

PG Department of Computer Science(SSC)

Course Structure (w.e.f.2021)

SEMESTER – I

Subject	Course Code	Course Title	Contact Hours / Week	Credits	Max. Marks		
					CIA	ESE	Total
Core I	21PCSC11	Design and Analysis of Algorithms	5	4	40	60	100
Core II	21PCSC12	Digital Image processing using MATLAB	5	4	40	60	100
Core III	21PCSC13	Mathematical Foundations for Computer Science	4	4	40	60	100
Core IV	21PCSC14	Compiler Design	4	4	40	60	100
Elective I	21PCSE11/ 21PCSE12	A. Advanced computer Architecture/ B. Cryptography & Network Security	4	4	40	60	100
Core Practical I	21PCSCR1	Design and analysis of algorithm Lab	4	2	40	60	100
Core Practical II	21PCSCR2	Digital Image processing using MATLAB Lab	4	2	40	60	100
			30	24			700

SEMESTER – II

Subject	Course Code	Course Title	Contact Hours / Week	Credits	Max. Marks		
					CIA	ESE	Total
Core V	21PCSC21	J2EE	5	4	40	60	100
Core VI	21PCSC22	Data Mining & R Programming	5	4	40	60	100
Core VII	21PCSC23	DDBMS	4	4	40	60	100
Core VIII	21PCSC24	Single Board Computers and IoT	4	4	40	60	100
Elective II	21PCSE21/ 21PCSE22	A. Advanced Computer Networks / B. Soft Computing	4	4	40	60	100
Core Practical III	21PCSCR3	J2EE Lab	4	2	40	60	100
Core Practical IV	21PCSCR4	Data Mining Lab (R Lab)	4	2	40	60	100
			30	24+2			800

SEMESTER –III

Subject	Course Code	Course Title	Contact Hours / Week	Credits	Max. Marks		
					CIA	ESE	Total
Core IX	21PCSC31	Software Testing	4	4	40	60	100
Core X	21PCSC32	Cloud Computing & Big Data	4	4	40	60	100
Core XI	21PCSC33	Data Science using Python	4	4	40	60	100
Core XII	21PCSC34	Research Methodology	4	4	40	60	100
Elective III	21PCSE31/ 21PCSE32	A. Organizational Behaviour / Object Oriented Software Engineering	4	4	40	60	100
Core Practical V	21PCSCR5	Python Lab	4	2	40	60	100
Project	21PCSCR6	Mini Project	6	4	40	60	100
Self-study Course/ MOOC/ Internship	21PCSSS1/ 21PCSM31/ 21PCSI31	Course for Competitive Exams		+2		100	100
			30	26+2			800

SEMESTER – IV

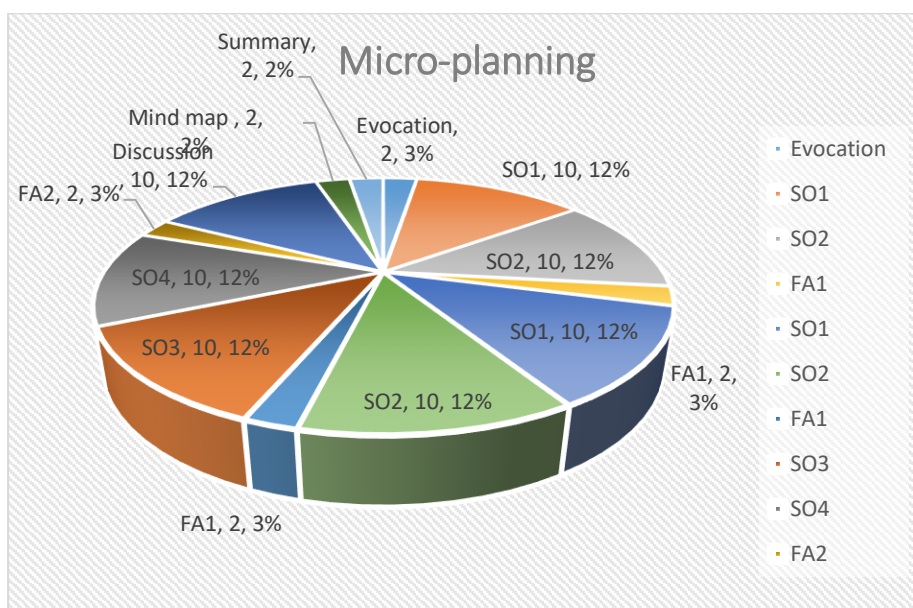
Subject	Course Code	Course Title	Contact Hours / Week	Credits	Max. Marks		
					CIA	ESE	Total
Project	21PCSP41	Project work	30	16	100	100	200
			30	16			200

Lesson Plan

Programme	M.Sc. Computer Science
Semester	I
Course Title	Design and Analysis of Algorithms
Code	21PCSC11
Hours	5
Total Hours	75
Credits	4
Max Marks	60
Unit & Title	Unit III & Depth First Search (DFS) and Breadth First Search (BFS)
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basic understanding of graphs, vertices, and edges

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

A brief explanation of graph traversal methods in real-life scenarios (e.g., navigating through social networks, exploring cities, etc.). Ask students to share where they think DFS or BFS might be applicable.

2. Topic Introduction:

2.1: General Objective:

- To understand graph traversal techniques, DFS and BFS, and their applications.
- To analyze their time and space complexities.

2.2: Specific Outcomes:

- SO1: Understand the working of Depth First Search (DFS).
- SO2: Understand the working of Breadth First Search (BFS).

- SO3: Compare the techniques of DFS and BFS.
- SO4: Explore the applications of both techniques.

First Phase:

SO1 (10 minutes):

- Explain Depth First Search (DFS), its stack-based implementation, and recursive nature.
- Highlight the order in which nodes are visited.

SO2 (10 minutes):

- Explain Breadth First Search (BFS), its queue-based implementation, and level-wise node exploration.
- Compare its nature to DFS in traversal.

Second Phase:

SO3 (10 minutes):

- Compare DFS and BFS side-by-side based on:
 - Space complexity
 - Time complexity
 - Applications (e.g., cycle detection, shortest path, etc.).

SO4 (10 minutes):

- Demonstrate real-world applications for both algorithms (e.g., maze solving with DFS, shortest path in unweighted graphs with BFS).

Mind Map (2 minutes):

Create a simple mind map comparing DFS and BFS, including:

- Key characteristics.
- Data structures used.
- Examples of use cases.

Summary (2 minutes):

Summarize the distinctions, advantages, and limitations of DFS and BFS. Highlight practical programs for both algorithms.

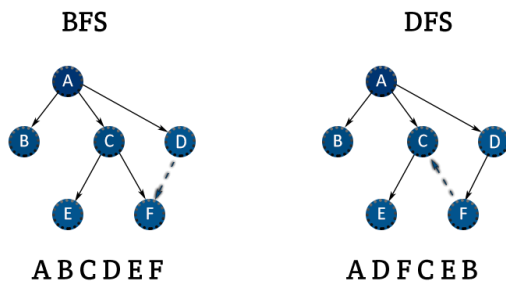
2.3: Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1	1				
B. Conceptual Knowledge			2	2,3		
C. Procedural Knowledge		1				2,3
D. Meta Cognitive Knowledge					2,3	

2.4: Key Words:

DFS, BFS, Graph Traversal, Recursion, Queue, Stack, Complexity

2.5: Key Diagrams:



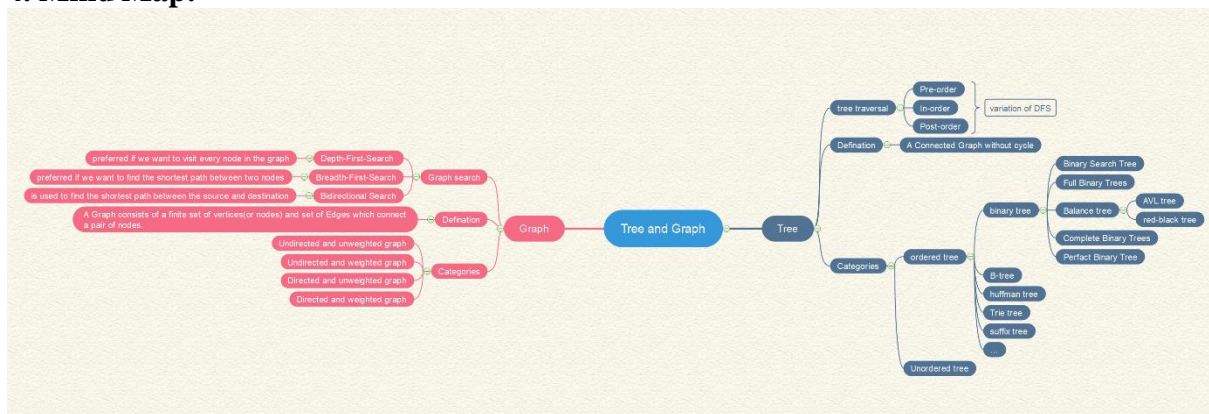
PPT Link: <https://gamma.app/docs/Graph-Traversal-DFS-and-BFS-Algorithms-2f5jpachpt3xqc3>

3. Discussion:

Students are asked to:

- Discuss differences between DFS and BFS with examples.
- Frame scenarios where DFS would fail to deliver optimal results but BFS would succeed, and vice versa.

4. Mind Map:



5. Summary:

DFS explores as far as possible along a branch, using stacks or recursion, suitable for pathfinding and cycle detection. BFS explores level by level, using queues, making it apt for shortest path algorithms. Understanding these techniques equips students to solve various graph-related problems.

6. Assessment:

Formative Assessment 1 (FA1):

- Students explain, step-by-step, how DFS would traverse a given graph.

Formative Assessment 2 (FA2):

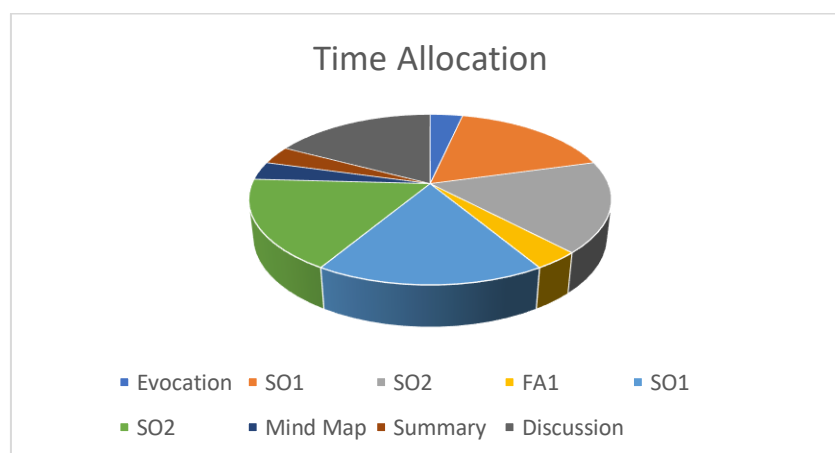


Lesson Plan

Programme	M.Sc Computer Science
Semester	I
Course Title	Digital Image Processing using MATLAB
Code	21PCSC12
Hours per Week	5
Total Hours	75
Credits	4
Maximum Marks	60
Unit & Title	Unit II: Basic Gray Level Transformation
Name of Faculty	A.Jenitta Jebamalar
T-L Tools	PowerPoint, Visual Demonstrations, Group Discussions, Interactive Coding Sessions

Pre-requisite Knowledge: Basics of image representation (pixels, grayscale images) and understanding of intensity values.

Micro-Planning (60 minutes)



- 1. Evocation (2 minutes):**
Discuss the real-life application of image enhancement and transformations, such as improving image clarity in medical imaging or photography.
- 2. Topic Introduction:**

2.1 General Objective:

- To understand the concept of basic gray-level transformations.
- To apply these transformations for image enhancement.

2.2 Specific Outcomes (SOs):

- Explain the concept of gray-level transformations
- Describe the mathematical formulations and examples for these transformations.

First Phase

SO1 (10 minutes):

- Define gray-level transformations and their importance in image processing.

- Demonstrate basic operations like:
 - Contrast Stretching: Expanding the range of intensity values.
 - Inversion: Transforming each pixel intensity $I(x,y)$ to $L-1-I(x,y)$, where L is the max intensity.

SO2 (10 minutes):

- Illustrate examples of practical applications:
 - Highlighting details in medical X-rays.
 - Adjusting brightness and contrast in photographs.
- Discuss the equations and transformations visually using graphs and PowerPoint.

Formative Assessment 1 (FA1 - 2 minutes):

Students are asked to perform inversion transformation on a sample grayscale image (in-class coding or discussion).

Second Phase

SO1 (10 minutes):

- Explain other gray-level transformation techniques:
 - Log Transformation: Emphasizing darker pixel values.
 - Power-law (Gamma) Transformation: Enhancing specific intensity ranges.

SO2 (10 minutes):

- Demonstrate these transformations using simple coding examples (MATLAB, Python, etc.).
- Discuss scenarios where each transformation is best applied.

Mind Map (2 minutes):

- Create a visual mind map comparing the key transformation techniques and their use cases.

Summary (2 minutes):

Summarize the advantages and limitations of basic gray-level transformations in improving image quality. Discuss their relevance in different domains (medical, surveillance, photography).

2.3 Taxonomy of Objectives

Taxonomy of Objectives						
Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1	1				
B. Conceptual Knowledge		1		1		
C. Procedural Knowledge			1		1	
D. Meta Cognitive Knowledge					1	1

2.4 Key Words:

Gray-level transformations, Contrast stretching, Inversion, Log transformation, Power-law transformation

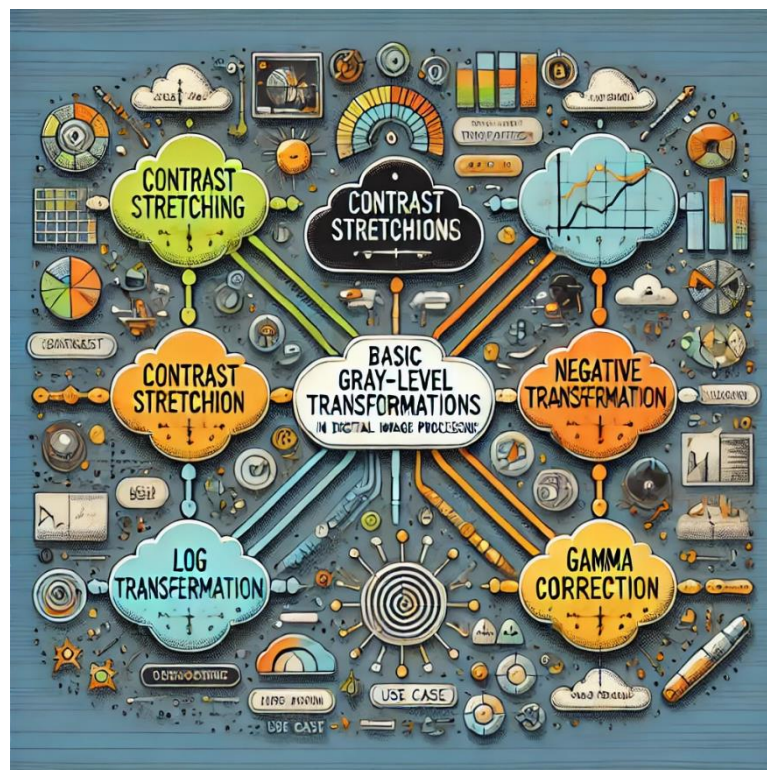
2.5 Powerpoint Presentation:

<https://gamma.app/docs/7a6ic1z3wc2g2tx>

3. Discussion (10 minutes):

Students discuss the advantages and trade-offs of applying different transformations to the same image. Groups present their observations.

4. Mind Map



5. Summary:

As Gray-level transformations modify pixel intensities to enhance image contrast, brightness, and visibility.

Types of Transformations:

1. Linear Transformations:
 - Contrast Stretching: Expands intensity range for better contrast.
 - Negative Transformation: Inverts pixel values, highlighting darker areas.
2. Logarithmic Transformations:
 - Log Transform: Enhances dark areas while compressing bright regions.
 - Inverse Log: Expands high-intensity values for better visibility.
3. Power-Law (Gamma) Transformation:
 - Adjusts brightness and contrast using a power function.
 - Used in gamma correction for display devices.

Applications:

- Enhancing medical images (e.g., X-rays).
- Improving low-light images (e.g., surveillance).
- Enhancing satellite images.

6. Assessment

- **Formative Assessment 2 (FA2 - 2 minutes):**
A quick review of the equations and concepts discussed.

7.FAQ:

1. What are gray-level transformations, and why are they important?
2. Compare contrast stretching and log transformation.
3. How would you apply gamma correction to improve image visibility in dim conditions?

8. References

1. Rajkumar Bansal, Ashok Kumar Goel and Manoj Kumar Sharma. *MATLAB and its Applications in Engineering*. Pearsons Publications, 2016.
2. Rafael C. Gonzalez. *Digital Image Processing using MATLAB*. 2nd Edition, 2010.

9. Verified by Subject Expert:

A. Leelitha Subramalar

Course In-Charge

R. Jayan

Approved by HoD



Lesson Plan

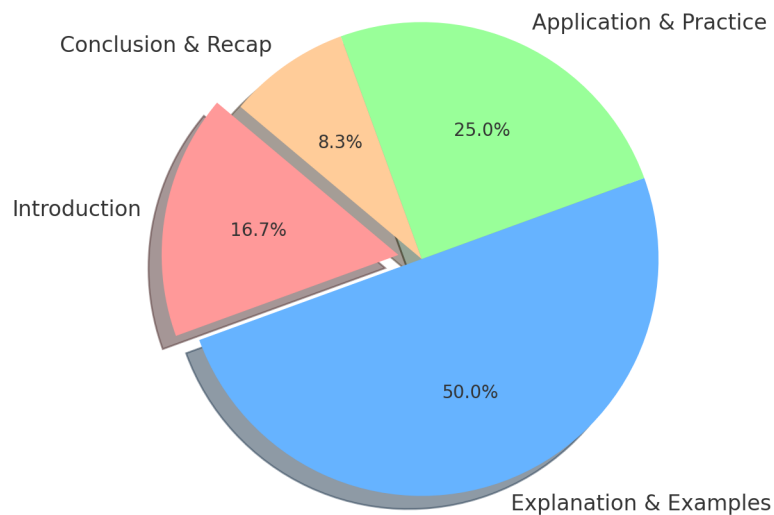
Programme	M.Sc
Semester	I
Course Title	Mathematical Foundation for Computer Science
Code	21PCSC13
Hours	4
Total Hours	60
Credits	4
Max Marks	100
Unit & Title	I / Moments, Skewness & Kurtosis
Name of the Faculty	C.Nayanthra Mascarenhas
T-L tools	PowerPoint

Lesson Objectives:

By the end of the lesson, students should be able to:

1. Understand the concept of moments and their significance in statistics.
2. Define and calculate skewness to determine the asymmetry of a distribution.
3. Define and calculate kurtosis to analyze the peakedness of a distribution.
4. Interpret statistical distributions using moments, skewness, and kurtosis.

Microteaching Time Allocation for Moments, Skewness & Kurtosis in Data Analytics



Lesson Structure:

1. Introduction (10 minutes)

Engagement Activity:

- Show two different histograms: one symmetric and one skewed. Ask students:
 - What differences do you observe in these graphs?
 - How do these shapes relate to data distribution?
- Introduce **Moments, Skewness, and Kurtosis** as statistical measures to describe data distributions.

Key Terms:

- **Moments:** Measures that describe the shape of a distribution.
- **Skewness:** Measure of asymmetry of a distribution.
- **Kurtosis:** Measure of the "tailedness" of a distribution.

2. Explanation & Examples (30 minutes)

SO1: Moments (10 minutes)

- Moments helps to summarize key distribution properties.
 - The **nth moment** about the mean is given by:

$$\mu_n = \frac{1}{N} \sum (X_i - \bar{X})^n$$

- Explain **first four moments**:
 - **1st Moment (Mean):** Central tendency
 - **2nd Moment (Variance):** Spread of data
 - **3rd Moment (Skewness):** Asymmetry
 - **4th Moment (Kurtosis):** Peakedness

Example Calculation:

Given data: $X = \{2, 4, 6, 8, 10\}$


1. Compute mean: $\bar{X} = 6$
2. Compute variance: $\mu_2 = \frac{1}{N} \sum (X_i - \bar{X})^2 = 8$

SO2: Skewness (10 minutes)

- **Formula for Skewness (Pearson's moment coefficient):**

$$Skewness = \frac{\sum (X_i - \bar{X})^3}{N \cdot \sigma^3}$$

- **Types of Skewness:**
 - **Positive Skewness (>0):** Right-tailed distribution
 - **Negative Skewness (<0):** Left-tailed distribution
 - **Zero Skewness (=0):** Symmetric


 **Example:** If skewness = 1.5, the distribution is **positively skewed** (right-skewed).

SO3:Kurtosis (10 minutes)

- **Formula for Kurtosis:**

$$Kurtosis = \frac{\sum (X_i - \bar{X})^4}{N \cdot \sigma^4}$$

- **Types of Kurtosis:**
 - **Mesokurtic (=3):** Normal distribution
 - **Leptokurtic (>3):** Sharp peak (heavy tails)
 - **Platykurtic (<3):** Flatter peak (light tails)

 **Example:** If kurtosis = 4.5, the distribution is **leptokurtic** (heavier tails).


3. Application & Practice (15 minutes)

Microteaching Chart Activity:

- Provide a dataset and ask students to compute moments, skewness, and kurtosis.
- Create a **summary chart** comparing different distributions based on shape, skewness, and kurtosis.

Real-Life Applications

 **Finance:** Skewness helps analyze stock returns.

 **Economics:** Kurtosis helps assess market risks.

 **Data Science:** Moments describe data distribution shapes.

4. Conclusion & Recap (5 minutes)

- Moments describe data shape.
- Skewness detects bias in distribution.
- Kurtosis highlights risk and anomalies.
- Real-world application in finance, marketing, and fraud detection.

Assessment & Homework:

Short Quiz:

1. Define skewness and kurtosis.
2. What does a negative skewness value indicate?

3. If a distribution has kurtosis = 2, what does it imply?
4. Why is it important to check moments in data preprocessing?

Homework:

- Find a dataset online and compute its skewness and kurtosis. Write a short analysis.

Why is skewness important in predicting customer behaviour?

How does kurtosis help detect anomalies in fraud detection?

Microteaching Chart: Moments, Skewness, and Kurtosis

Concept	Definition	Formula	Interpretation	Example
First Moment (Mean, μ)	Average value of the dataset.	$\mu = \frac{\sum X_i}{N}$	Represents the center of the distribution.	If $X = \{2, 4, 6, 8, 10\}$, then $\mu = 6$.
Second Moment (Variance, σ^2)	Spread of data around the mean.	$\sigma^2 = \frac{\sum (X_i - \mu)^2}{N}$	Higher variance means more spread.	$\sigma^2 = 8$ for the dataset $\{2, 4, 6, 8, 10\}$.
Third Moment (Skewness, S)	Measure of asymmetry in the data.	$S = \frac{\sum (X_i - \mu)^3}{N \cdot \sigma^3}$	$S > 0$ = Right skewed, $S < 0$ = Left skewed, $S = 0$ = Symmetric.	If $S = 1.2$, the data is right-skewed.
Fourth Moment (Kurtosis, K)	Measure of the peak and tail heaviness.	$K = \frac{\sum (X_i - \mu)^4}{N \cdot \sigma^4}$	$K > 3$ = Leptokurtic, $K < 3$ = Platykurtic, $K = 3$ = Normal.	If $K = 4.5$, the data has heavy tails (Leptokurtic).

Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge		1,2,3				
B. Conceptual Knowledge				1,2,3		
C. Procedural Knowledge						1,2,3

Reference:

1. **Mathematical Statistics** – Jun Shao.
2. **Mathematical Statistics and Data Analysis**– John A. Rice

PPT Link :

<https://docs.google.com/presentation/d/1l2Vi6Jtq1K7ePBiNf5SvGTQrtRs567vy/edit?usp=sharing&ouid=113456501758187936448&rtpof=true&sd=true>

Verified by Subject Expert:

Rajan

Course In-charge

Rajan

Approved by HoD

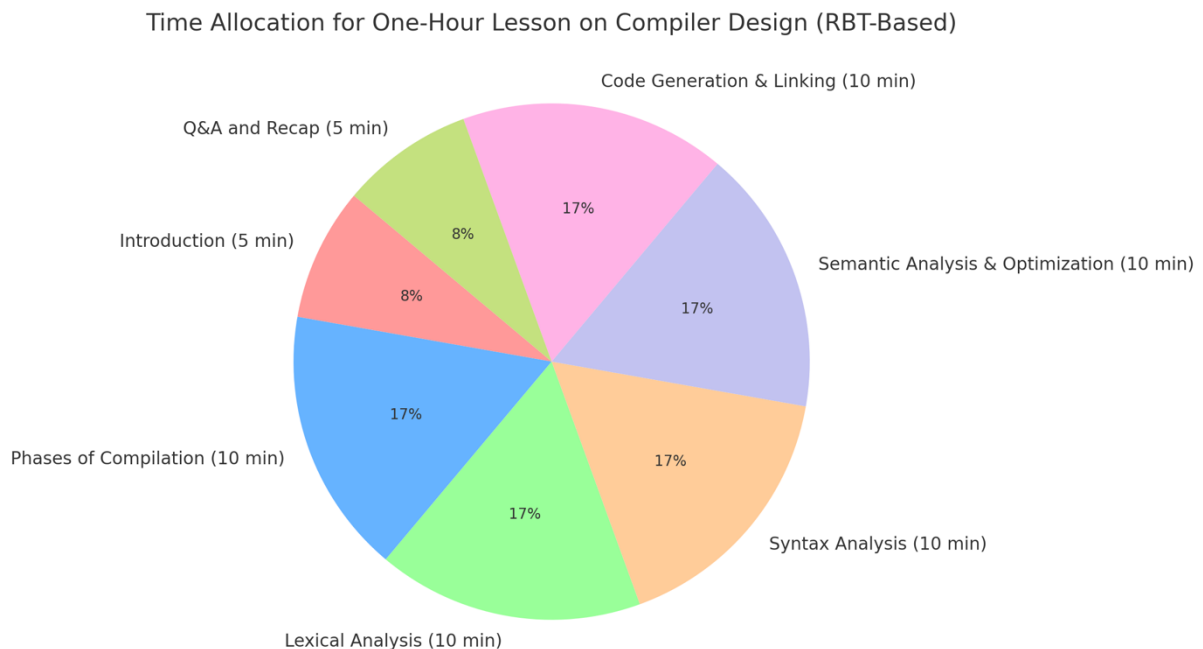


Programme	M.Sc
Semester	I
Course Title	Compiler Design
Code	21PCSC14
Hours	4
Total Hours	60
Credits	4
Max Marks	100
Unit & Title	I / Lexical Analysis
Name of the Faculty	C. Nayanthra Mascarenhas
T-L tools	PowerPoint , Mind Maps, Visual Diagrams (Phases of Compiler)

Pre-requisite Knowledge:

Basic understanding of programming languages, execution process, and data flow in a computer.

Micro Planning: 60 minutes



1. Evocation (2 min)

- Ask students: *"Have you ever wondered how your C/Python/Java programs are converted into machine code?"*
- Introduce real-world compilers (GCC, LLVM, Java Compiler).

2. SO1: Phases of Compilation: (10 minutes)

Explain the major phases of a compiler:

- Lexical Analysis
- Syntax Analysis
- Semantic Analysis
- Intermediate Code Generation
- Optimization
- Code Generation.

3. SO2: Lexical Analysis (Tokenization): (10 minutes)

- Explain the process of breaking down the source code into a stream of tokens (e.g., identifiers, keywords, operators).
- Briefly discuss the role of a lexical analyzer (scanner).

Syntax Analysis (Parsing): (10 minutes)

- Explain how the compiler checks if the sequence of tokens conforms to the grammar of the programming language.
- Introduce different parsing techniques (e.g., top-down, bottom-up).

FA1 – Quick Assessment (Q&A)

- **Ask students:**
 - What is the difference between tokens and lexemes?
 - Mention the types of Parsing?

4. SO1: Semantic Analysis & Code Optimization : (10 minutes)

- Explain the process of checking for semantic errors (e.g., type mismatches, undeclared variables).
- Discuss the generation of intermediate code.
- Explain the process of improving the efficiency of the generated code (e.g., reducing code size, improving execution speed).

5. SO2: Intermediate Code Generation & Code Generation: (10 minutes)

- Explain the generation of an intermediate representation of the source code (e.g., three-address code).
- Explain the translation of the intermediate code into machine code specific to the target architecture.

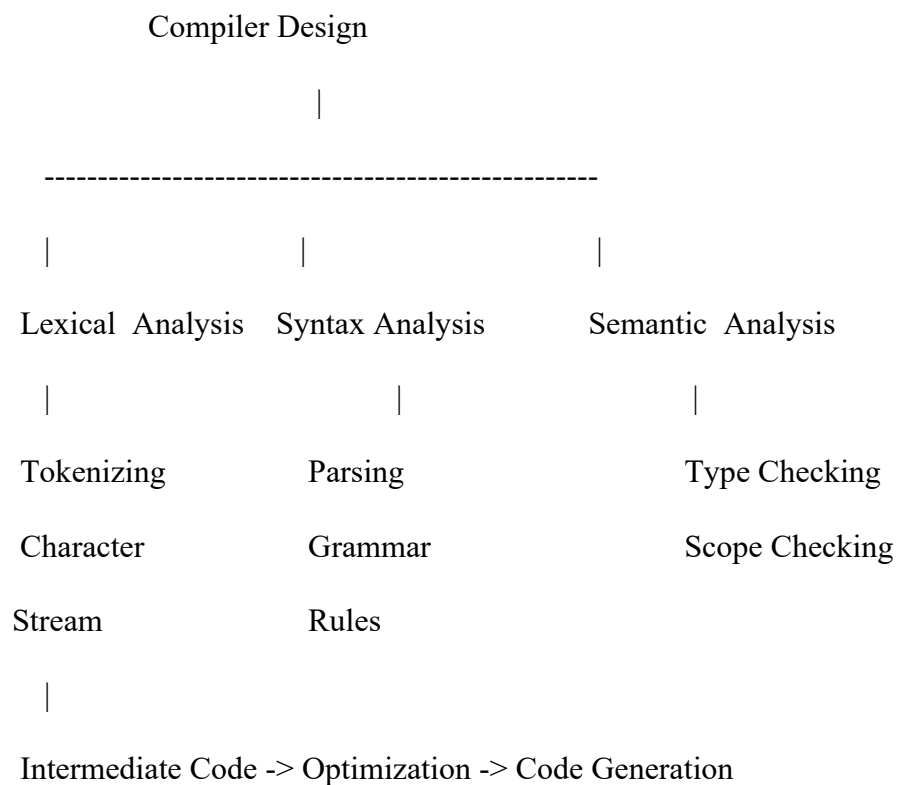
FA2

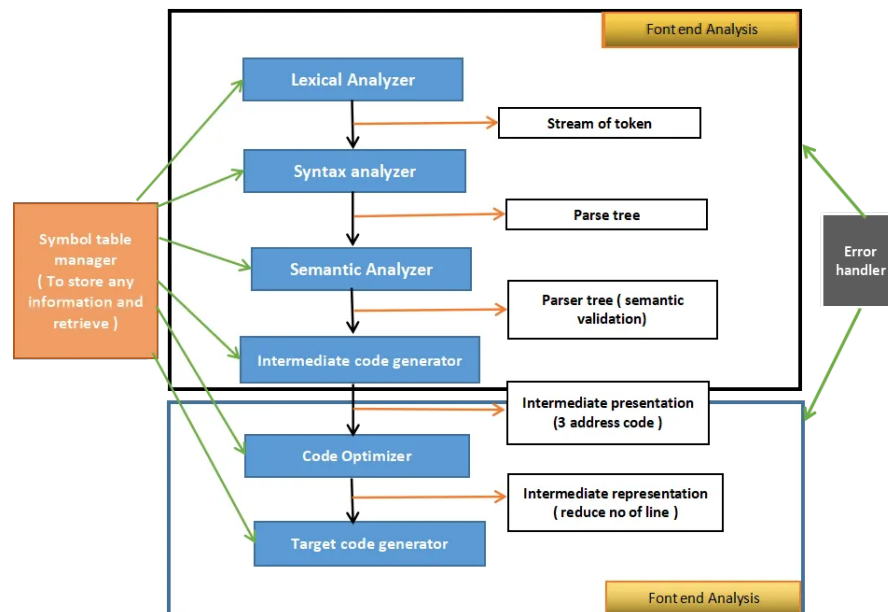
Group activity: Identify phases of compilation in a given example code

6. Conclusion (5 minutes)

- Summarize the key concepts covered in the lesson.
- Briefly discuss the evolving role of compilers in modern computing (e.g., optimizing for parallel architectures, supporting new programming paradigms).

Mind Map





Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge		1				
B. Conceptual Knowledge				2		
C. Procedural Knowledge					1	

PowerPoint Link: <https://docs.google.com/presentation/d/1y-yOngAUGfpPk7aM2nV3wJiDqlmu8EHm/edit?usp=sharing&oid=113456501758187936448&rtfpof=true&sd=true>

References

- Aho, Lam, Sethi, Ullman. *Compilers: Principles, Techniques, and Tools* (Dragon Book).
- Alfred V. Aho, Monica S. Lam. *Modern Compiler Implementation in C*.

Verified by Subject Expert:

Rhayan

Course In-charge

Rhayan

Approved by HoD

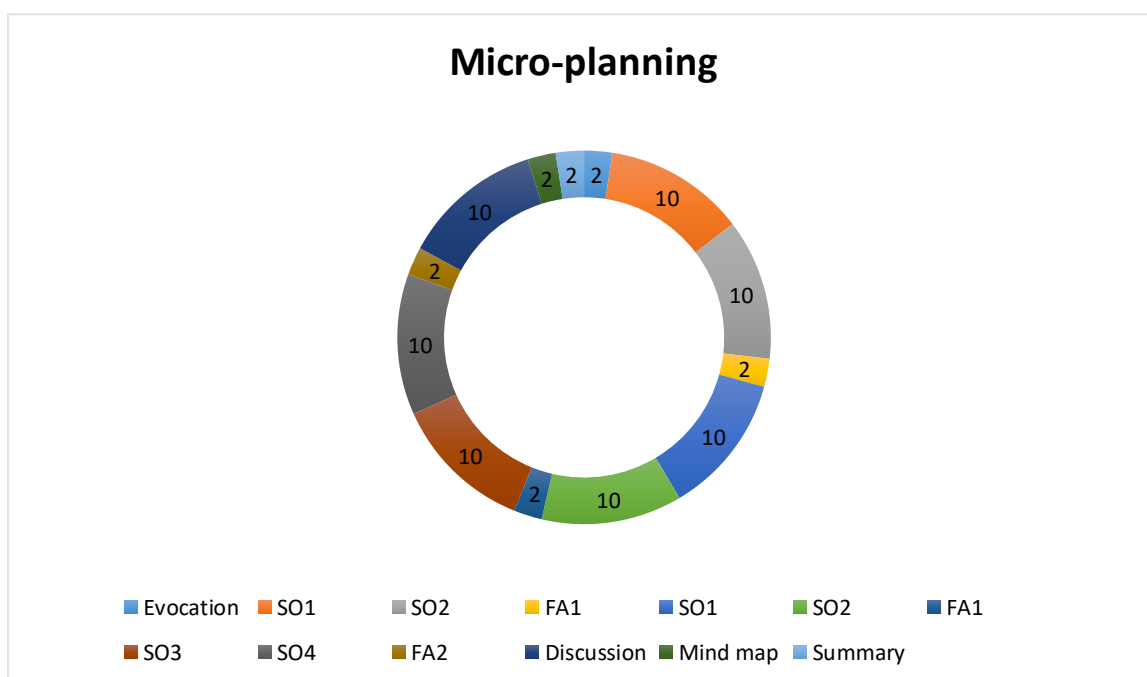


Lesson Plan

Programme	M.Sc. Computer Science
Semester	I
Course Title	Cryptography and Network Security
Code	21PCSE12
Hours	4
Total Hours	60
Credits	4
Max Marks	60
Unit & Title	Unit III & RSA Algorithm
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basics of cryptography

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

Encryption plays a crucial role in securing online communication. Ask students if they have ever used a password-protected system or heard about data breaches. Discuss how public key cryptography, like RSA, helps protect sensitive information such as online banking transactions and secure messaging.

2. Topic Introduction:

2.1: General Objective:

- To understand the fundamentals of the RSA Algorithm in public key cryptography.
- To learn its practical applications in secure communication.

2.2: Specific Outcomes:

- SO1: Explain the need for public key cryptography.
- SO2: Understand RSA algorithm concepts and key generation.
- SO3: Implement basic RSA encryption and decryption operations.
- SO4: Analyze RSA's role in cybersecurity and secure transactions.

First Phase:

SO1 (10 minutes):

- Define asymmetric cryptography and its significance.
- Explain the role of public and private keys.

SO2 (10 minutes):

- Discuss the history and development of the RSA Algorithm.
- Introduce key mathematical principles: prime factorization, modular arithmetic, Euler's theorem.

Second Phase:

SO3 (10 minutes):

- RSA Key Generation:
 - Choose two large prime numbers (p, q).
 - Compute $n = p \times q$ and Euler's totient function $\phi(n)$.
 - Select public exponent (e) and compute private exponent (d).

SO4 (10 minutes):

- Encryption: Use the public key (n, e) to encrypt messages.
- Decryption: Use the private key (n, d) to decrypt ciphertexts.

Mind Map (2 minutes):

- Simple mind map including the steps involved in RSA algorithm.

Discussion (10 min):

- Importance of RSA
- Strength of RSA

Summary (2 minutes):

- Summarize RSA's working principles and its importance in cybersecurity.

2.3: Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1	1				
B. Conceptual Knowledge			2	2		
C. Procedural Knowledge			2			

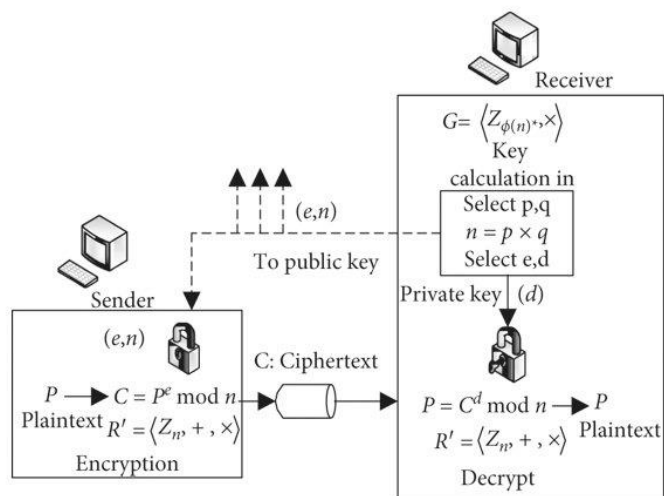
D. Meta Cognitive Knowledge					2,3	
-----------------------------------	--	--	--	--	-----	--

2.4: Key Words:

Public Key Cryptography, RSA Algorithm, Key Generation, Encryption, Decryption

PPT Link: https://docs.google.com/presentation/d/1p1rd-0zuE1aUY1-jgSeMHFme_v9BASnmixLxwSICkvw/edit?usp=sharing

2.5: Key Diagram:

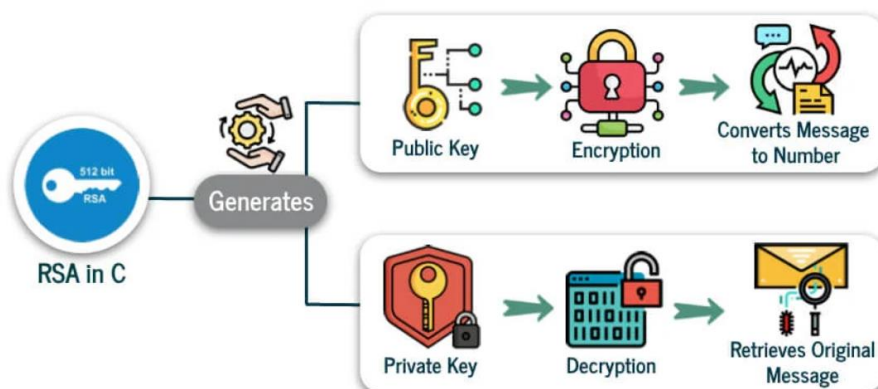


3. Discussion:

Students are asked to:

- **Limitations and Challenges:** Discuss RSA's computational intensity, key size requirements, and vulnerabilities like brute-force attacks if the key length is too short.
- **Real-world Use Cases:** Encourage students to explore how RSA is used in SSL/TLS certificates, blockchain technology, and government encryption standards.

4. Mind Map:



5. Summary:

The RSA algorithm is a fundamental method of public key cryptography, ensuring secure data transmission through asymmetric encryption. By leveraging prime factorization and modular arithmetic, RSA provides confidentiality and authentication. Its widespread use in digital signatures, online banking, and secure communications highlights its importance in modern cybersecurity.

6. Assessment:

Formative Assessment 1 (FA1) – 2 min:

- Quick quiz: Identify the steps in RSA key generation.

Formative Assessment 2 (FA2) – 2 min:

- Short discussion on the strengths and weaknesses of RSA.

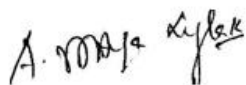
7. FAQs (MSQs/Descriptive Questions):

1. What is the RSA Algorithm, and how does it work?
2. How does RSA ensure secure communication?
3. Compare RSA with other cryptographic algorithms.
4. Explain the role of prime numbers in RSA.
5. Why is key length important in RSA security?

8. References:

1. William Stallings. Cryptography and Network Security Principles and Practices. 6th Edition, 2013.
2. Behrouz A. Ferouzan. Cryptography & Network Security, Tata McGraw Hill, 2007.
3. Charlie Kaufman, Radia Perlman and Mike Speciner. Network Security. Prentice Hall of India, 2002.

Verified by Subject Expert:



Course In-Charge



Approved by HoD



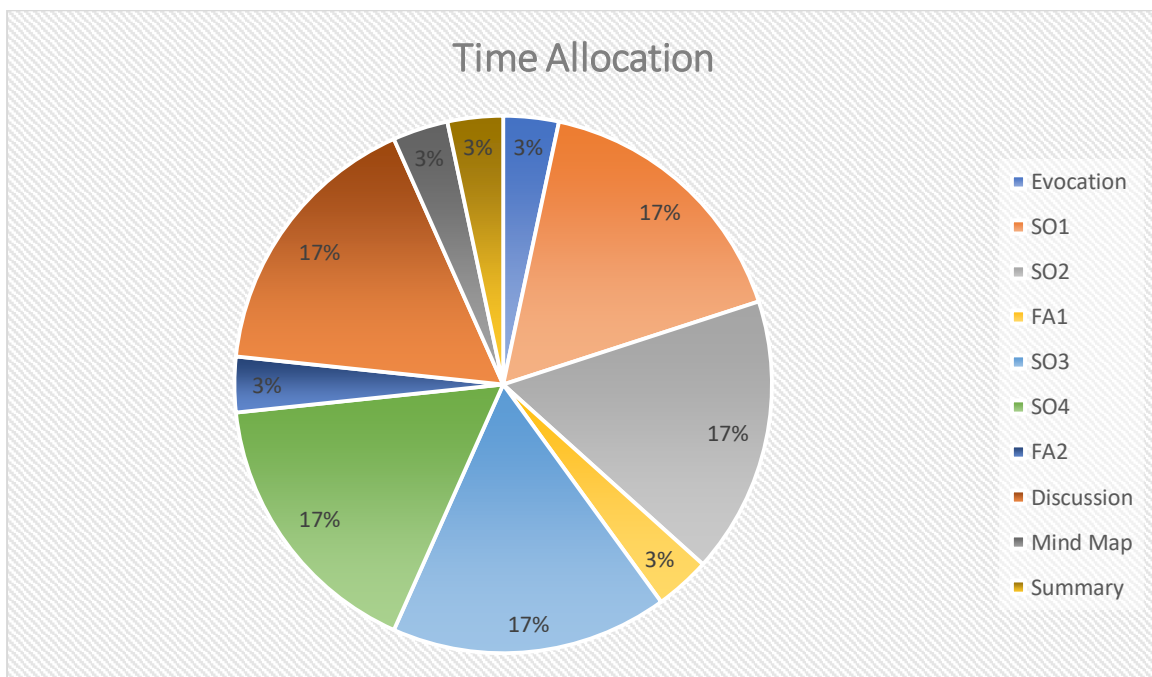
Lesson Plan

Programme	M.Sc Computer Science
Semester	II
Course Title	J2EE
Code	21PCSC21
Hours per Week	5
Total Hours	75
Credits	4
Maximum Marks	60
Unit & Title	Unit III: Servlet Filtering
Name of Faculty	A.Jenitta Jebamalar
T-L Tools	PowerPoint, Visual Demonstrations, Group Discussions, Interactive Coding Sessions

Pre-requisite Knowledge:

- Basics of Servlets and HTTP
- Understanding of Request and Response flow in Web Applications

Micro-Planning (60 minutes)



1. Evocation (2 min)

- Brief discussion on how Servlets work in web applications.
- Ask students about their understanding of request handling in web applications.

2. Topic Introduction

2.1 General Objectives:

- To understand what Servlet Filtering is and how it works.
- To learn how to apply Filters in a J2EE Web Application.

2.2 Specific Outcomes (SO):

- **SO1:** Explain what Servlet Filters are and their role in a web application.
- **SO2:** Describe the Filter lifecycle and implementation process.
- **SO3:** Differentiate between various types of Filters and their use cases.
- **SO4:** Demonstrate the process of implementing Filters in a Servlet-based application.

First Phase:

SO1 (10 min):

- Introduction to Servlet Filtering.
- Importance of Filters in Web Applications.

SO2 (10 min):

- Discuss the Filter lifecycle: `init()`, `doFilter()`, `destroy()`.
- Explain Filter chaining and request-response modification.

Second Phase:

SO3 (10 min):

- Types of Filters:
 - Authentication Filter
 - Logging and Auditing Filter
 - Compression Filter
 - XSS and Security Filter
 - Image Conversion Filter
- Real-world applications of Filters.

SO4 (10 min):

- Step-by-step implementation of a Servlet Filter:
 - Writing a Filter class implementing `javax.servlet.Filter`.
 - Configuring the filter in `web.xml` or using `@WebFilter`.
 - Demonstrating filter chaining.

Mind Map (2 min)

- Visual representation of how Servlet Filters intercept and modify request/response flow.

Summary (2 min)

- Servlet Filters enhance request processing without modifying Servlets directly.
- Filters are useful for security, logging, transformation, and request modification.

2.3 Taxonomy of Objectives

Taxonomy of Objectives						
Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Knowledge		1				
C. Procedural Knowledge			1	1		
D. Meta Cognitive Knowledge					1	1

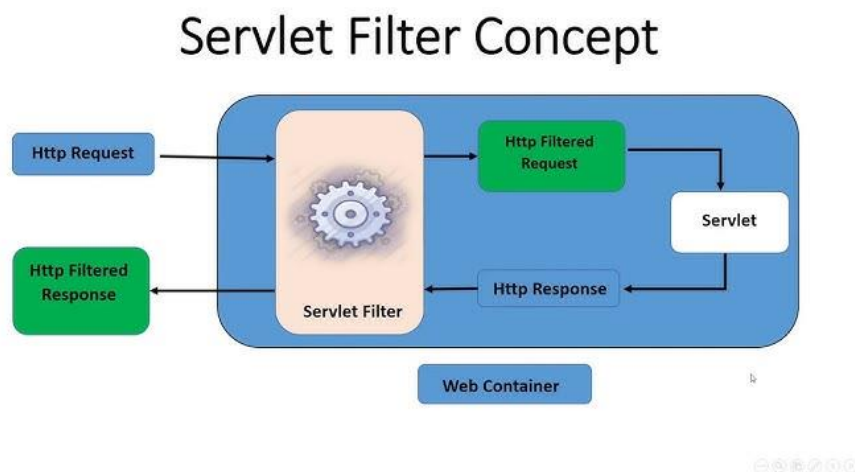
2.4 Powerpoint Presentation:

<https://gamma.app/docs/Servlet-Filteringpptx-0a5fgfz8nysqtzx>

3. Discussion (10 min)

- How can Filters be used for authentication and logging?
- Discuss real-world examples where Filters improve security and performance.

4. Mind Map



5. Summary

Servlet Filtering in J2EE allows intercepting and modifying HTTP requests and responses before reaching a Servlet or after processing. Filters are used for security, logging, compression, and request validation in web applications.

- **Lifecycle:** `init()`, `doFilter()`, `destroy()`
- **Types:** Authentication, Logging, Compression, Security, Image Conversion
- **Implementation:** Configured via `web.xml` or `@WebFilter` annotation
- **Use Cases:** Enhancing security, optimizing performance, and monitoring requests

6. Assessment

Formative Assessment 1 (FA1) (2 min):

- Quick Q&A: Explain the role of `doFilter()` in the Filter lifecycle.

Formative Assessment 2 (FA2) (2 min):

- Mini coding exercise: Write a simple logging filter in Java.

7. FAQ (Multiple Choice & Descriptive Questions)

1. What is the primary purpose of a Servlet Filter?
2. Explain the difference between Servlet Filters and Listeners.
3. How do you configure a Filter using annotations in Java?
4. What is Filter Chaining, and how does it work?

8. References

- Java Servlet Specification (Oracle Documentation)
- Online resources: Java EE Tutorials from Oracle
- Budi Kurniawan. Java for Web with Servlets, JSP and EJB: A Developer's Guide to J2EE Solutions. New Riders Publishing.

9. Verified by Subject Expert:

A. Leenita Lebanalar

Course In-Charge

Rhayan

Approved by HoD



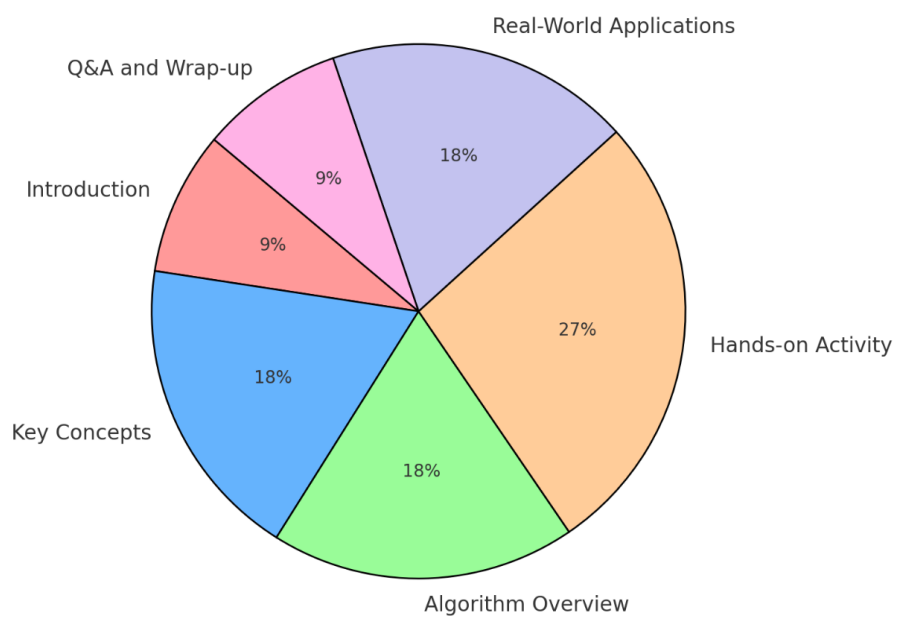
Lesson Plan

Programme	M.Sc
Semester	II
Course Title	DATA MINING & R PROGRAMMING
Code	21PCSC22
Hours	5
Total Hours	75
Credits	4
Max Marks	100
Unit & Title	v / Association rules
Name of the Faculty	C.Nayanthra Mascarenhas
T-L tools	PowerPoint

Learning Objective: By the end of the lesson, students will:

- Understand association rules and their significance.
- Learn key concepts like support, confidence, and lift.
- Explore the Apriori and FP-Growth algorithms.
- Apply concepts through a hands-on activity.
- See real-world applications of association rules.

Microteaching Plan: Association Rules in Data Mining



•

Lesson Structure

1. Introduction (5 min - 8%)

- Define **Association Rules**: Explain the concept with an example (e.g., Market Basket Analysis – "If a customer buys bread, they are likely to buy butter").
- Why is it important? Explain how businesses use association rules in retail, healthcare, and recommendation systems.

2. SO1-Key Concepts (10 min - 17%)

- **Support**: Frequency of an itemset in the dataset.
- **Confidence**: Likelihood that a rule is true (e.g., $P(\text{Butter} | \text{Bread})$).
- **Lift**: Measures if the rule is better than random chance.
- Example calculations with a small dataset.

3. SO2-Algorithm Overview (10 min - 17%)

- **Apriori Algorithm**: Explain step-by-step how frequent itemsets are found.

4. Hands-on Activity (15 min - 25%)

- **Dataset Example**: Provide a small transaction dataset.
- **Task**: Students manually compute support, confidence, and lift for given itemsets.
- **Discussion**: Compare results and clarify doubts.

5. Real-World Applications (10 min - 17%)

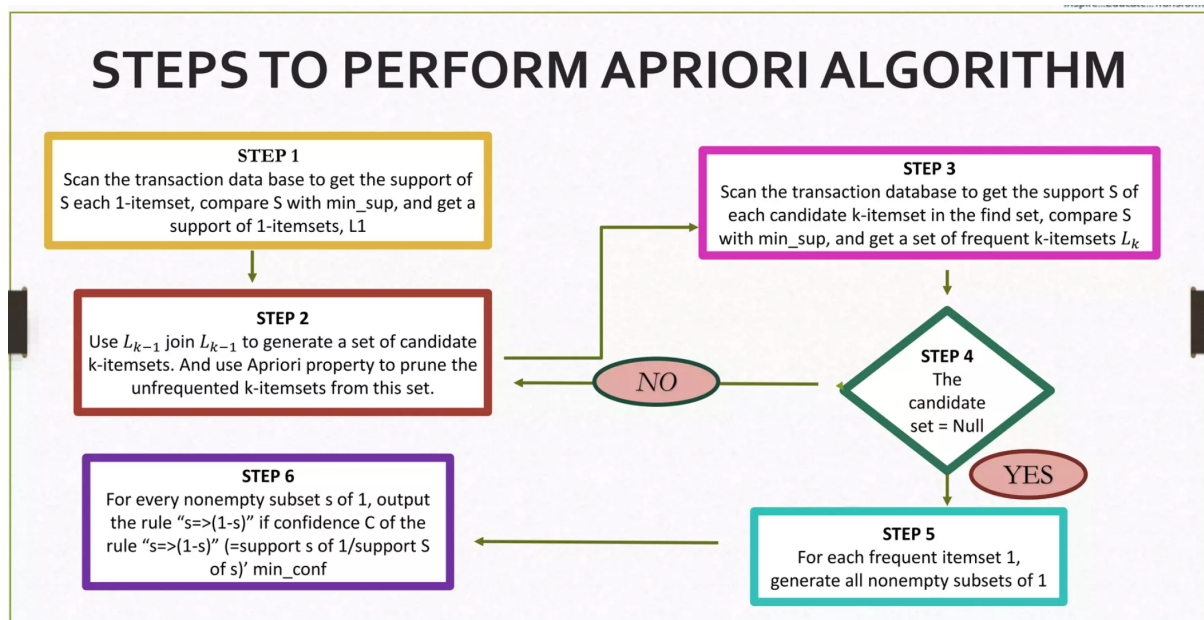
- Retail: Market Basket Analysis (Amazon, Walmart).
- Healthcare: Drug interactions and disease prediction.
- Fraud Detection: Identifying unusual transaction patterns.

6. Q&A and Wrap-up (5 min - 8%)

1. What are **association rules** in data mining?
2. How is **market basket analysis** related to association rules?
3. What is the difference between **support, confidence, and lift**?

Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge		1				
B. Conceptual Knowledge			2			
C. Procedural Knowledge				2		



APRIORI ALGORITHM EXAMPLE

Database D
Minsup = 0.5

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Scan D

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

itemset	sup
{2 3 5}	2

Scan D

itemset	sup
{2 3 5}	2

Reference:

1. Margaret H. Dunham. Data Mining Introductory and Advanced Topics. Pearson publications, Ninth Impression, 2002.
2. Bing Liu. Web Data Mining: Exploring Hyperlinks, Contents and Usage Data. Springer, 2006.

PPT Link :

https://docs.google.com/presentation/d/1UVLie9IFUjsr4G_ek_ZUhZySe2K38CB6/edit?usp=s_haring&ouid=113456501758187936448&rtpof=true&sd=true

Verified by Subject Expert:

Rhayan

Course In-charge

Rhayan

Approved by HoD



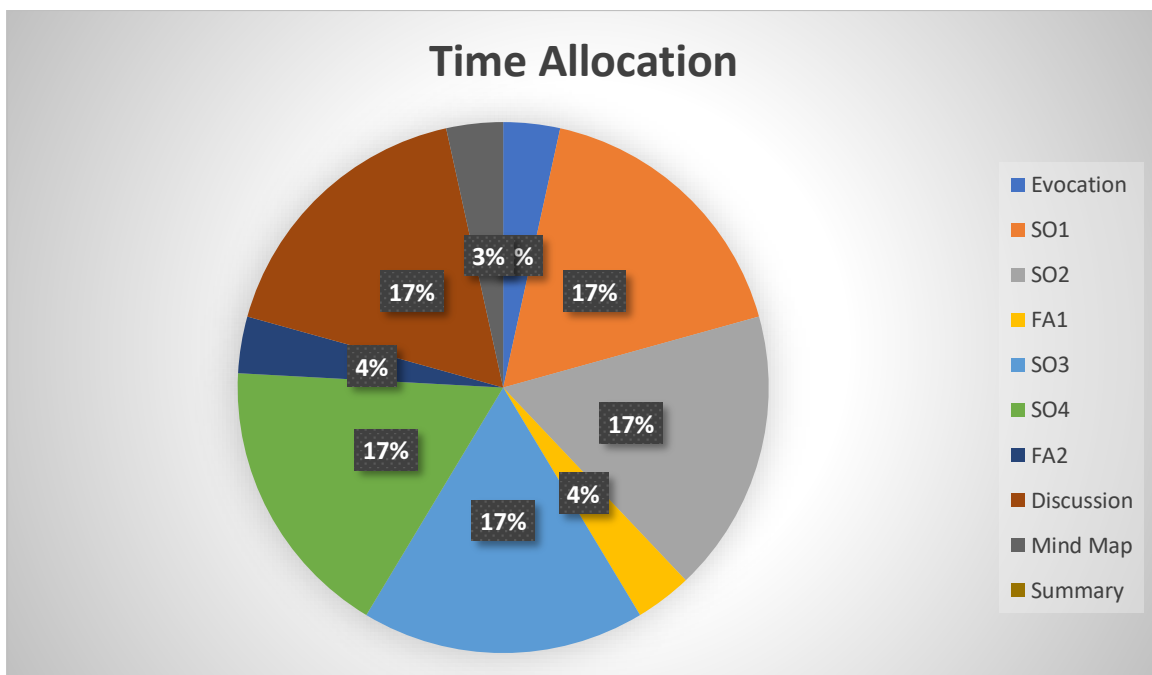
Lesson Plan

Programme	M.Sc Computer Science
Semester	II
Course Title	Distributed Database Management System
Code	21PCSC23
Hours per Week	4
Total Hours	60
Credits	4
Maximum Marks	60
Unit & Title	Unit I: Transparency Issues
Name of Faculty	A.Jenitta Jebamalar
T-L Tools	PowerPoint, Visual Demonstrations, Group Discussions, Interactive Coding Sessions

Pre-requisite Knowledge:

- Basics of Databases and Database Management Systems (DBMS)
- Understanding of Centralized vs. Distributed Databases

Micro-Planning (60 minutes)



1. Evocation (2 min)

- Brief discussion on DDBMS and how it differs from centralized databases.
- Ask students about challenges in managing distributed databases.

2. Topic Introduction

2.1 General Objectives:

- To understand the concept of transparency in DDBMS.
- To analyze issues faced in ensuring transparency in a distributed environment.

2.2 Specific Outcomes (SO):

- **SO1:** Define transparency and its significance in DDBMS.
- **SO2:** Identify different types of transparency in distributed databases.
- **SO3:** Explain key transparency issues and their impact on performance.
- **SO4:** Discuss solutions and best practices for handling transparency challenges.

First Phase:

SO1 (10 min):

- Define Transparency in a DDBMS.
- Why is transparency important for data consistency, availability, and management?

SO2 (10 min):

- Discuss the different types of transparency:
 - Data Distribution Transparency – Users shouldn't need to know where data is stored.
 - Replication Transparency – Ensures multiple copies appear as one.
 - Failure Transparency – Database remains functional despite failures.
 - Access Transparency – Uniform access regardless of location.
 - Concurrency Transparency – Multiple users can access the database without conflicts.
 - Performance Transparency – Ensuring optimal query performance across locations.

Second Phase:

SO3 (10 min):

- Key Transparency Issues in DDBMS:
 - Complexity in maintaining data consistency.
 - Network latency affecting performance.
 - Replication conflicts when updating multiple copies.
 - Security concerns across distributed locations.

SO4 (10 min):

- **Case Study Discussion:**
 - Real-world challenges in implementing transparency in banking systems, e-commerce, and cloud databases.

Mind Map (2 min)

- Visual representation of transparency layers in DDBMS.

Summary (2 min)

- Transparency in DDBMS ensures users don't need to worry about data location, replication, or failure management.
- However, transparency introduces complexity, performance trade-offs, and security risks.

2.3 Taxonomy of Objectives

Taxonomy of Objectives						
Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Knowledge		1				
C. Procedural Knowledge			1	1		
D. Meta Cognitive Knowledge					1	1

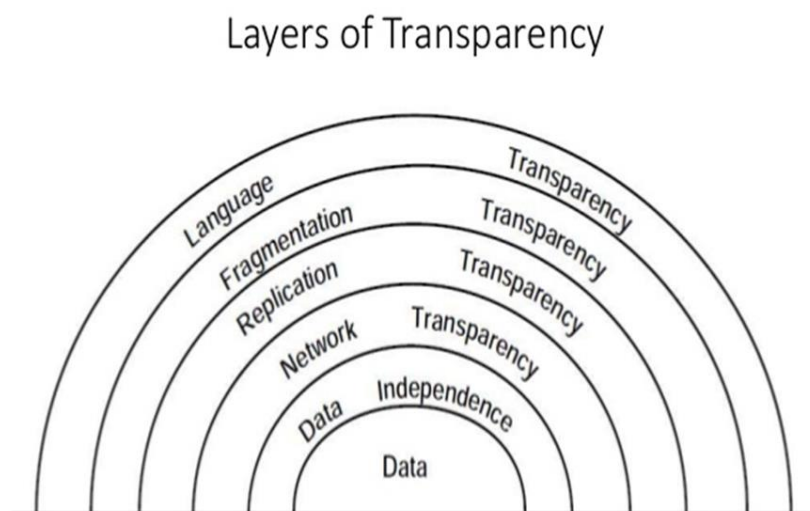
2.4 Powerpoint Presentation:

<https://gamma.app/docs/Transparency-Issuespptx-41p2lxb8o5dcrlc>

3. Discussion (10 min)

- How do organizations balance **transparency vs. performance** in DDBMS?
- Examples of **transparency issues** in real-world applications.

4. Mind Map



5. Summary

In a Distributed Database Management System (DDBMS), transparency ensures that users can access and manipulate data without needing to know its physical location, replication details, or system failures. While transparency improves usability, it also introduces several challenges.

Key Transparency Issues:

1. Data Distribution Transparency – Managing data across multiple locations without user intervention.
2. Replication Transparency – Ensuring consistency when data is replicated across sites.
3. Failure Transparency – Maintaining availability despite system or network failures.
4. Access Transparency – Providing a uniform way to access data, regardless of location.

5. Concurrency Transparency – Handling multiple simultaneous transactions without conflicts.
6. Performance Transparency – Balancing load distribution and query optimization across the network.

Challenges:

- Maintaining consistency in replicated data.
- Performance overhead due to synchronization and network latency.
- Security concerns when data is distributed across different locations.

Effectively managing transparency issues in DDBMS is critical for ensuring data integrity, system reliability, and optimal performance in modern distributed environments

6. Assessment

Formative Assessment 1 (FA1) (2 min):

- Quick Q&A: List and explain two types of transparency in DDBMS.

Formative Assessment 2 (FA2) (2 min):

- Mini activity: Analyze a real-world scenario where failure transparency is needed.

7. FAQ (Multiple Choice & Descriptive Questions)

1. What is Transparency in DDBMS?
2. Explain Replication Transparency and why it is important.
3. How does Failure Transparency help in disaster recovery?
4. What are some challenges in implementing transparency in a distributed system?

8. References

- Principles of Distributed Database Systems – M. Tamer Özsu & Patrick Valduriez
- Database System Concepts – Abraham Silberschatz, Henry F. Korth
- Online Tutorials: IBM Distributed Databases, Oracle DDBMS Documentation

9. Verified by Subject Expert

A. Leenitha Lebamalar

Course In-Charge

R. Jayan

Approved by HoD

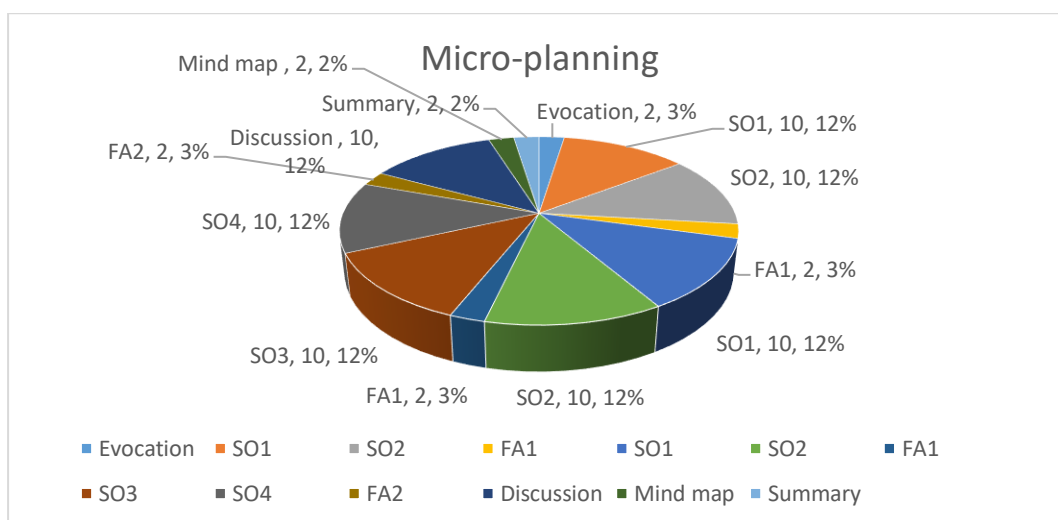


Lesson Plan

Programme	M.Sc. Computer Science
Semester	II
Course Title	Single Board Computers and IoT
Code	21PCSC24
Hours	4
Total Hours	60
Credits	4
Max Marks	60
Unit & Title	Unit IV & Communication Protocols
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basic understanding of networking, IoT architecture, and wireless communication

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

Brief explanation of communication protocols in IoT. Ask students to share their knowledge of how IoT devices communicate.

2. Topic Introduction:

2.1: General Objective:

- To understand different communication protocols in IoT
- To learn their real-world applications.

2.2: Specific Outcomes:

- SO1: Identify and differentiate between various IoT communication protocols such as MQTT, CoAP, HTTP, Zigbee, Bluetooth, and LoRa.
- SO2: Analyze the suitability of different protocols for specific IoT scenarios, such as smart homes, industrial automation, and healthcare.

- SO3: Demonstrate how to set up a basic IoT network using at least one communication protocol
- SO4: Evaluate the security and efficiency aspects of different IoT protocols in relation to data transmission and energy consumption.

First Phase:

SO1 (10 minutes):

- Introduction to communication protocols
- Classification of protocols (Wired & Wireless).

SO2 (10 minutes):

- Discuss protocol requirements for IoT (low power, security, scalability)

Second Phase:

SO3 (10 minutes):

- Explain different IoT protocols:
 - Application Layer: MQTT, CoAP, HTTP
 - Network Layer: IPv6, 6LoWPAN
 - Data Link & Physical Layer: Zigbee, Bluetooth, LoRa, Wi-Fi

SO4 (10 minutes):

- Explore real-world examples of protocol usage in smart homes, healthcare, and industry

Mind Map (2 minutes):

- Simple mind map including all the communication protocols involved in Internet of Things

Summary (2 minutes):

- Discussion on the advantages and disadvantages of various protocols.

2.3: Taxonomy of objectives:

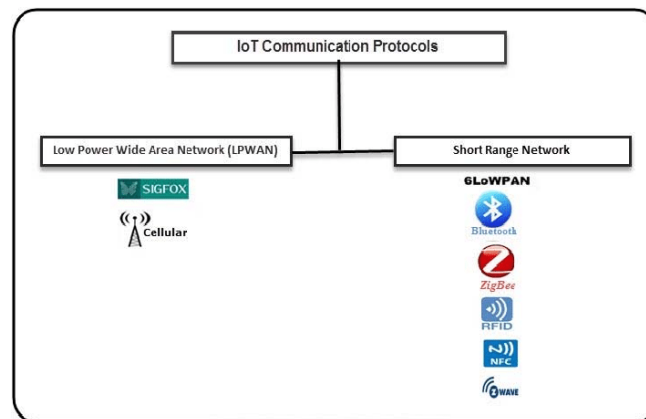
Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyze	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Knowledge		1				
C. Procedural Knowledge			2	2		
D. Meta Cognitive Knowledge				2	2,3	2,3

2.4: Key Words:

IoT Protocols, MQTT, CoAP, 6LoWPAN, Zigbee, LoRa

PPT Link: <https://docs.google.com/presentation/d/1xNysAenWI9gZjeP36ic-T31-jKhB3yrIt8gekq2nnVk/edit?usp=sharing>

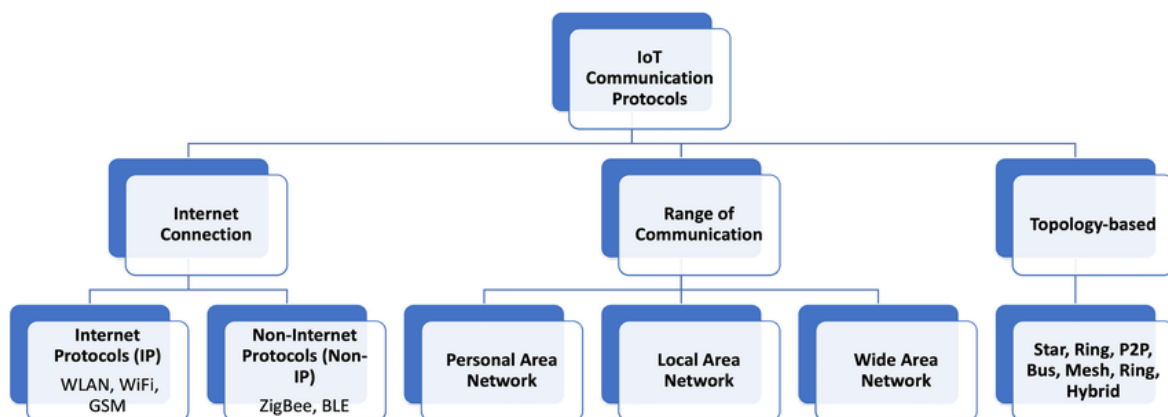
2.5: Key Diagram:



3. Discussion:

- Students discuss and analyze various IoT protocols.
- Case study: Selecting the right protocol for a smart city application.

4. Mind Map:



5. Summary:

- Communication in IoT relies on various protocols designed for efficiency, low power, and security.
- Different protocols serve different purposes, e.g., MQTT for lightweight messaging, Zigbee for short-range communication..

6. Assessment:

Formative Assessment 1 (FA1):

- Identify a real-world IoT device and the protocol it uses.

Formative Assessment 2 (FA2):

- Quick review: Compare two protocols based on speed, security, and energy consumption.

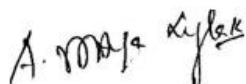
7. FAQs (MSQs/Descriptive Questions):

1. What are IoT communication protocols? List their types.
2. Compare and contrast MQTT and CoAP.
3. Which protocol is best for a smart agriculture system? Justify.

8. References:

1. ArshdeepBahga and Vijay Madisetti. Internet of Things-A Hands-on Approach. Universities Press Amir (India), 2015.
2. RajkumarBuyya and VahidDastjerdi. Internet of Things: Principles and Paradigms. Cloud Computing and Distributed Systems (Clouds) Laboratory, Manja Soft Pty Ltd., Australia, 2016.

Verified by Subject Expert:



Course In-Charge



Approved by HoD

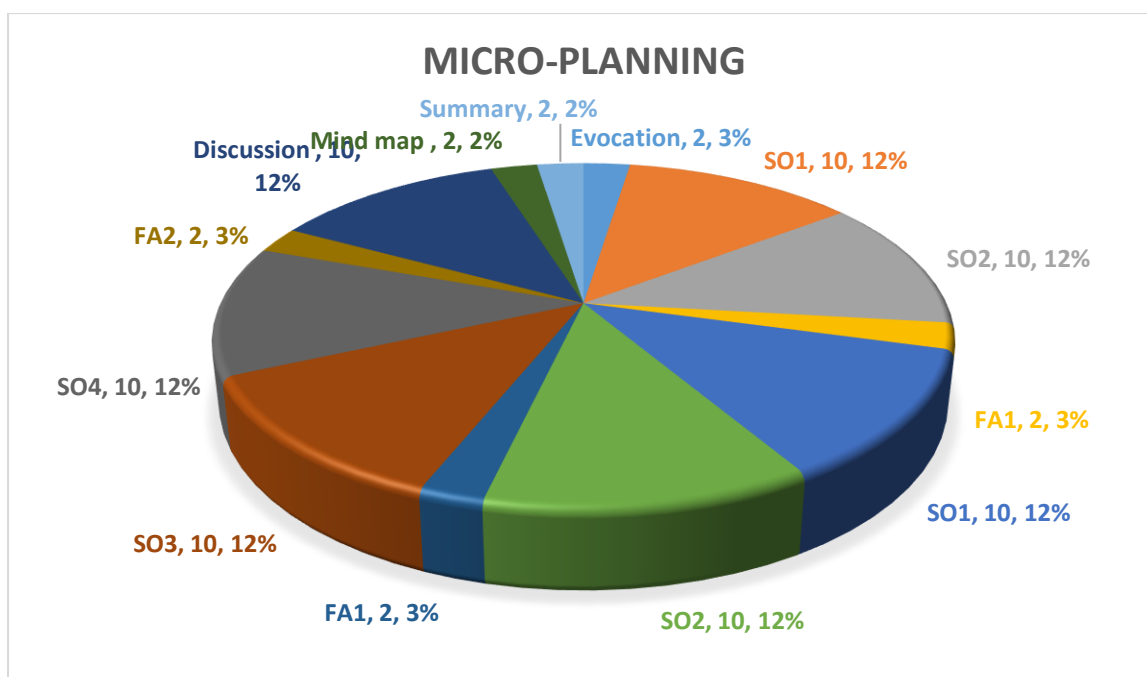


Lesson Plan

Programme	M.Sc. Computer Science
Semester	II
Course Title	Soft Computing
Code	21PCSE22
Hours	4
Total Hours	60
Credits	4
Max Marks	60
Unit & Title	Unit II & Back Propagation Network
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basics of Artificial Neural Networks and Understanding of Supervised Learning

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

Brief explanation about backpropagation and its usage in neural networks. Students share prior knowledge and experience with neural networks.

2. Topic Introduction:

2.1: General Objective:

- To understand the working and importance of backpropagation in training neural networks.
- To learn the practical application of backpropagation in real-world problems

2.2: Specific Outcomes:

- SO1: To understand the different layers in a backpropagation network.
- SO2: To learn the step-by-step process of backpropagation.
- SO3: To analyze activation functions and their impact.
- SO4: To evaluate the advantages and limitations of backpropagation.

First Phase:

SO1 (10 minutes):

- Definition and importance of backpropagation.
- Real-world applications of backpropagation in neural networks.

SO2 (10 minutes):

- Input layer, hidden layers, and output layer.
- Explanation of neuron connections and weight updates.

Second Phase:

SO3 (10 minutes):

- Explanation of Sigmoid, Tanh, and ReLU functions.
- Their roles in optimizing neural network performance.

SO4 (10 minutes):

- Backpropagation Algorithm
- Initializing weights and biases
- Forward propagation process
- Error computation using loss functions (MSE)
- Backward propagation (gradient descent for weight updates)
- Repeating until convergence

Mind Map (2 minutes):

- Simple mind map for backpropagation network in neural networks

Discussion (10 min):

- Discuss real-world case studies where backpropagation led to breakthroughs in AI.

Summary (2 minutes):

- Summarize the key concepts of backpropagation and its importance in training neural networks.

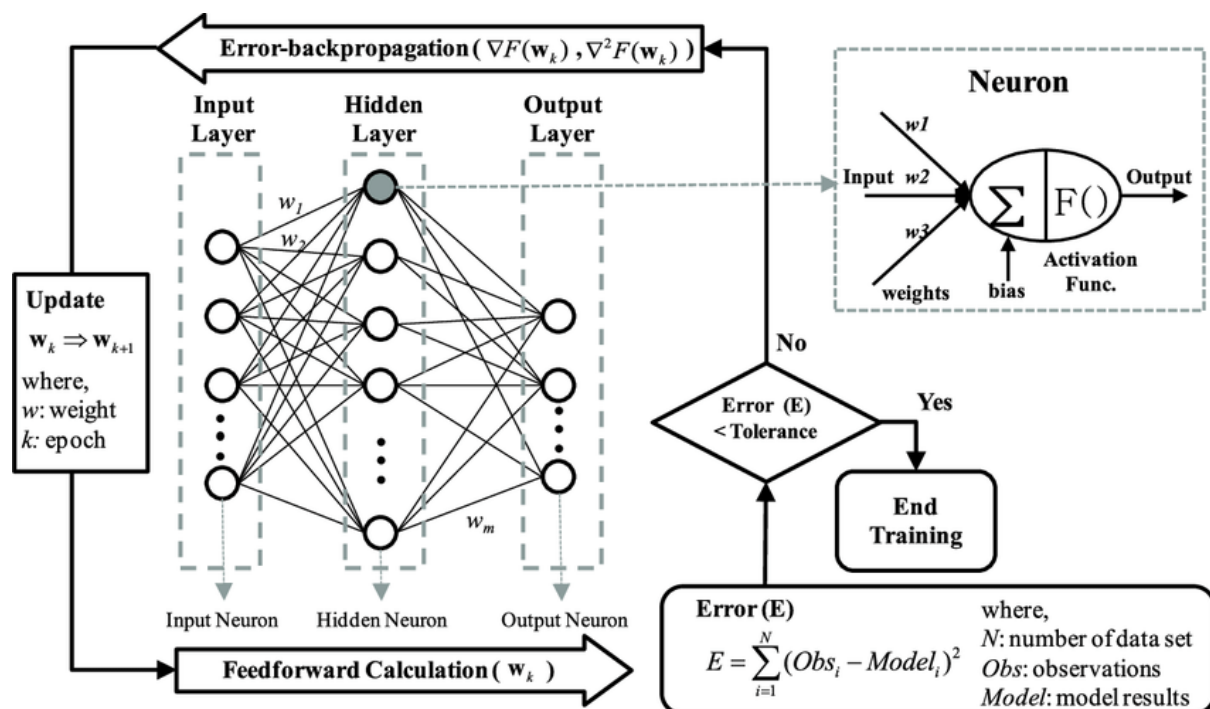
2.3: Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Knowledge	1					
C. Procedural Knowledge		1	2	2		
D. Meta Cognitive Knowledge		1				2,3

2.4: Key Words:

Backpropagation, Neural Networks, Activation Functions, Gradient Descent

2.5: Key Diagram:



PPT Link:

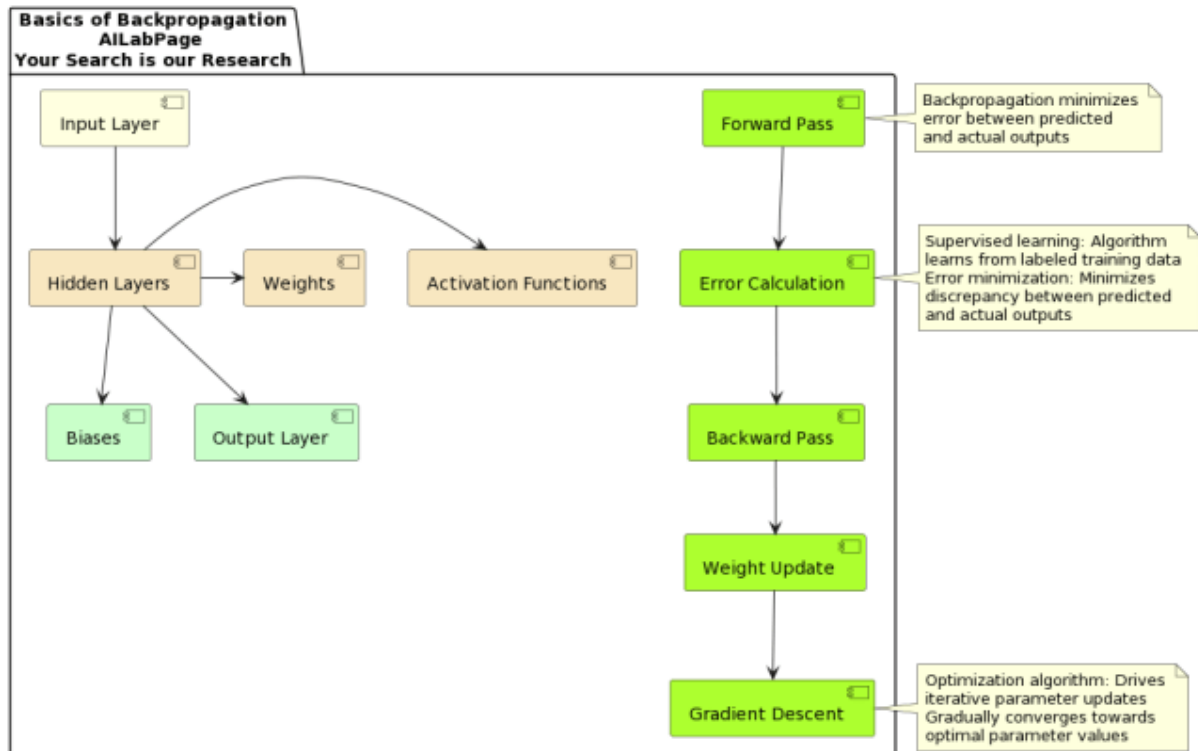
https://docs.google.com/presentation/d/1o5GDiCZWKiJdaH69bS2mlO4OJpgr80ibgHvjPPP_Lsk/edit?usp=sharing

3. Discussion:

Students are asked to:

- Explore possible modifications to the backpropagation algorithm to improve efficiency.

4. Mind Map:



5. Summary:

A backpropagation network is a type of artificial neural network that learns by adjusting its weights through backpropagation. It uses a supervised learning algorithm where errors are propagated backward from the output layer to the input layer, allowing the network to minimize errors. This process involves calculating gradients and updating weights using optimization techniques like gradient descent.

6. Assessment:

Formative Assessment 1 (FA1) – 2 min:

- Quick quiz on architecture and working of backpropagation.
-

Formative Assessment 2 (FA2) – 2 min:

- Compare different activation functions in backpropagation.

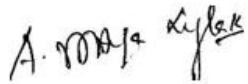
7. FAQs (MSQs/Descriptive Questions):

1. What is backpropagation and why is it important?
2. Explain the different layers in a backpropagation network.
3. How does gradient descent help in backpropagation?
4. Compare Sigmoid, Tanh, and ReLU activation functions.
5. Discuss two real-world applications of backpropagation networks.

8. References:

1. Soft Computing: Fundamentals and Applications by S.N. Sivanandam and S. Sumathi.
2. Soft Computing and Intelligent Systems Design by Fakhreddine O. Karray and Mohamed S. Deen
3. Introduction to Soft Computing by S. Rajasekaran and G. A. Vijayalakshmi Pai

Verified by Subject Expert:



Course In-Charge



Approved by HoD

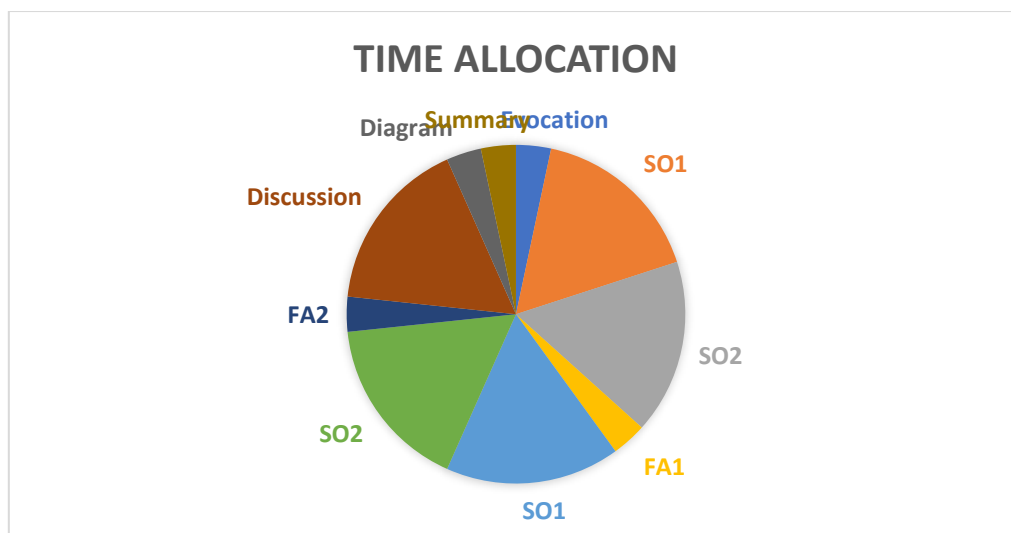


Lesson Plan

Programme	M.Sc Computer Science
Semester	III
Course Title	Software Testing
Code	21PCSC31
Hours per Week	4
Total Hours	60
Credits	4
Maximum Marks	60
Unit & Title	Unit I: Whitebox Testing
Name of Faculty	A.Jenitta Jebamalar
T-L Tools	Diagrams, PowerPoint Presentation, Hands-on Practice, Group Discussion

Pre-requisite Knowledge: Basics of software testing, understanding of program control structures (loops, branches, etc.)

Micro-Planning: 60 minutes



1.Topics for Learning through Evocation:

Brief explanation of Whitebox Testing and its importance in ensuring code coverage. Ask students to share their initial thoughts on testing internal structures.

2. Topic Introduction:

2.1 General Objective:

- To understand the principles and practices of Whitebox Testing.
- To apply Whitebox Testing techniques in real-world scenarios.

2.2 Specific Outcomes:

- To identify different Whitebox Testing techniques (SO1).
- To demonstrate the application of these techniques through examples (SO2).

First Phase:

- **SO1 (10 minutes):** Introduce Whitebox Testing and explain its types: statement coverage, branch coverage, and path coverage.
- **SO2 (10 minutes):** Discuss the significance of these techniques in different types of software applications.

Second Phase:

- **SO1 (10 minutes):** Detail the process of performing statement coverage and branch coverage with sample code snippets.
- **SO2 (10 minutes):** Explore practical examples of path coverage and discuss how it ensures thorough testing.

Diagram (2 minutes):

Create a flowchart illustrating the differences and applications of statement, branch, and path coverage in software testing.

Summary (2 minutes):

Summarize the benefits and challenges of Whitebox Testing, emphasizing the importance of code coverage in software quality assurance.

2.3 Taxonomy of Objectives:

Taxonomy of Objectives						
Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Know		1				
C. Procedural Knowledge			1	1		
D. Meta Cognitive Knowledge					1	

2.4 Keywords:

Whitebox Testing, Statement Coverage, Branch Coverage, Path Coverage, Code Coverage

2.5 Powerpoint Presentation

<https://gamma.app/docs/Whitebox-Testing-in-Software-Testing-6kz9yid4l48f9fi>

3. Discussion:

Students will engage in a group discussion on the advantages and limitations of Whitebox Testing and will be tasked to frame a mini-project applying these techniques.

4. Mind Map



5. Summary

Whitebox Testing is a software testing technique that examines the internal structure, code, and logic of an application. Testers use knowledge of the code to verify different paths, conditions, and loops, ensuring thorough test coverage. Key techniques include statement coverage, branch coverage, and path coverage. This method helps identify hidden errors, optimize performance, and improve software quality by testing how the system processes inputs internally.

6. Assessment:

- **FA1 (2 minutes):** Students explain the different types of Whitebox Testing.
- **FA2 (2 minutes):** Quick review of a case study on Whitebox Testing application.

7. FAQ's / Descriptive Questions:

1. What is Whitebox Testing? List its types.
2. Compare and contrast Statement Coverage and Branch Coverage.

8. References:

- Kshirasagar. *Software Testing and Quality Assurance, Theory and Practice*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2008
- Srinivasan Desikan and Gopalswami Ramesh. *Software Testing: Principles and Practices*. Pearson Education, 1st Edition, 2008.

9. Verified by Subject Expert:

A. Leelitha Lebamalar

Course In-Charge

R. Jayan

Approved by HoD

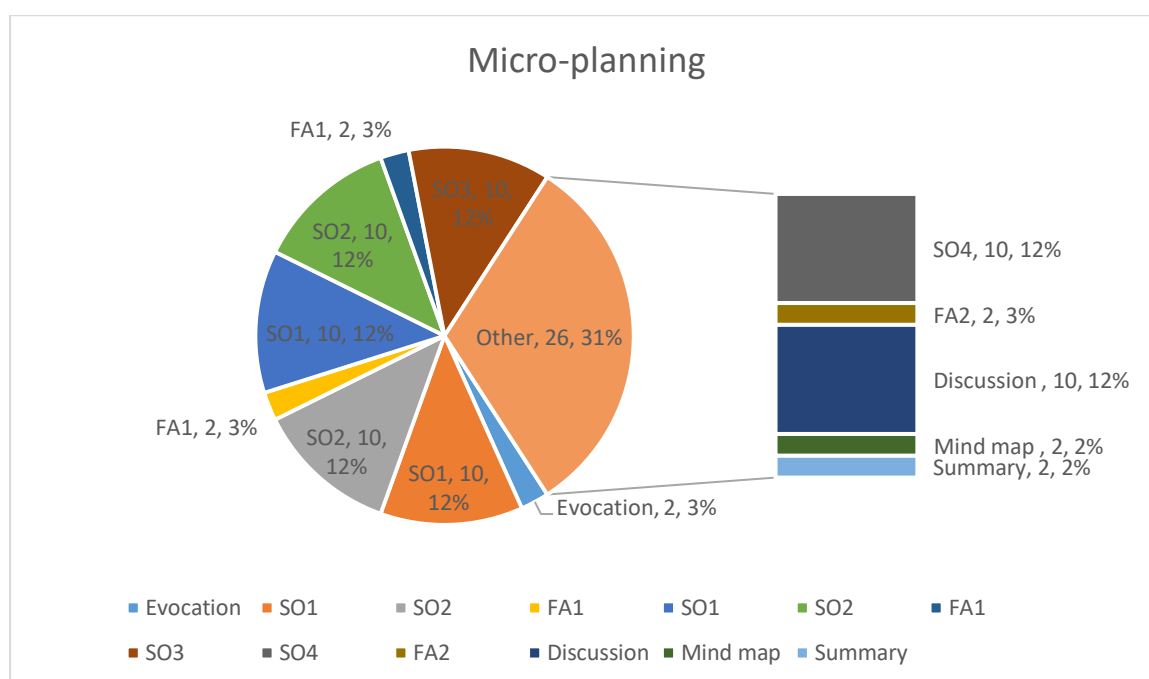


Lesson Plan

Programme	M.Sc. Computer Science
Semester	III
Course Title	Cloud Computing and Big Data
Code	21PCSC32
Hours	4
Total Hours	60
Credits	4
Max Marks	60
Unit & Title	Unit IV & MapReduce
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basic understanding of research methodology

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

Brief explanation about distributed computing and its role in cloud computing. Ask students to share their knowledge about data processing in large-scale environments.

2. Topic Introduction:

2.1: General Objective:

- To understand the concept of MapReduce in Cloud Computing.
- To learn its applications in processing large-scale data.

2.2: Specific Outcomes:

- SO1: To define MapReduce and its importance in cloud environments.
- SO2: To understand the working mechanism of Map and Reduce functions.
- SO3: To analyze the advantages and challenges of MapReduce.
- SO4: To apply MapReduce in real-world big data scenarios.

First Phase:

SO1 (10 minutes):

- Explain the concept of MapReduce and why it is essential in cloud computing.

SO2 (10 minutes):

- Discuss the Map and Reduce functions with simple examples.

Second Phase:

SO3 (10 minutes):

- Clarify the workflow of a MapReduce job, including input splitting, mapping, shuffling, reducing, and output generation.

SO4 (10 minutes):

- Explore practical applications of MapReduce in large-scale data processing such as log analysis and recommendation systems.

Mind Map (2 minutes):

- Simple mind map for MapReduce concept

Summary (2 minutes):

- Summarize the key points about MapReduce through a group discussion.

2.3: Taxonomy of objectives:

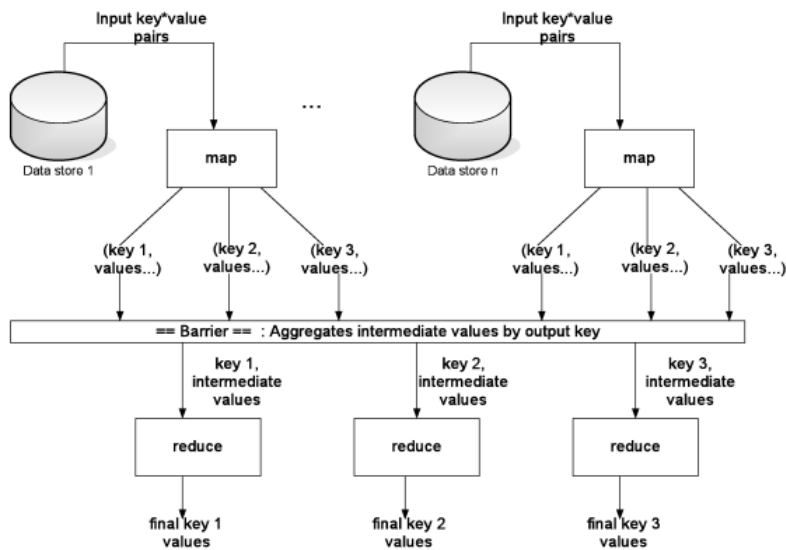
Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyze	Evaluate	Create
A. Factual Knowledge	1	1				
B. Conceptual Knowledge		1	2			
C. Procedural Knowledge			2	2		
D. Meta Cognitive Knowledge					3	3

2.4: Key Words:

MapReduce, Hadoop, Big Data, Parallel Processing, Distributed Computing, Cloud Computing

PPT Link: https://docs.google.com/presentation/d/1M4fpr69hFUmiGfR9X7ESNZPzYpCY-vY6xmb-B_TwH7M/edit?usp=sharing

2.5: Key Diagram:



3. Discussion:

Students are asked to discuss different real-world applications of MapReduce and frame a case study on using MapReduce for big data analytics

4. Mind Map:



5. Summary:

MapReduce is a powerful model for processing large datasets in cloud environments. It enables parallel and distributed processing, making it highly scalable and efficient. The model consists of two core functions—Map and Reduce—which work together to process data efficiently.

6. Assessment:

Formative Assessment 1 (FA1):

- Students are asked to differentiate between Map and Reduce functions with examples.

Formative Assessment 2 (FA2):

- A quick review on the advantages and limitations of MapReduce in cloud computing.

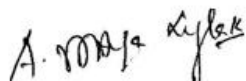
7. FAQs (MSQs/Descriptive Questions):

1. What is MapReduce? Explain its significance in cloud computing.
2. Compare and contrast the Map and Reduce functions.
3. Discuss the advantages and challenges of MapReduce.

8. References:

1. Rajkumar Buyya, Christian Vecchiola and S.Thamarai Selvi. Mastering Cloud Computing. TMGH,2013.
2. Kai Hwang, Geoffrey C Fox and Jack G Dongarra.Distributed and Cloud Computing, From Parallel Processing to the Internet of Things. Morgan Kaufmann Publishers, 2012.

Verified by Subject Expert:



Course In-Charge

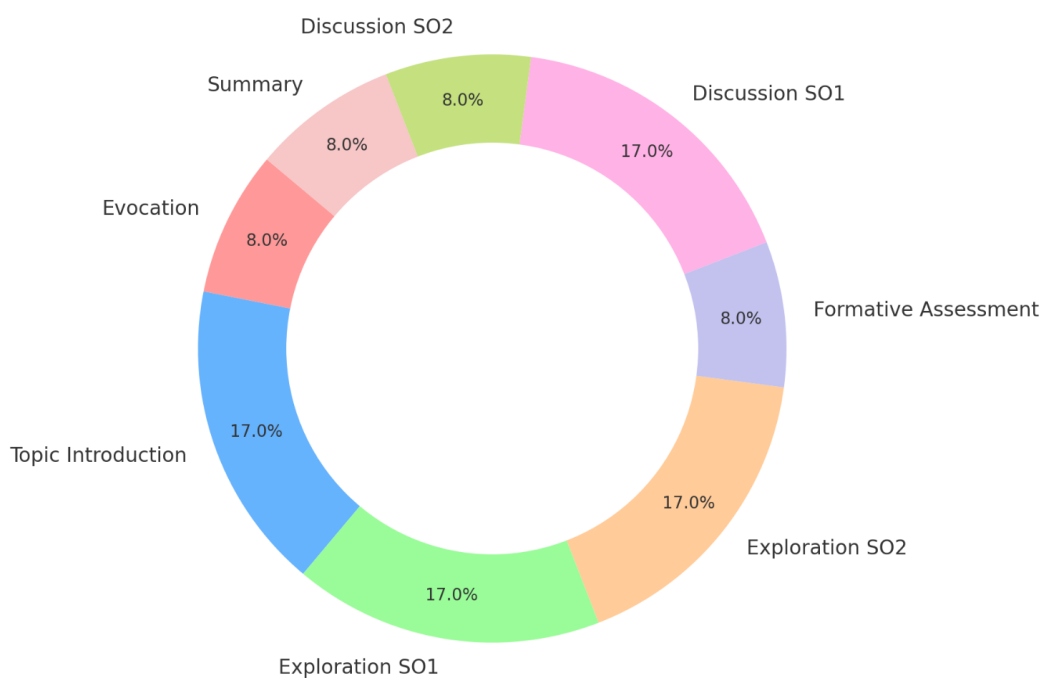


Approved by HoD



Programme	M.Sc
Semester	III
Course Title	Data Science using Python
Code	21PCSC33
Hours	4
Total Hours	60
Credits	4
Max Marks	100
Unit & Title	III / Basics of Numpy Array
Name of the Faculty	C.Nayanthra Mascarenhas
T-L tools	PowerPoint, Python lab ,Cheat sheet

Microteaching Plan for Basics of NumPy (60 minutes)



Lesson Plan: Basics of NumPy (60 minutes)

1. Evocation (5 minutes) – 8%

- Brief introduction to NumPy.

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

- Ask students what they know about arrays and numerical computing

2. Topic Introduction (10 minutes) – 17%

- **Objective:** Understand the purpose and importance of NumPy
- Explain why NumPy is used in Python for scientific computing

3. Exploration (25 minutes) – 42%

- **SO1 (10 minutes) – 17%:** Explain NumPy arrays vs. Python lists.
- **SO2 (10 minutes) – 17%:** Demonstrate basic operations (array creation, indexing, slicing)

1. Array Creation

To use NumPy, first, import it:

```
import numpy as np
```

Creating an array from a list:

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

Creating a 2D array (matrix):

```
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix)
```

Creating arrays with zeros, ones, and random values:

```
zeros_array = np.zeros((3,3)) # 3x3 matrix filled with zeros
ones_array = np.ones((2,2)) # 2x2 matrix filled with ones
random_array = np.random.rand(2,3) # 2x3 matrix with random numbers

print(zeros_array, "\n")
print(ones_array, "\n")
print(random_array)
```

2. Indexing & Slicing

Accessing elements:

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[0]) # First element (10)
print(arr[-1]) # Last element (50)
```

Indexing in a 2D array:

```
matrix = np.array([[10, 20, 30], [40, 50, 60]])
print(matrix[0, 1]) # Element in row 0, column 1 (20)
```

Slicing arrays:

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[1:4]) # [20, 30, 40]
```

```
print(arr[:3])    # First 3 elements [10, 20, 30]
print(arr[-3:])   # Last 3 elements [30, 40, 50]
```

Slicing in a 2D array:

```
matrix = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])
print(matrix[:2, 1:]) # First two rows, last two columns
```

- **FA1 (5 minutes) – 8%:** Short quiz or hands-on practice on array manipulations

4. Discussion & Application (15 minutes) – 25%

- **SO3 (10 minutes) – 17%:** Matrix operations, broadcasting, and reshaping

Basic Mathematical Operations

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b) # Element-wise addition [5, 7, 9]
print(a * b) # Element-wise multiplication [4, 10, 18]
print(a ** 2) # Squaring each element [1, 4, 9]
```

Matrix multiplication:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

result = np.dot(A, B) # Matrix multiplication
print(result)
```

Broadcasting in NumPy

Broadcasting allows NumPy to perform element-wise operations on arrays of different shapes without explicitly replicating them.

Example 1: Adding a scalar to an array

```
import numpy as np

arr = np.array([1, 2, 3])
result = arr + 5 # Each element gets added to 5
print(result) # Output: [6, 7, 8]
```

Example 2: Broadcasting a smaller array

```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([1, 2, 3]) # 1D array

result = A + B # Broadcasting happens here
print(result)
```

Output:

```
[[2 4 6]
 [5 7 9]]
```

- Here, NumPy automatically expands B to match the shape of A.

Example 3: Column Broadcasting

```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[1], [2]]) # 2D column vector

result = A + B
print(result)
```

Output:

```
[[2 3 4]
 [6 7 8]]
```

- NumPy stretches `B` across columns.

Reshaping Arrays

Reshaping changes the structure of an array without altering its data.

Example 1: Converting 1D to 2D

```
arr = np.array([1, 2, 3, 4, 5, 6])
reshaped = arr.reshape(2, 3) # 2 rows, 3 columns
print(reshaped)
```

Output:

```
[[1 2 3]
 [4 5 6]]
```

Example 2: Flattening a 2D array

```
matrix = np.array([[1, 2, 3], [4, 5, 6]])
flattened = matrix.flatten()
print(flattened)
```

Output:

```
[1 2 3 4 5 6]
```

Example 3: Using -1 for automatic reshaping

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
reshaped = arr.reshape(2, -1) # NumPy automatically determines
columns
print(reshaped)
```

Output:

```
[[1 2 3 4]
 [5 6 7 8]]
```

- **SO2 (5 minutes) – 8%:** Real-world applications of NumPy in data science and machine learning

5. Conclusion & Summary (5 minutes) – 8%

- Recap key points
- Answer student questions

Numpy Cheat Sheet

Functions and Operations

Array Creation <ul style="list-style-type: none"> • <code>array()</code> • <code>asarray()</code> • <code>zeros()</code> • <code>ones()</code> • <code>empty()</code> • <code>full()</code> • <code>arange()</code> • <code>linspace()</code> • <code>logspace()</code> • <code>meshgrid()</code> • <code>fromfunction()</code> • <code>fromiter()</code> • <code>frombuffer()</code> 	Statistical Functions <ul style="list-style-type: none"> • <code>mean()</code> • <code>median()</code> • <code>std()</code> • <code>var()</code> • <code>min()</code> • <code>max()</code> • <code>percentile()</code> • <code>quantile()</code> • <code>sum()</code> • <code>prod()</code> • <code>cumsum()</code> • <code>cumprod()</code> 	Linear Algebra <ul style="list-style-type: none"> • <code>dot()</code> • <code>vdot()</code> • <code>inner()</code> • <code>outer()</code> • <code>matmul()</code> • <code>tensordot()</code> • <code>einsum()</code> • <code>linalg.svd()</code> • <code>linalg.inv()</code> • <code>linalg.pinv()</code> • <code>linalg.qr()</code> • <code>linalg.eig()</code> • <code>linalg.eigh()</code> • <code>linalg.solve()</code> • <code>linalg.det()</code> 	Set Operations <ul style="list-style-type: none"> • <code>unique()</code> • <code>intersect1d()</code> • <code>union1d()</code> • <code>setdiff1d()</code> • <code>setxor1d()</code> • <code>in1d()</code> • <code>isin()</code>
Array Manipulation <ul style="list-style-type: none"> • <code>reshape()</code> • <code>flatten()</code> • <code>ravel()</code> • <code>transpose()</code> • <code>swapaxes()</code> • <code>concatenate()</code> • <code>stack()</code> • <code>vstack()</code> • <code>hstack()</code> • <code>split()</code> • <code>hsplit()</code> • <code>vsplit()</code> • <code>append()</code> • <code>insert()</code> • <code>delete()</code> • <code>repeat()</code> • <code>tile()</code> 	Random Functions <ul style="list-style-type: none"> • <code>random()</code> • <code>rand()</code> • <code>randn()</code> • <code>randint()</code> • <code>random_sample()</code> • <code>choice()</code> • <code>shuffle()</code> • <code>permutation()</code> • <code>beta()</code> • <code>binomial()</code> • <code>chisquare()</code> • <code>gamma()</code> • <code>geometric()</code> • <code>hypergeometric()</code> • <code>logistic()</code> • <code>normal()</code> • <code>poisson()</code> • <code>uniform()</code> 	Trigonometric Functions <ul style="list-style-type: none"> • <code>sin()</code> • <code>cos()</code> • <code>tan()</code> • <code>arcsin()</code> • <code>arccos()</code> • <code>arctan()</code> • <code>arctan2()</code> • <code>sinh()</code> • <code>cosh()</code> • <code>tanh()</code> • <code>arcsinh()</code> • <code>arccosh()</code> • <code>arctanh()</code> 	Bitwise Operations <ul style="list-style-type: none"> • <code>bitwise_and()</code> • <code>bitwise_or()</code> • <code>bitwise_xor()</code> • <code>invert()</code> • <code>left_shift()</code> • <code>right_shift()</code>
Array Creation <ul style="list-style-type: none"> • <code>add()</code> • <code>subtract()</code> • <code>multiply()</code> • <code>divide()</code> • <code>mod()</code> • <code>power()</code> • <code>dot()</code> • <code>matmul()</code> • <code>outer()</code> • <code>inner()</code> • <code>tensordot()</code> 	Mathematical Functions <ul style="list-style-type: none"> • <code>sqrt()</code> • <code>cbt()</code> • <code>square()</code> • <code>absolute()</code> • <code>fabs()</code> • <code>sign()</code> • <code>heaviside()</code> • <code>maximum()</code> • <code>minimum()</code> • <code>clip()</code> • <code>fmod()</code> 	Exponential and Logarithmic Functions <ul style="list-style-type: none"> • <code>exp()</code> • <code>expm1()</code> • <code>exp2()</code> • <code>log()</code> • <code>log10()</code> • <code>log2()</code> • <code>log1p()</code> 	Indexing & Slicing <ul style="list-style-type: none"> • <code>where()</code> • <code>nonzero()</code> • <code>argmax()</code> • <code>argmin()</code> • <code>argsort()</code> • <code>searchsorted()</code>
		Fourier Transform <ul style="list-style-type: none"> • <code>fft()</code> • <code>ifft()</code> • <code>fft2()</code> • <code>ifft2()</code> • <code>fftn()</code> • <code>ifftn()</code> 	Miscellaneous <ul style="list-style-type: none"> • <code>copy()</code> • <code>save()</code> • <code>load()</code> • <code>savez()</code> • <code>loadtxt()</code> • <code>savetxt()</code> • <code>genfromtxt()</code> • <code>corrcoef()</code> • <code>cov()</code>



Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge		1				
B. Conceptual Knowledge			2			
C. Procedural Knowledge					3	2,3

Reference:

- 1. Python Data Science Handbook** – by Jake VanderPlas
- 2. Guide to NumPy** – by Travis Oliphant
- 3. Numerical Python: A Practical Techniques Approach for Industry** – by Robert Johansson.

PPT Link :

<https://docs.google.com/presentation/d/1ChNtNZnspqgybRzhkOJZo3CwXd74daMI/edit?usp=sharing&ouid=113456501758187936448&rtpof=true&sd=true>

Verified by Subject Expert:



Course In-charge



Approved by HoD

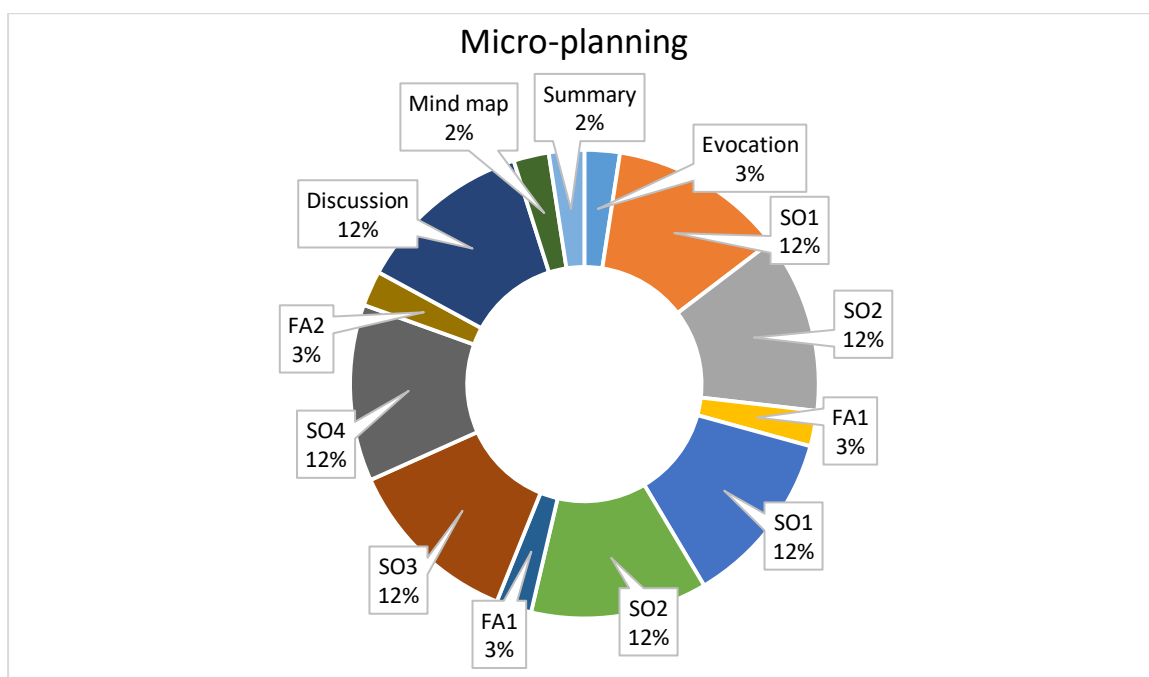


Lesson Plan

Programme	M.Sc. Computer Science
Semester	III
Course Title	Research Methodology
Code	21PCSC34
Hours	4
Total Hours	60
Credits	4
Max Marks	60
Unit & Title	Unit I & Types of Research
Name of the Faculty	Dr. A. Vithya Vijayalakshmi
T-L tools	Mindmap, Powerpoint

Pre-requisite Knowledge : Basic understanding of research

Micro- Planning : 60 minutes



1. Topics for Learning through Evocation:

Brief explanation about research and its role in daily decision-making. Ask the students to share their knowledge about different types of research.

2. Topic Introduction:

2.1: General Objective:

- To understand different types of research.
- To learn their applications in various fields.

2.2: Specific Outcomes:

- SO1: To identify and differentiate various types of research.
- SO2: To understand their significance and applicability.
- SO3: To analyze the advantages and disadvantages of each type of research.
- SO4: To apply different research methods in real-world scenarios.

First Phase:

SO1 (10 minutes):

- Explain what research is and why it is essential.

SO2 (10 minutes):

- Discuss different classifications of research
 - Basic vs. Applied Research
 - Qualitative vs. Quantitative Research

Second Phase:

SO3 (10 minutes):

- Clarify the distinctions between Descriptive, Exploratory, Experimental, and Analytical Research.

SO4 (10 minutes):

- Explore examples of each type in business and academia.

Mind Map (2 minutes):

- Mindmap focusing on the key words in types of research.

Summary (2 minutes):

- Summarize the key points about research types through a group discussion.

2.3: Taxonomy of objectives:

Taxonomy of objectives						
Knowledge Dimension	The Cognitive Process Dimension					
	Remember	Understand	Apply	Analyze	Evaluate	Create
A. Factual Knowledge	1	1				
B. Conceptual Knowledge		1	2			
C. Procedural Knowledge			2	2,3		
D. Meta Cognitive Knowledge					2,3	2,3

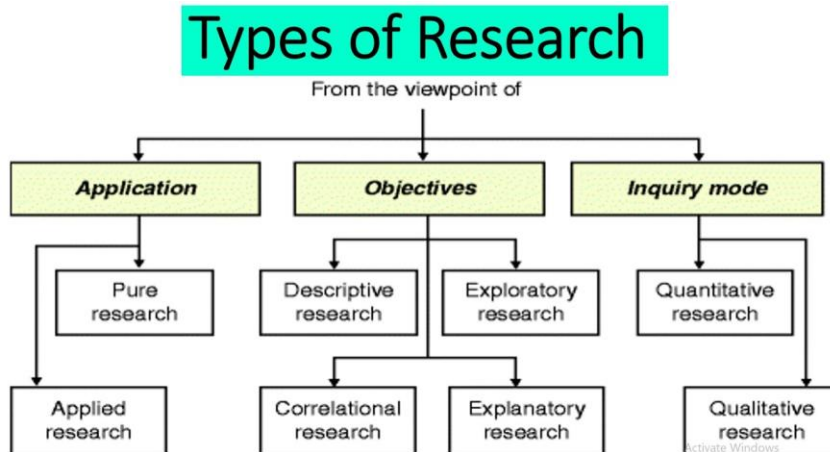
2.4: Key Words:

Basic Research, Applied Research, Qualitative Research, Quantitative Research, Descriptive Research, Exploratory Research, Experimental Research, Analytical Research

PPT Link:

https://docs.google.com/presentation/d/1faGPmVerLSKHU_3uVCd81pxKaxk3Htx7XIV3gGHxhgc/edit?usp=sharing

2.5: Key Diagram:



3. Discussion:

Students are asked to discuss different research types and their real-life applications. They are encouraged to frame a case study on applying research methods in business decision-making.

4. Mind Map:



5. Summary:

Research plays a crucial role in various fields, helping in data-driven decision-making. It can be broadly classified into different types based on purpose, method, and application. Each type has its unique significance in academic and business research.

6. Assessment:

Formative Assessment 1 (FA1):

- Students are asked to differentiate between Qualitative and Quantitative research with examples.

Formative Assessment 2 (FA2):

- A quick review on the advantages and limitations of different types of research.

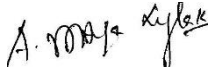
7. FAQs (MSQs/Descriptive Questions):

1. What are the different types of research? Explain with examples.
2. Compare and contrast Qualitative and Quantitative research.
3. Differentiate between Basic and Applied research.

8. References:

1. Kothari, C.R. Research Methodology: Methods and Techniques. New Delhi, New Age International, 2nd Edition, 2012.
2. Janathan Anderson, Berry H. Durston and Millicent Poole. Thesis and Assignment Writing. Wiley Eastern Limited, 1992.

Verified by Subject Expert:



Course In-Charge



Approved by HoD

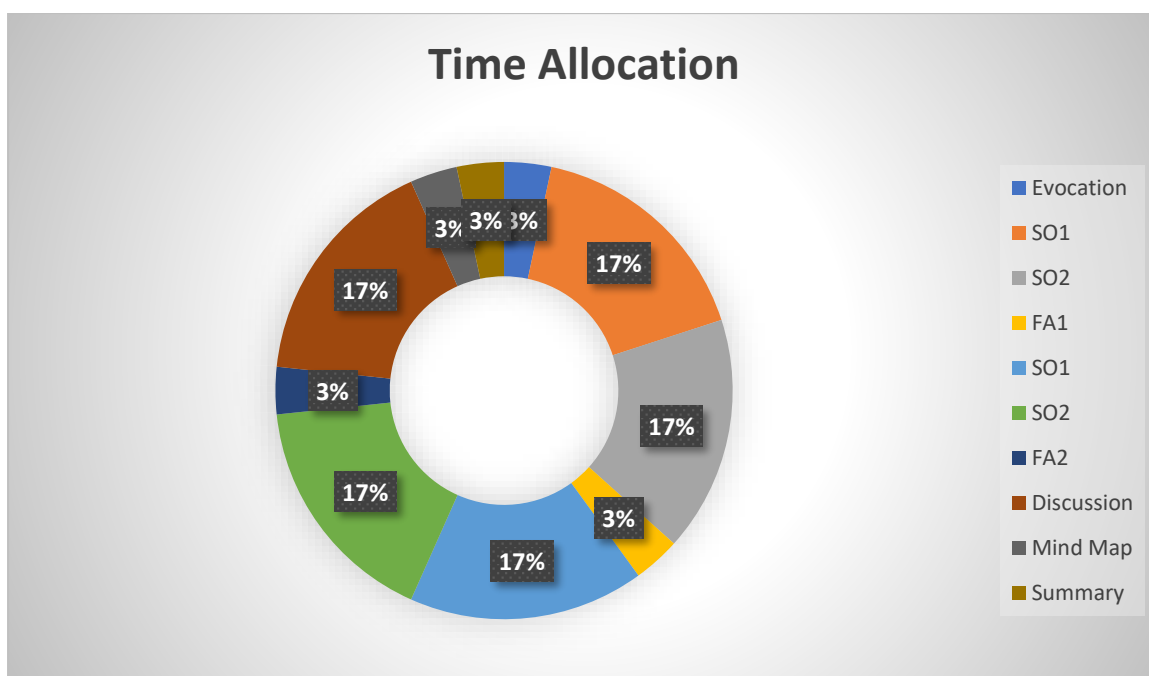


Lesson Plan

Programme	M.Sc Computer Science
Semester	III
Course Title	Object-Oriented Software Engineering
Code	21PCSE32
Hours per Week	4
Total Hours	60
Credits	4
Maximum Marks	60
Unit & Title	Unit II: CRC Approach
Name of Faculty	A.Jenitta Jebamalar
T-L Tools	Mind Maps, PowerPoint Presentation, Group Activity, Case Study Discussion

Pre-requisite Knowledge: Basics of Object-Oriented Concepts, Classes, and Objects

Micro-Planning: 60 minutes



1.Topics for Learning through Evocation:

A brief introduction to the CRC (Class-Responsibility-Collaborator) approach and its role in object-oriented design. Engage students by asking them to share their understanding of object-oriented modeling.

2.Topic Introduction:

2.1.General Objective:

- To comprehend the CRC approach and its application in identifying classes and their responsibilities.
- To apply the CRC approach in designing object-oriented systems.

2.2.Specific Outcomes:

- To identify and define classes using the CRC approach (SO1).

- To explain the collaboration between classes in a system (SO2).

First Phase:

- **SO1 (10 minutes):** Introduce the CRC approach, explaining its components—Class, Responsibility, and Collaborator.
- **SO2 (10 minutes):** Discuss how the CRC approach aids in visualizing object interactions and responsibilities.

Second Phase:

- **SO1 (10 minutes):** Demonstrate the CRC card method through an example, highlighting how to document classes, responsibilities, and collaborators.
- **SO2 (10 minutes):** Analyze a case study to identify classes, responsibilities, and collaborations, illustrating practical application.

Mind Map (2 minutes):

Create a mind map showing the relationship between classes, their responsibilities, and collaborators in a sample object-oriented system.

Summary (2 minutes):

Summarize the CRC approach's benefits in organizing object-oriented systems and improving design clarity and collaboration.

2.3.Taxonomy of Objectives:

Taxonomy of Objectives						
Knowledge Dimension	Cognitive Process Dimension					
	Remember	Understand	Apply	Analyse	Evaluate	Create
A. Factual Knowledge	1					
B. Conceptual Know		1				
C. Procedural Knowledge			1	1		
D. Meta Cognitive Knowledge					1	

2.4 Keywords:

CRC Approach, Class, Responsibility, Collaborator, Object-Oriented Design

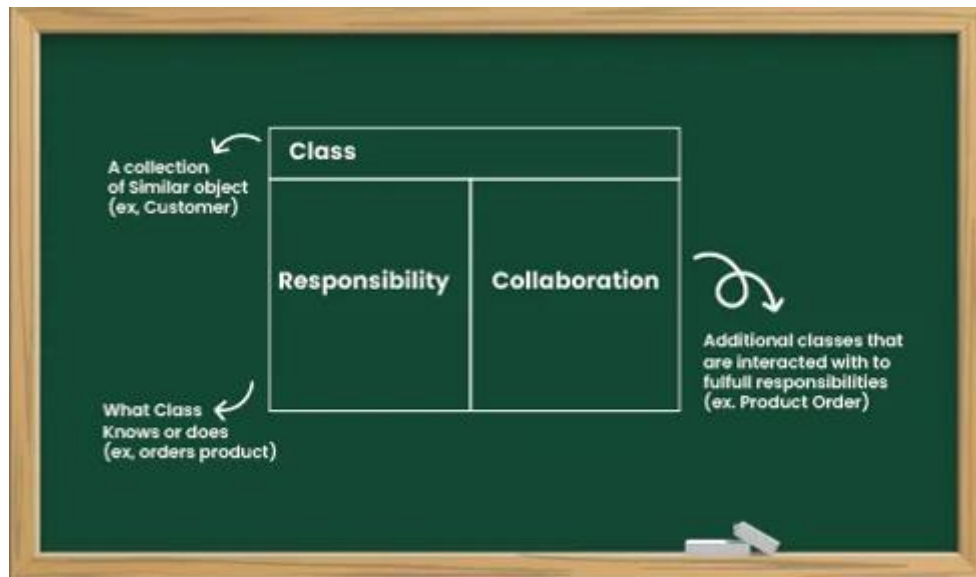
2.5 Powerpoint Presentation:

<https://gamma.app/docs/CRC-Approachpptx-d8jfvmskejpmrod>

3.Discussion:

Students will discuss the advantages of using the CRC approach and how it compares to other object-oriented design methods. They will frame a case study based on a real-world problem using the CRC method.

4. Mind Map



5. Summary

The CRC (Class-Responsibility-Collaborator) Approach is a technique in Object-Oriented Software Engineering used to identify and design classes in a system. It involves creating CRC cards, where each card represents a class, its responsibilities (what it does), and its collaborators (other classes it interacts with). This approach helps in organizing system design, improving clarity, and ensuring better collaboration between objects.

6. Assessment

- **FA1 (2 minutes):** Quick explanation of the components of a CRC card.
- **FA2 (2 minutes):** Review the practical application of CRC in a given scenario.

7. FAQ's / Descriptive Questions:

1. What is the CRC approach? Explain its components.
2. Describe how the CRC approach is used to design an object-oriented system.

8. References:

- Bernd Bruegge, Allen H. Dutoit .*Object-Oriented Software Engineering Using UML, Patterns, and Java*, 3rd Edition ,Pearson.
- Craig Larman. *Applying UML and Patterns*. Pearson Education, 3rd edition 2005.

9. Verified by Subject Expert:

A. Leelitha Lebamalar

Course In-Charge

R. Jayaram

Approved by HoD

