Tools, Focusing on Chapter 3.
- Implement a machine-learning algorithm covered in Chapter 4.

5. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6. Books(4-10 books):

- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor. An Introduction to Statistical Learning: with Applications in Python. New York :Springer, 2023
- Jason Brownlee: Master Machine Learning Algorithms: Discover How they Work and Implement Them From Scratch. 2016

- Machine Learning. Tom M. Mitchell. McGraw-Hill International Editions. Computer Science Series, 1997.
- Machine Learning : The Art of Science and Algorithms that Make Sense of Data by Peter Flach

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Advanced Compiler Design**

Course Code	MCS 702	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description:

An in-depth study of compiler backend design for high-performance architectures. Basic topics include control-flow and dataflow analysis, optimization, instruction scheduling, register allocation and JIT. Advanced topics include memory dependence analysis, dynamic compilation, and runtime systems. The focus is backend compilation, thus a familiarity with both computer architecture and compilers is recommended.

2. Course Objectives

- To understand the principles of backend compilation process.
- To understand the method of optimization and runtime environment
- To be able to design and implement an advanced compiler technique within the LLVM compiler infrastructure
- To deliver latest research article under the advanced compiler design topics

3. Learning Outcomes

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

- Explain the overall structure and architecture of advanced compilers, including the role of each phase in the compilation process.
- Perform semantic analysis including type checking, scope management, and intermediate code generation using various intermediate representations.
- Apply optimization techniques at local and global levels to improve the efficiency of generated code through data flow analysis and control flow graph transformations.
- Generate optimized target **code** considering machine architecture constraints, including efficient instruction selection and register allocation strategies.
- Analyze and implement advanced optimization methods such as interprocedural analysis, profile-guided optimization, and just-in-time compilation.
- Design and manage runtime systems, including memory management and garbage collection techniques suitable for modern programming languages.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction and Overview of Compiler Design (4 hrs)

- Role and structure of advanced compilers
- Review of phases in compiler design
- Intermediate representations (IR)
- Compiler frameworks and architecture
- Tools and environments (LLVM, GCC, ANTLR, etc.)

Unit 2: Lexical Analysis and Syntax Analysis (6 Hrs)

- Advanced lexical analysis techniques
- Regular expressions and automata revisited
- Lexical analyzer generators
- Context-free grammars and parsing strategies
- LL(k), LR(k), LALR parsing
- Error detection and recovery in parsing
- Syntax-directed translation

Unit 3: Semantic Analysis and Intermediate Code Generation (6 Hrs)

- Symbol tables and scope management
- Type checking and type inference
- Attribute grammars
- Intermediate representations: Abstract Syntax Trees (AST), Three-Address Code, Static Single Assignment (SSA) form
- Translation schemes and syntax-directed translation

Unit 4: Optimization Techniques (10 Hrs)

- Local and global optimization
- Data flow analysis: reaching definitions, live variables, available expressions
- Control flow graphs and dominators
- Loop optimizations: loop unrolling, loop fusion, induction variable elimination Machine-independent optimizations
- Introduction to machine-dependent optimizations
- Profile-guided optimizations
- Just-in-time (JIT) compilation techniques
- Speculative optimizations and deoptimization

Unit 5: Code Generation and Register Allocation **(8 Hrs)**

- Target machine architecture considerations
- Instruction selection and scheduling
- Register allocation strategies: graph coloring, linear scan
- Peephole optimizations
- Handling procedure calls and parameter passing
- Code generation for modern architectures

Unit 6: Runtime Systems and Garbage Collection **(8 Hrs)**

- Memory management strategies
- Stack and heap management
- Garbage collection algorithms: reference counting, mark-and-sweep, generational GC
- Real-time and concurrent garbage collection
- Exception handling and runtime support

Unit 7: Research Trends and Case Studies **(8 Hrs)**

- Recent research topics in compiler design
- Machine learning in compiler optimization
- Case studies of modern compilers (LLVM, GCC, Java HotSpot)
- Future directions: quantum compilers, neural compilers
- Project presentations and discussions

5. Laboratory Work **(16 hrs)**

- Implement a lexical analyzer for a subset of a programming language (e.g., arithmetic expressions or a mini-language) using tools like Flex or ANTLR.
- Design and implement a parser using LL(k) or LR(k) parsing techniques for the same language subset. Error detection and recovery can be integrated.
- Implement data flow analysis algorithms (reaching definitions, live variable analysis) and use them to perform simple optimizations like dead code elimination or constant propagation on intermediate code.
- Generate target code for a simple RISC-like architecture from the intermediate code. Implement a register allocation algorithm such as graph coloring or linear scan for efficient register usage.
- Literature survey and presentation on a recent compiler research topic (e.g., machine learning in compiler optimizations).
- Final project: Design and implement an advanced compiler component or optimization technique integrating multiple concepts from the course.

6. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

7. Books(4-10 books):

- Advanced Compiler Design and Implementation by Steven Muchnick *Publisher:* Morgan Kaufmann
- Engineering a Compiler (2nd Edition) by Keith Cooper, Linda Torczon, *Publisher:* Morgan Kaufmann
- Modern Compiler Implementation in C/Java/ML by Andrew W. Appel, *Publisher:* Cambridge University Press
- The Definitive ANTLR 4 Reference by Terence Parr, *Publisher:* Pragmatic Bookshelf
- LLVM Essentials by Suyog Sarda, Mayur Pandey, *Publisher:* Packt Publishing
- Garbage Collection: Algorithms for Automatic Dynamic Memory Management by Richard Jones, Rafael Lins, *Publisher:* Wiley

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Distributed and Parallel System**

Course Code	MCS 703	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course introduces the principles, architectures, and programming models of distributed and parallel computing systems. Topics include parallel algorithms, multithreading, message-passing (MPI), shared-memory models (OpenMP), distributed systems design, consistency models, fault tolerance, and scalability. Students will gain hands-on experience through programming assignments and projects.

2. Course Objectives

By the end of this course, students will:

- To understand the fundamental challenges in distributed and parallel systems (e.g., concurrency, synchronization, communication latency, fault tolerance).
- To learn parallel programming paradigms (multithreading, GPU computing, MPI, OpenMP) and their trade-offs.
- To explore distributed system architectures (client-server, peer-to-peer, cloud computing)

3. Learning Outcomes

After completing this course, students will be able to:

- Familiarize with models and architectures of parallel and distributed computing and related theoretical background
- Explain the differences between shared-memory (OpenMP) and message-passing (MPI) models,Compare synchronous vs. asynchronous communication in distributed systems.
- Analyze scalability and performance bottlenecks in parallel algorithms.

4. Course Details

4.1 Theory **(48 hrs)**

Unit 1: Introduction (6 hrs)

- Distributed Computing Basics
  Definition and importance of Distributed Computing, Distributed systems versus parallel systems, Characteristics of distributed systems (scalability, fault tolerance, transparency), Challenges (concurrency, partial failures, consistency), Models of distributed systems (client-server, peer-to-peer, microservices)
- Parallel Computing Basics
  Definition and need for parallelism, Flynn's Taxonomy (SISD, SIMD, MISD, MIMD), Types of parallelism (data, task, pipeline), Types of parallel architectures (shared memory, distributed memory, GPU computing)

Unit 2: Model and Architecture (6 hrs)

- Parallel Programming Models
  Shared memory (threads, OpenMP), Message passing (MPI), Data parallelism (CUDA, OpenCL), MapReduce and Hadoop
- Distributed System Architectures
  Cluster computing, Grid computing, Cloud computing (IaaS, PaaS, SaaS), Edge and fog computing

Unit 3: Processes and Communication (8 hrs)

- Threads
  Introduction to Threads, Threads in Distributed Systems
- Virtualization
  The Role of Virtualization in Distributed Systems, Architectures of Virtual Machines
- Clients
  Networked User Interfaces, Client-Side Software for Distribution Transparency
- Servers
  General Design Issues, Server Clusters, Managing Server Clusters
- Communication Fundamentals
  Layered Protocols, Types of Communication
- Remote Procedure Call
  Basic RPC Operation, Parameter Passing
- Message Oriented Communication
  Message-Oriented Transient Communication, Message-Oriented Persistent Communication

Unit 4: Synchronization (8 hrs)

- Clock Synchronization
  Physical Clocks, Global Positioning System, Clock Synchronization Algorithms
- Logical Clocks
  Lamport's Logical Clocks, Vector Clocks
- Mutual Exclusion: Centralized Algorithm, Decentralized Algorithm, Distributed Algorithm, Token Ring Algorithm
- Global Positioning Of Nodes
- Election Algorithms : Traditional Election Algorithms, Elections in Wireless Environments, Elections in Large-Scale Systems

Unit 5: Fault Tolerance **(6 hrs)**

- Introduction To Fault Tolerance
  Basic Concepts, Failure Models, Failure Masking by Redundancy
- Process Resilience
  Design Issues, Failure Masking and Replication, Agreement in Faulty Systems
  Failure Detection
- Reliable Client-Server Communication
  Point-to-Point Communication, RPC Semantics in the Presence of Failures

Unit 6: Security **(6 hrs)**

- Introduction to Security: Security Threats, Policies, and Mechanisms, Design Issues, Cryptography
- Access Control: General Issues in Access Control, Firewalls, Secure Mobile Code, Denial of Service

Unit 7: Future Trends and Emerging Technologies **(8 hrs)**

- Distributed Storage Systems: Distributed file systems (HDFS, Google File System), Distributed databases (Cassandra, MongoDB), Object storage (Amazon S3)
- Edge Computing and IoT: Edge vs cloud computing, Challenges in distributed IoT networks
- Quantum Computing and Distributed Models: Basics of quantum computing, Quantum algorithms and their distributed nature

5. Laboratory Work **(16 hrs)**

A. Multithreading (Pthreads)

1. Matrix Multiplication with Threads
   - Write a C program using Pthreads to multiply two matrices.
   - Divide rows among threads and compare performance with a single-threaded version.
2. Producer-Consumer Problem
   - Implement a bounded-buffer solution using threads and mutexes/semaphores.
3. Parallel Merge Sort
   - Split the array into subarrays, sort them in separate threads, and merge the results.

B. Multiprocessing (Fork, IPC)

4. Process-Based Word Count
   - Use fork() to create child processes that count words in different sections of a file.
   - Aggregate results using shared memory or pipes.
5. Message Passing with Pipes/FIFOs
   - Implement a client-server model where the client sends commands via a pipe/FIFO, and the server executes them.
6. Shared Memory for Inter-Process Communication
   - Create two processes: one writes data to shared memory, and the other reads it.

C. Distributed Computing (Sockets, RPC)

7. TCP/UDP Chat Application
   - Build a simple chat server and client using sockets (one-to-one or multicast).
8. Remote File Reader
   - A client sends a filename to a server via sockets, and the server returns the file's contents.
9. Distributed Prime Number Checker
   - The client sends a number to multiple server nodes; each checks if it's prime and returns the result.

D. Synchronization & Fault Tolerance

10. Lamport's Logical Clock

6. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

7. Books(4-10 books):

- "Distributed Systems: Principles and Paradigms" (2nd Ed.)Andrew S. Tanenbaum, Maarten van Steen
- "Distributed Systems: Concepts and Design" (5th Ed.) George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair
- Introduction to Parallel Computing, Second EditionAnanth Grama, Anshul Gupta, George Karypis, Vipin Kumar
- Parallel Programming in C with MPI and OpenMP, Michael J Quinn

Lumbini Technological University
Est: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Remote Sensing**

Course Code	MCS 706	Year/Semester	II/I
Credit Weightage	3	Lecture	2 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course explores the principles, technologies, and applications of remote sensing (RS) for Earth observation. Students will learn to acquire, process, and analyze satellite/airborne imagery using digital tools, with a focus on environmental monitoring, urban planning, and disaster management. The course integrates theoretical foundations (e.g., electromagnetic spectrum, sensor physics) with hands-on labs (e.g., image classification, change detection) using software like QGIS, Google Earth Engine, and ENVI (Environment for Visualizing Images).

2. Course Objectives

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

- Analyze the electromagnetic spectrum and its interaction with Earth surface for RS applications.
- Evaluate sensor types (optical, SAR, LiDAR) and platforms (satellites, UAVs) for diverse use cases.
- Process and classify satellite imagery using preprocessing, enhancement, and machine learning techniques.
- Apply RS data to real-world problems (e.g., NDVI for agriculture, SAR for flood mapping).

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Remember key RS concepts (e.g., spectral signatures, resolution types).
- Apply radiometric/geometric corrections to raw imagery.
- Analyze image statistics (histograms, PCA) for feature extraction.
- Evaluate classification accuracy using confusion matrices/Kappa coefficients.
- Create GIS-integrated RS projects (e.g., land-use change maps).

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Fundamentals of Remote Sensing (10 hrs)

- Introduction : Definition, history, and applications (agriculture, forestry, disasters)
- Electromagnetic Spectrum : VIS, NIR, SWIR, TIR, microwave bands.
- Energy-Matter Interaction : Reflection, absorption, spectral signatures.
- Resolution Types: Spatial, spectral, temporal, radiometric trade-offs.

Unit 2: Sensors and Platforms (8 hrs)

- Optical Sensors : Multispectral (Landsat, Sentinel-2) vs. Hyperspectral.
- Active Sensors : SAR (Sentinel-1), LiDAR (topography, forestry).
- Platforms : Satellites (geo/polar-orbiting), UAVs, and aerial photogrammtry.

Unit 3: Image Preprocessing (8 hrs)

- Radiometric Correction : Atmospheric scattering (Dark Object Subtraction).
- Geometric Correction : Georeferencing, orthorectification.
- Image Fusion : Pan-sharpening, NDVI calculation.

Unit 4: Image Classification (8 hrs)

- Supervised: SVM, Random Forest.
- Unsupervised: K-Means, ISODATA.
- Accuracy Assessment: Confusion matrices, Kappa coefficient.

Unit 5: Advanced Applications of Remote Sensing (8 hrs)

- Change Detection: Time-series analysis (urban sprawl, deforestation).
- Thermal RS: Land surface temperature monitoring.
- Disaster Management: Flood mapping (SAR), wildfire detection (NIR).

Unit 6: Emerging Trends in Remote Sensing (6 hrs)

- AI/ML in RS: Deep learning for object detection.
- CubeSats and Hyperspectral Cubes.

4.2 Laboratory Work (16 hrs)

Lab	Practical Title	Hours
1	Download/visualize Optical Remote Sensing datasets	2 hrs
2	Using Spectral Profiles to Distinguish Land Cover Types in Remote Sensing	2 hrs
3	Band Ratio and Spectral Indices Analysis for Land Use and Land Cover (LULC)	2 hrs
4	Supervised classification (urban vs. forest)	2 hrs
5	SAR-based flood mapping	2 hrs
6	Change detection (2010–2025 land cover)	2 hrs
7	Thermal image analysis (urban heat islands)	2 hrs
8	AI-Driven Object Detection in Remote Sensing Using Optical Imagery	2 hrs

For each lab work, students are supposed to implement the algorithm with performance analysis.

5. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

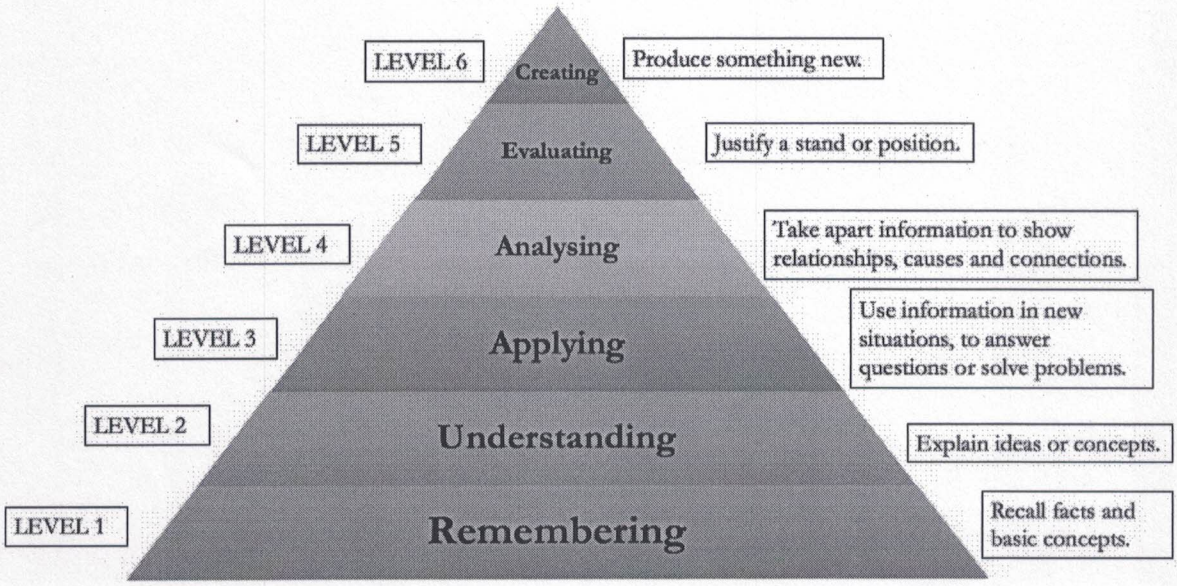
6. Books (4-10 books):

1. Lillesand, T., Kiefer, R. W., & Chipman, J. (2015). Remote Sensing and Image Interpretation. Wiley.
2. Jensen, J. R. (2015). Introduction to Remote Sensing. CRC Press.
3. Campbell, J. B., & Wynne, R. H. (2011). Introduction to Remote Sensing. Guilford Press.
4. Schott, J. R. (2007). Remote Sensing: The Image Chain Approach. Oxford University Press.
5. Mather, P. M., & Koch, B. (2011). Computer Processing of Remotely-Sensed Images: An Introduction. Wiley.
6. Pettorelli, N. (2013). The Normalization of Remote Sensing Data. Wiley-Blackwell.
7. Turner, W., J. L. T., & Spector, S. S. (2015). Free and Open Access Data for Remote Sensing Applications. Springer.
8. Kramer, G. (2002). Observing the Earth and the Universe: The Relevance of Remote Sensing for Global Change Research. Springer.
9. **ESA** (2023). *Sentinel Application Guides* (Open Access)

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and

develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks, using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange *Define* *Describe* *List* *Match* *Name* *Order* *Recall* *Reproduce*

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify *Discuss* *Explain* *Identify* *Report* *Summarise*

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply *Calculate* *Demonstrate* *Interpret* *Show* *Solve* *Suggest*

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse *Appraise* *Compare* *Contrast* *Distinguish* *Explore* *Infer* *Investigate*

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Natural Language Processing**

Course Code	MCS 707	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course provides a comprehensive introduction to Natural Language Processing (NLP), covering fundamental concepts, techniques, and applications. It explores human language processing, computational models, and their real-world applications, with a focus on both theoretical foundations and practical implementations. The syllabus is divided into 6 units, addressing morphology, parsing, semantics, and modern NLP applications.

2. Course Objectives

The specifics objectives of the course are:

- To understand the fundamentals of human languages, NLP paradigms, and applications.
- To explore morphology, its diversity in Indian languages, and computational models for morphological analysis.
- To study word-level processing, shallow parsing, and statistical models for NLP tasks.
- To understand syntactic parsing theories, algorithms, and ambiguity resolution.
- To explore semantic analysis, lexical networks, and word sense disambiguation.
- To study advanced NLP applications with a focus on Web 2.0 and multilinguality.

3. Learning Outcomes

Upon successful completion of the course, learners will be able to:

- Explain the structure and challenges of human language processing.
- Describe the phases and applications of NLP.
- Compare different NLP processing paradigms.
- Understand morphological structures and their diversity in Indian languages.
- Implement FSM and FST for morphological analysis.
- Analyze techniques for automatic morphology learning.
- Apply shallow parsing techniques for text analysis.
- Implement NER using statistical models.
- Evaluate and compare HMM, MaxEnt, SVM, and CRF for NLP tasks.
- Implement constituency and dependency parsing algorithms.

- Analyze and resolve syntactic ambiguities.
- Develop robust parsing systems for noisy text.
- Build and utilize lexical knowledge networks like WordNet.
- Implement WSD algorithms for monolingual and multilingual settings.
- Analyze metaphors and coreferences in text.
- Develop sentiment analysis and text entailment systems.
- Implement scalable machine translation and CLIR systems.
- Evaluate NLP techniques for Web 2.0 applications.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction to NLP and Human Language (6 hrs)

- Introduction to human languages: Structure, ambiguity, and complexity
- NLP processing paradigms: Rule-based, statistical, and neural approaches
- Phases in NLP: Tokenization, morphological analysis, syntactic parsing, semantic analysis, and discourse processing
- Applications of NLP: Chatbots, machine translation, sentiment analysis, question answering
- Challenges in NLP: Ambiguity, multilinguality, and noisy text processing

Unit 2: Morphology and Word Forms (10 hrs)

- Morphology fundamentals: Morphemes, inflection, derivation
- Morphological diversity of Indian languages (e.g., Hindi, Tamil, Bengali)
- Morphology paradigms: Inflectional, agglutinative, and fusional languages
- Finite State Machine (FSM) based morphology
- Finite State Transducers (FST) for morphological analysis
- Automatic morphology learning: Supervised and unsupervised approaches

Unit 3: Words, Shallow Parsing, and Statistical Models (10 hrs)

- Shallow parsing: Chunking and part-of-speech tagging
- Named Entity Recognition (NER)
- Statistical models: N-grams, smoothing techniques (Laplacian, Kneser-Ney)
- Entropy and perplexity in language modelling
- Hidden Markov Models (HMM) for sequence labelling
- Maximum Entropy (MaxEnt) models
- Support Vector Machines (SVM) and Conditional Random Fields (CRF)

Unit 4: Syntactic Structures and Parsing (8 hrs)

- Theories of parsing: Constituency and dependency parsing
- Parsing algorithms: CYK, Earley, and chart parsing
- Robust parsing for noisy text (e.g., web documents)
- Hybrid parsing: Combining rule-based and probabilistic approaches
- Scope ambiguity and attachment ambiguity resolution

- Parsing for Indian languages: Challenges and techniques

Unit 5: Semantics and Meaning Representation **(8 hrs)**

- Lexical knowledge networks: WordNet theory and Indian language WordNets
- Multilingual dictionaries and cross-lingual semantics
- Semantic roles and thematic relations
- Word Sense Disambiguation (WSD): Supervised, unsupervised, and knowledge-based approaches
- WSD in multilingual settings
- Metaphor detection and coreference resolution

Unit 6: Advanced NLP Applications and Web 2.0 **(6 hrs)**

- Sentiment analysis: Techniques and applications
- Text entailment: Recognizing textual entailment
- Machine translation: Statistical, neural, and scalable approaches
- Question answering in multilingual settings
- Cross-Lingual Information Retrieval (CLIR)
- NLP for Web 2.0: Social media analysis, scalability, and robustness.

5. Laboratory Work **(16 hrs)**

1. **Tokenization and Ambiguity Analysis**

- Implement a rule-based tokenizer for English and an Indian language (e.g., Hindi or Tamil) using Python. Compare the challenges faced due to language-specific ambiguities (e.g., word boundaries in Hindi).
- Test the tokenizer on a dataset containing ambiguous sentences and report the errors. Suggest improvements to handle ambiguity.

2. **NLP Pipeline Implementation**

- Develop a basic NLP pipeline using NLTK or spaCy that performs tokenization, morphological analysis, syntactic parsing, and semantic analysis on a small English text corpus.
- Extend the pipeline to process a paragraph in an Indian language (e.g., Bengali) and document the challenges encountered.

3. **Chatbot Prototype**

- Create a simple rule-based chatbot using Python that responds to user queries about a specific domain (e.g., university information).
- Integrate a statistical or neural-based approach (e.g., TF-IDF or a pre-trained transformer like BERT) to improve response accuracy.

4. **Finite State Transducer (FST) for Morphological Analysis**

- Implement a Finite State Transducer using a library like pyfst or openfst to perform morphological analysis for an inflectional language (e.g., Hindi).
- Test the FST on a set of Hindi verbs and nouns, evaluating its accuracy in generating correct word forms.

5. **Morphological Analyzer for Indian Languages**

- Develop a rule-based morphological analyzer for an agglutinative Indian language (e.g., Tamil) using Python.
- Compare its performance with an unsupervised morphology learning approach using a tool like Morfessor on the same dataset.

6. **Supervised Morphology Learning**

- Train a supervised model (e.g., using scikit-learn or PyTorch) to predict morpheme boundaries in a dataset of Bengali words.
- Evaluate the model's performance using precision, recall, and F1-score, and discuss challenges in handling morphological diversity.

7. **Part-of-Speech (POS) Tagging with HMM**

- Implement a Hidden Markov Model (HMM) for POS tagging using Python and a library like hmmlearn. Train it on an English or Indian language (e.g., Hindi) tagged corpus.
- Evaluate the model's accuracy and compare it with a pre-trained POS tagger from NLTK or spaCy.

8. **Named Entity Recognition (NER) with CRF**

- Develop a Conditional Random Field (CRF) model using sklearn-crfsuite for NER on a dataset containing English and Hindi news articles.
- Test the model's ability to identify entities (e.g., person, organization) and report performance metrics.

9. **N-gram Language Model with Smoothing**

- Build an N-gram language model (bigram or trigram) with Kneser-Ney smoothing using Python. Train it on a corpus of social media text in English and an Indian language (e.g., Tamil).
- Compute perplexity scores and compare the model's performance across languages.

10. **Constituency Parsing with CYK Algorithm**

- Implement the Cocke-Younger-Kasami (CYK) parsing algorithm in Python for a context-free grammar (CFG) designed for simple English sentences.
- Test the parser on a small set of ambiguous sentences and analyze its ability to handle scope ambiguity.

11. **Dependency Parsing for Indian Languages**

- Use spaCy or Stanford NLP to perform dependency parsing on a corpus of Hindi or Bengali sentences.
- Evaluate the parser's accuracy and discuss challenges in handling free word order in Indian languages.

12. **Robust Parsing for Noisy Text**

- Develop a hybrid parsing approach combining rule-based and probabilistic methods to parse noisy text (e.g., tweets in English and Hindi).
- Compare the hybrid parser's performance with a standard parser on the same dataset

13. **Word Sense Disambiguation (WSD)**

- Implement a supervised WSD system using scikit-learn or PyTorch for English words using WordNet as the sense inventory.
- Extend the system to handle an Indian language WordNet (e.g., Hindi WordNet) and evaluate its performance on a multilingual dataset.

14. **Coreference Resolution**

- Use a library like neuralcoref or AllenNLP to implement coreference resolution on an English news article.
- Test the system on a Hindi or Tamil text and document challenges in resolving pronouns across languages.

15. **Semantic Role Labeling (SRL)**

- Develop a system for semantic role labeling using a pre-trained model (e.g., from AllenNLP) on an English dataset.
- Adapt the system for an Indian language (e.g., Bengali) and report on the challenges of cross-lingual SRL.

16. **Sentiment Analysis on Social Media**

- Build a sentiment analysis model using a transformer-based approach (e.g., BERT via Hugging Face) for English and Hindi tweets.
- Evaluate the model's performance on a labeled dataset and discuss scalability for Web 2.0 applications.

17. **Statistical Machine Translation**

- Implement a phrase-based statistical machine translation system using a tool like Moses for English-to-Hindi translation.
- Compare its performance with a neural machine translation model (e.g., using Fairseq) on a parallel corpus.

18. **Cross-Lingual Information Retrieval (CLIR)**

- Develop a CLIR system that retrieves Hindi documents based on English queries using a multilingual embedding model (e.g., LASER or mBERT).
- Evaluate the system's precision and recall on a small dataset of bilingual documents.

6. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

7. Books(4-10 books):

- Jurafsky, D., & Martin, J. H. *Speech and Language Processing* (3rd ed.), Chapters 3, 4, 5, 8, 10, 12–14, 16–20.
- NLTK documentation: https://www.nltk.org
- spaCy documentation: https://spacy.io
- James A.. Natural language Understanding 2e, Pearson Education, 1994
- Bharati A., Sangal R., Chaitanya V.. Natural language processing: a Paninian perspective, PHI, 2000

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Big Data Analytics**

Course Code	MCS 708	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course will provide students with a broad introduction to Big Data technologies including Hadoop based architectures, data ingestion, data transformation, data management, analytics and predictive analytics for manipulating and discovering perception. The major topics include Hadoop, HDFS, MapReduce, NoSQL and Big data & Machine learning. The course also includes case studies and applications.

2. Course Objectives

- To display a sound understanding of the principles of organisation, validation, transformation and analyse large volumes of data on specialized platforms (Big Data) from various data sources files, databases, server logs, etc.
- To use the Big Data platform ecosystem for processing Big Data.
- To understand the potential use in a corporate environment.
- To comprehend the advantages and limitations of Big Data technologies, including predictive analytics and build the confidence to interpret data as insights to drive organisational success.

3. Learning Outcomes

- Ability to analyze complex data sets using big data tools and techniques.
- Applying different data mining and statistical methods for extracting big data from different sources.
- Ability to apply big data analytics techniques to real-world business and decision-making processes.
- Discuss on database management systems and data warehousing technologies.
- Knowledge on data visualization and present findings in meaningful way.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Introduction of Big Data (4 hrs)

- Big data: definition and taxonomy
- Big data value for the organization
- Setting up the demo environment
- Using Big Data for business point of view
- Role of data Scientist
- Recent Trend in Big Data Analytics

Unit 2: The Hadoop ecosystem (10 hrs)

- Introduction to Hadoop, Hadoop components: MapReduce/Hive/HBase/Yarn/Spark/Storm
- Loading data into Hadoop
- Handling files in Hadoop
- Getting data from Hadoop
- HDFS, Hadoop YARN Architecture
- Exploring Pig and Oozie
- Query Languages for Hadoop
- Hadoop and Amazon Cloud Contents

Unit 3: Querying big data with Hive (10 hrs)

- Introduction to the SQL Language
- From SQL to Hive QL
- Using Hive to query Hadoop files
- Hive for massively parallel ondisk data processing

Unit 4: NoSQL, Searching and Indexing Big Data (8 hrs)

- Structured and Unstructured Data
- Taxonomy and NoSQL Implementation,
- Architecture of Hbase, Cassandra and MongoDb
- Full text Indexing and Searching, Distributed Searching with Elastic search Indexing with Lucene

Unit 5: Big data & Machine learning (8 hrs)

- Big Data & Machine Learning
- Machine learning tools (Spark & SparkML, H2O, Azure ML)
- Big data analysis using graph databases: Analyzing big data with Neo4j using Cypher queries

Unit 6: Big data analysis using graphs (8 hrs)

- Introduction to graphs
- Graph analytics (Path analytics, connectivity analytics, community analytics, centrality analytics)

- Graph analytics with Neo4j
- Graph analytics with GraphX
- Apache Spark's API for graphs and graph parallel computation

5. Laboratory Work **(16 hrs)**

Work in big data technologies using real world problems that will cover all the aspects included in the course. It will assist students to gain practical skill in knowing about problems faced and tackle prosecuting knowledge of tools learned in course.

- HDFS: Setup a HDFS in a single node to multi node cluster, perform basic file system operation on it using commands provided, monitor cluster performance.
- Map-Reduce (MR): Write various MR programs dealing with different aspects of it as studied in course
- Hbase: Setup of Hbase in single node and distributed mode, write program to write into hbase and query it.
- Elastic Search: Setup elastic search in single mode and distributed mode, Define template, Write data in it and finally query it.

6. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

7. Books(4-10 books) :

- Jugnesh Kumar and Anubhav Kumar. Big Data and Analytics: The key concepts and practical applications of big data analytics, 2024.
- EMC Education Services. Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data, 2015.
- Thomas H. Davenport. Big Data at Work: Dispelling the Myths, Uncovering the Opportunities
- David Loshin. Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph.

- Balamarugan Balusamy, Nandhini Abirami R, Seifedine Kadry and Amir Gandomi. Big Data: Concepts, Technology and Architecture. Wiley Publisher.

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Mobile Application Development**

Course Code	MCS 712	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

Android app development offers a wealth of opportunities for building innovative and feature-rich mobile applications. As Android continues to evolve, developers have the opportunity to create apps that cater to a global audience and enhance the mobile experience for users worldwide.

2. Course Objectives

After Successful completion of the above course, students will be able to:

- **Acquire** an insight into concepts of mobile application development terminologies, environment and architecture
- **Design** mobile applications using various UI components and layouts.
- **Develop** software with reasonable complexity on mobile platforms.
- **Deploy** applications on mobile devices incorporating most of the key aspects of the platform.

3. Course Details

3.1 Theory (48 hrs)

Unit 1: Introduction to Android Operating System, Development using Kotlin and UI Designer: (4 hrs)

Overview, Android Architecture, Android Ecosystem, Android Versioning and APIs, Development Environment, Setup and Interface, Android Emulator and ADB (Android Debug Bridge) setup, The Structure of Android's Code, Android Components, Activity Lifecycle and debugging, Migration from JAVA to Kotlin, OOP concepts in Kotlin, Functions and Lambda in Kotlin and Intents.

Unit 2: Android Resource handling, Layouts, Views and Material Design for Multi-Screen (Responsive): (6 hrs)

- **Resources**: String, Color, Drawable, Styles, Theme,
- **Menus**: Option and Context
- **Layouts**: Linear Layout, Frame Layout, Relative Layout, Constraint Layout, Dynamic Implementation of Layout.
- **Views**: Dialog boxes, Navigation Drawer, Snackbar, Toast and basic views

Material Design: UI widgets with properties, events and methods, and wiring up layouts with Kotlin code

- **Multi-Screen**: Orientations, and Design for all Screen Sizes

Unit 3: Advanced Views, Fragments, Adapters and Binding Models: (5 hrs)

- **Views**: GridView, WebView, ScrollView, ListView, RecyclerView, CardView
- **Fragment**: Introduction, life Cycle, Implementation, Adapters and Array-List, Model-View-Controller, Data-binding and View-binding with Kotlin

Unit 4: Data Storage Techniques **(4 hrs)**

Connectivity using Firebase, Authenticating Users, Real-Time Database and CRUD, Cloud Messaging using Firebase, Storage using firebase, SQL-lite Connectivity and CRUD operations, Room persistence library, Data Store preference

Unit 5: Version Control System with GitHub **(3 hrs)**

Introduction to GitHub, Making Repository, Cloning Repository, Push, Pull and Fork

Unit 6: Web Application Integration Techniques **(4 hrs)**

API Communication with Web API, Introduction to JSON Data, JSON Parsing, Working with REST API, Data Classes in Kotlin, Introduction to Glide, Loading images from URL, Google API, Web-view

Unit 7: Overview of Jetpack and Sensor Managers **(3hrs)**

Introduction to Jetpack, CameraX, Regular Sensors and Coroutines

Unit 8: Polish and Publish Application **(4 hrs)**

Different Ways to Monetize, Versioning, Signing AAB, Packaging and Beta Test of Mobile Application, Distributing Application on Mobile Market Place

3.2 Laboratory Work **(16 hrs)**

- Installation of Android studio.
- Development Of Hello World Application
- Create an application that takes the name from a text box and shows hello message along with the name entered in text box, when the user clicks the OK button
- Create a screen that has input boxes for User Name, Password, Address, Gender (radio buttons for male and female), Age (numeric), Date of Birth (Date Picket), State (Spinner) and a Submit button. On clicking the submit button, print all the data below the Submit Button (use any layout)
- Design an android application to create page using Intent and one Button and pass the Values from one Activity to second Activity
- Design an android application Send SMS using Intent
- Create an android application using Fragments
- Design an android application Using Radio buttons
- Design an android application for menu.
- Create a user registration application that stores the user details in a database table.

4. Evaluation Scheme

Evaluation of student's performances is divided into two main parts: (1) Internal Evaluation and (2) Semester examination, with breakdowns as follows:

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10

	Total Internal Marks		60	30
Semester-End Examination			40	20

5. Books(4-10 books):

- John Horton,"Android Programming with Kotlin for Beginners" 2nd Edition, Packt Publication.
- Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, Mike Dunn, G. Blake Meike, Title: "Programming Android with Kotlin", O`Reilly.
- Reto Meier, "Professional Android 4 Application Development", John Wiley & Sons.
- Alex Forrester; Eran Boudjnah; Alexandru Dumbravan; Jomar Tigcal, "How to Build Android Apps with Kotlin", Packt.

Lumbini Technological University
Estd: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Information Security Audit**

Course Code	MCS 713	Year/Semester	I/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course equips students with the knowledge and practical skills to perform effective Information Systems (IS) Audits in modern environments. Covering core concepts, security issues, risk-based auditing, and emerging challenges, it emphasizes hands-on application using current tools and techniques relevant to cloud, hybrid infrastructure, DevOps, data privacy regulations, and e-commerce. Students will learn to assess controls, identify vulnerabilities, analyze risks, and evaluate business continuity in alignment with industry standards and market demands.

2. Course Objectives

The specifics objectives of the course are:

- To explore concepts and evolving frameworks of IS auditing.
- To understand hardware, software, cloud, and network security issues in contemporary environments.
- To plan, conduct, and report on IS audits using practical methodologies and tools.

3. Learning Outcomes

Upon successful completion of the Knowledge Management course, learners will be able to:

- Articulate the principles, theories, and frameworks of IS Audit
- Explain the role of IS Audit in enhancing organizational performance, innovation, and competitive advantage.
- To apply risk-based principles to audit planning, scoping, and execution.
- To design and audit business continuity and disaster recovery plans focused on ransomware and cloud resilience.
- To evaluate security controls and compliance specifically within e-commerce, mobile, API, and cloud ecosystems.
- To understand and perform fundamental security testing techniques (vulnerability scanning, log analysis).
- To analyze the impact of data privacy regulations (GDPR, CCPA) and third-party risk on IS audits.
- Analyze case studies to identify best practices and challenges in IS Audit implementation across industries.
- Propose solutions to real-world IS Audit problems, integrating ethical considerations and stakeholder perspectives.

4. Course Details

4.1 Theory (48 hrs)

Unit 1: Foundations of Modern IS Auditing (5 hrs)

- Information Systems Audit: Evolution & Role
- The IS Auditor: Skills, Knowledge (CISA), & Ethics
- Legal & Regulatory Landscape: SOX, GDPR, CCPA, HIPAA implications
- Systems Environment: On-Prem, Cloud (IaaS/PaaS/SaaS), Hybrid, DevOps
- Information Assets: Data Classification (PII, PHI, IP)
- Classification of Controls: Preventive, Detective, Corrective
- Impact of Cloud, Mobile, IoT on Information & Auditing
- IS Audit Coverage & Scope

Unit 2: Infrastructure & Hardware Security (4 hrs)

- Hardware Security Objectives in Cloud Era
- Endpoint Security: Laptops, Mobile, IoT, and Storage Media Encryption
- Cloud Infrastructure Security Considerations: Hypervisors, SDN
- Authentication Devices: MFA, Biometrics, Hardware Tokens
- Acquisition: Cloud Service Provider (CSP) Assessments, SLAs
- Maintenance & Obsolescence: Patch Management Verification
- Disposal: Data Sanitization Standards (NIST SP 800-88)
- Problem & Change Management: Auditing CMDBs & Processes

Unit 3: Software & Development Security (5 hrs)

- Software Types: OSS Licensing & Compliance Risks, Container Security, SaaS
- Elements of Software Security: SDLC Security (Secure Coding), DevSecOps
- Control Issues: CI/CD Pipeline Security, Container Image Scanning, Infrastructure as Code (IaC) Security
- Open Source License Management & Audit
- API Security Fundamentals
- Problem & Change Management: Auditing in Agile/DevOps

Unit 4: Risk-Based IS Audit Fundamentals (6 hrs)

- Audit Requirements & Planning
- Modern Risk Analysis Frameworks: NIST RMF, ISO 27005
- Threats, Vulnerabilities, Likelihood, Impact, Risk Appetite
- IS Control Objectives: COBIT 2019, NIST CSF mapping
- IS Audit Objectives: Effectiveness, Efficiency, Compliance (Privacy!)
- Information Systems Abuse & Fraud Detection
- Asset Safeguarding: Data Lifecycle Controls
- Evidence Collection: Electronic Evidence Handling, Chain of Custody
- Logs & Audit Trails: SIEM Concepts, Centralized Log Management

Unit 5: Conducting the IS Audit (7 hrs)

- Audit Program & Plan: Agile Audit Techniques
- Procedures & Approaches: Data Analytics (ACL, IDEA basics), Continuous Auditing/Monitoring
- System Understanding: Documenting Cloud Architectures, Microservices

- Compliance & Substantive Testing: Automated Testing Scripts
- Auditing Tools: Vulnerability Scanners (Nessus, OpenVAS), Configuration Compliance Scanners
- Sampling Techniques: Data-Driven Auditing
- Audit Documentation: Electronic Workpapers
- Audit Report: Clear Communication of Technical Findings

Unit 6: Advanced Risk-Based Audit **(4 hrs)**

- Integrating Risk into Audit Lifecycle
- Risk Assessment: Quantitative Methods Intro, Risk Heat Maps
- Risk Matrix & Prioritization
- Risk & Audit Sample Determination: Focusing on High-Risk Areas
- Audit Risk Assessment Model: Inherent, Control, Detection Risk
- Third-Party Risk Management (TPRM) Audit Considerations

Unit 7: Business Continuity & Disaster Recovery **(8 hrs)**

- BC/DR Process: Ransomware Focus, Cloud DR Strategies (AWS DR, Azure Site Recovery)
- Business Impact Analysis (BIA): RTO/RPO for Critical Cloud Services
- Cyber Incident Response Plan (CIRP) Integration
- DR Plans: Cloud-Based Failover, Testing in Cloud Environments
- Auditing Resilience against Ransomware: Multi-Cloud, Immutable Backups
- Plan Testing: Tabletop Exercises, Simulation Drills
- Checklists & Maintenance: Automating DR Validation Checks

Unit 8: Auditing Digital & E-Commerce **(5 hrs)**

- Expanded Scope: Mobile Commerce (m-Commerce), API Ecosystems, Digital Payment Gateways
- Audit Objectives: API Security, Mobile App Security, Fraud Prevention
- General Overview: OWASP Top 10 for Web, API, Mobile
- Auditing Functions: Shopping Cart, Payment Processing (PCI DSS Intro), User Authentication, Inventory Management (Cloud)
- Policies & Procedures: Acceptable Use, Data Retention, Incident Response for Digital
- Internal Control Impact: Segregation in Microservices, Tokenization, Fraud Analytics

Unit 9: Security Testing & Cyber Threats **(6 hrs)**

- Cybersecurity Landscape: Ransomware, Supply Chain Attacks, Phishing
- Cybercrimes: Business Email Compromise (BEC), Cryptojacking
- Attack Surfaces: Cloud Misconfigurations, APIs, SaaS Settings, Employees
- Attack Vectors: Exploit Kits, Credential Stuffing, Zero-Days
- Vulnerability Assessment vs. Penetration Testing
- Vulnerability Management Process Lifecycle
- Introduction to Penetration Testing (Ethical Hacking) Concepts
- Cyber Forensics: Digital Evidence Acquisition Basics, Volatility
- Auditor's Role in Security Testing Review & Oversight

4.2 Laboratory Work **(16 hrs)**

- Practical 1: Case Study - Identifying applicable regulations for a given scenario; exploring CISA domains.
- Practical 2: Lab - Reviewing cloud provider (AWS/Azure/GCP) security documentation & SLAs; Simulating hardware disposal audit steps.
- Practical 3: Lab - Using SAST/DAST tools (e.g., OWASP ZAP, Bandit) on sample code; Analysing OSS licenses for compliance; Reviewing IaC templates for misconfigurations.
- Practical 4: Workshop - Performing a qualitative risk assessment on a cloud migration project; Defining audit objectives based on NIST CSF controls.
- Practical 5: Simulation - Creating an audit program for a SaaS application; Using a vulnerability scanner (e.g., OpenVAS) against a test environment; Documenting findings in a sample audit report section.
- Practical 6: Exercise - Building a risk matrix for an e-commerce platform; Prioritizing audit tests based on risk assessment output.
- Practical 7: Workshop - Reviewing a sample cloud DR plan; Designing a tabletop exercise scenario for a ransomware attack; Auditing backup encryption and immutability settings.
- Practical 8: Lab - Analysing e-commerce site/auth flows for OWASP Top 10 vulnerabilities (using proxies like Burp Suite Community); Reviewing PCI DSS SAQ requirements.
- Practical 9 : Lab - Interpreting vulnerability scan reports (critical/high findings); Basic log analysis with Splunk/ELK demo; Hands-on with OSINT tools (e.g., the Harvester); Overview of a pen test report structure.

5. Evaluation Scheme

Evaluation of student's performances is divided into two main parts: (1) Internal Evaluation and (2) Semester examination, with breakdowns as follows:

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

6. Books (4-10 books):

- Books: "Knowledge Management in Theory and Practice" by Kimiz Dalkir, "The Knowledge-Creating Company" by Ikujiro Nonaka.
- Tools: Confluence, SharePoint, Microsoft Teams, Trello.
- Articles: Harvard Business Review on KM, case studies from KMWorld.

Lumbini Technological University
Est: 2079 BS (2022 AD)
Banke, Nepalgunj

Level: Master of Technology **Subject: Spatial Database Management System**

Course Code	MCS 714	Year/Semester	II/I
Credit Weightage	3	Lecture	3 hrs/wk
		Tutorial	N/A
		Practical	1 hrs/wk
		Total	64 hrs

1. Course Description

This course covers the design, implementation, and optimization of spatial databases for managing geospatial data. Students will learn spatial data models (relational, object-oriented), SQL extensions for spatial queries (OGIS/PostGIS), indexing methods (R-trees, Quadtrees), and computational geometry. Hands-on labs using PostgreSQL/PostGIS will emphasize real-world applications in urban planning, logistics, and environmental monitoring.

2. Course Objectives

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

- Understand the principles of SDBMS and their advantages over traditional DBMS.
- Design spatial data models (ER/UML) and implement them in PostgreSQL/PostGIS.
- Optimize spatial queries (joins, buffers) and evaluate indexing strategies.
- Apply SDBMS techniques to solve problems in disaster response, IoT, and smart cities.

3. Learning Outcomes (Follow Bloom's Taxonomy to the extent possible. See Annex A)

Upon completing this course, Students will be able to:

- Recall core ACID properties of DBMS and spatial data types (points, polygons, rasters).
- List spatial indexing methods (R-trees, Quadtrees) and their use cases.
- Explain the differences between field-based vs. object-based spatial models.
- Summarize OGIS SQL standards for spatial operations (Intersects, union etc).
- Implement spatial ER diagrams with UML extensions.
- Write SQL queries with PostGIS functions (Distance, Buffer etc)
- Diagnose query performance bottlenecks using execution plans.
- Compare trade-offs of spatial indexing (R-tree vs. Grid indexing)
- Justify the choice of data model (relational vs. NoSQL) for a given geospatial application.
- Assess normalization trade-offs for spatial databases.
- Design a web-based spatial application (e.g., real-time traffic analysis) using PostGIS.
- Develop a spatial data pipeline for IoT sensor networks.

4. Lecture Course Details

4.1 Theory (48hrs)

Unit 1: Introduction to SDBMS (8hrs)

- Definition, evolution, and importance of SDBMS.
- Comparison with traditional DBMS : ACID (Atomicity, Consistency, Isolation, and Durability) properties, spatial extensions.
- GIS, location-based services (LBS), IoT, urban planning.
- Handling large-scale geospatial data, topology, projections.

Unit 2: Spatial Data Models & Standards (10 hrs)

- Field-based: Raster grids, TINs (Triangulated Irregular Networks).
- Object-based: Points, lines, polygons, trajectories.
- Hybrid models: Network models (e.g., road networks).
- Standards: OGC Simple Features: WKT, WKB, GeoJSON; ISO 19107: Spatial schema standards.
- Shapefiles data formats: Shapefiles, GeoPackage, GeoParquet.
- Open Data Initiatives: OpenStreetMap, NASA Earthdata.

Unit 3: Spatial SQL & Query Processing (8 hrs)

- SQL Extensions: OGIS SQL: Intersects, Within, ConvexHull etc; Window functions for spatial analytics.
- Query Optimizations: Spatial joins (hash joins, nested loops); Cost-based optimization for spatial queries.
- Procedural Extensions: PL/pgSQL for spatial workflows.
- Temporal-Spatial Queries : Time-enabled spatial data (e.g., tracking moving objects).

Unit 4: Spatial Indexing & Storage (8 hrs)

- Indexing Methods: R-trees, Quadtrees, KD-trees, Grid files; Space-filling curves: Z-order, Hilbert curves.
- Storage Optimization: Clustering strategies (geohashing, tile-based storage); Columnar storage for rasters (GeoParquet).
- Distributed Spatial Indexing: Apache Sedona, GeoMesa for big spatial data.
- Benchmarking : Performance comparison of indexing methods.

Unit 5: Computational Geometry for SDBMS (8 hrs)

- Core Algorithm : Convex hull, Voronoi diagrams, Delaunay triangulation ; Shortest path for networks analysis.
- Spatial Operations : Buffering, overlay, simplification (Douglas-Peucker).
- Topological Relationship : DE-9IM model (contains, covers, overlaps).
- Machine Learning Integration: Spatial clustering (DBSCAN), pattern detection.

Unit 6: Advanced SDBMS & Ethics (8 hrs)

- NoSQL Spatial Databases: MongoDB Geospatial, Neo4j Spatial, Elasticsearch Geo
- Cloud Natives SDBMS: Google BigQuery GIS, AWS Aurora PostGIS, Azure SQL Spatial.
- Big Data & Spatial Analytics: Apache Sedona, GeoSpark for distributed processing.
- Ethics & Privacy: GDPR (General Data Protection Regulation) for geospatial data, anonymization techniques.

4.2 Practical (16 hrs)

Lab	Practical Title	Hours
1	Setting Up a Spatial Database (PostGIS)	2 hrs
2	Exploring Raster and Vector Models	2 hrs
3	Writing Spatial SQL Queries	2 hrs
4	Spatial Joins and Query Optimization	2 hrs
5	Spatial Indexing Techniques	2 hrs
6	Distributed Spatial Processing with Apache Sedona	2 hrs
7	Shortest Path and Network Analysis	2 hrs
8	Ethics and Anonymization in Geospatial Data	2 hrs

For each lab work, students are supposed to implement the algorithm with performance analysis.

5. Evaluation Scheme

Evaluation of students' performance is divided into two main parts: (1) Internal assessment and (2) Semester examination with breakdown as follows.

Evaluation	Components	Weightage	Full Marks	Pass Marks
Internal Evaluation	Class Attendance and Performance	5	25	12.5
	Assignment	5		
	Seminar/Project/Presentation	15		
	Practical/Lab Examination	15	15	7.5
	Internal Examination	20	20	10
	Total Internal Marks		60	30
Semester-End Examination			40	20

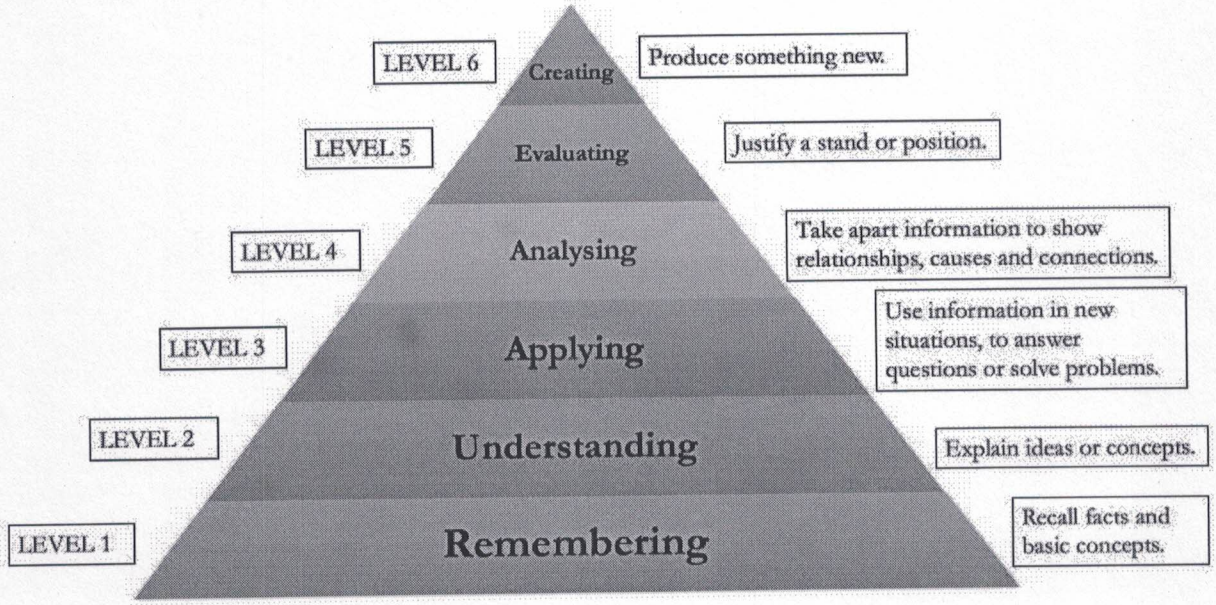
6. Books (4-10 books):

1. Shashi Shekhar, Sanjay Chawla (2003), Spatial Database a tour.
2. Philippe Rigaux. Michel Scholl, Anges Voisard (2002), Spatial Databases with Application to GIS. Morgan Kaufmann Publishers
3. Raghu Ramakrishna, Johannes Gehrke (2007), Database Management System/McGraw-Hill Education.
4. Abraham Siberschatz, Henry Korth, S. Sudarshan (2010), Database system concepts, McGraw-Hill Education.
5. Database system concepts by Silberschatz, Korth and Sudarshan
6. Database management system by Ram Krishna Gehrke

Annex A: Bloom's Taxonomy action verbs

As Bloom's taxonomy is a hierarchy of progressive processes ranging from the simple to the complex, in which it is necessary to first master those lower down the pyramid before being able to master those higher up, the framework promotes what Bloom termed 'mastery learning'. In other words, by moving up the taxonomy, students become more knowledgeable, more skilled and develop an improved understanding of the content they are learning. Thus, by creating lesson plans and tasks,

using the examples of verbs (in italics) provided, teachers can align with the different levels of the taxonomy.



Level 1, Remembering, is the most basic, requiring the least amount of cognitive rigour. This is about students recalling key information, for example, the meaning of a word.

Arrange Define Describe List Match Name Order Recall Reproduce

Level 2, Understanding, is to do with students demonstrating an understanding of the facts remembered. At this level, the student who recalls the definition of a word, for example, would also be able to show understanding of the word by using it in the context of different sentences.

Classify Discuss Explain Identify Report Summarise

Level 3, Applying, is concerned with how students can take their knowledge and understanding, applying it to different situations. This usually involves students answering questions or solving problems.

Apply Calculate Demonstrate Interpret Show Solve Suggest

Level 4, Analysing, is about students being able to draw connections between ideas, thinking critically, to break down information into the sum of its parts.

Analyse Appraise Compare Contrast Distinguish Explore Infer Investigate

Level 5, Evaluating, is reached when students can make accurate assessments or judgements about different concepts. Students can make inferences, find effective solutions to problems and justify conclusions, while drawing on their knowledge and understanding.

Argue *Assess* *Critique* *Defend* *Evaluate* *Judge* *Justify*

Level 6, Creating, is the ultimate aim of students' learning journey. At this final level of Bloom's taxonomy, students demonstrate what they have learnt by creating something new, either tangible or conceptual. This might include, for example, writing a report, creating a computer program, or revising a process to improve its results.

Compose *Construct* *Create* *Devise* *Generate* *Organise* *Plan* *Produce*