ANDHRA KESARI UNIVERSITY ::ONGOLE

Programme: BCA Honours Data Science (Major)
(w.e.f. Academic Year 2025–26)

COURSESTRUCTURE

Year	semester	Course Code	Title of the Course	No.of Hours	Credits
I	I	1	Computer Fundamentals and Office Automation	3	3
			Computer Fundamentals and Office Automation Lab	2	1
		2	Problem Solving Using C	3	3
			Problem Solving Using C Lab	2	1
	II	3	Python Programming and Data Structures	3	3
			Python programming and Data Structures lab	2	1
		4	Statistical Foundations for Data Science	3	3
			Statistical Foundations for Data Science lab	2	1
II	III	5	Database Management Systems	3	3
			Database Management Systems Lab	2	1
		6	Data Science with R	3	3
			Data Science With R lab	2	1
		7	Object oriented programming using Java	3	3
			Object oriented programming using Java Lab	2	1
	IV	8	Data Mining & Data warehousing	3	3
			Data Mining & Data warehousing Lab	2	1
		9	Exploratory Data Analysis and Visualization	3	3
			Exploratory Data Analysis and Visualization lab	2	1
		10	Software Engineering	3	3
			Software Engineering Lab	2	1
	V	11	Business Intelligence Tools	3	3
			Business Intelligence Tools Lab	2	1

SEA: Skill Elective Track A (Application Development)					
III	V- Track A	12	SEA1: Web Design & Development Fundamentals	3	3
			SEAP1: Web Design & Development Fundamentals Lab	2	1
		13	SEA2: Web Application Development using PHP & MySQL	3	3
			SEAP2: Web Application Development using 2 1 PHP & MySQL Lab	2	1
	V- Track A	14	SEA3: Mobile Application Development	3	3
			SEAP3: Mobile Application Development lab	2	1
		15	SEA4: MERN Stack	3	3
			SEAP4: MERN Stack Lab	2	1
	SEB: Skill Elective Track B (AIML & Data Engineering)				
	V- Track B	12	SEB1: Artificial Intelligence	3	3
			SEBP1: Artificial Intelligence Lab	2	1
		13	SEB2: Big Data Technologies	3	3
			SEBP2: Big Data Technologies lab	2	1
	V- Track B	14	SEB3: Machine Learning	3	3
			SEBP3: Machine Learning Lab	2	1
		15	SEB4: Cloud computing for Data Science	3	3
			SEBP4: Cloud computing for Data Science Lab	2	1

C5P: Database Management Systems Lab

Experiment 1 : Database: Inventory Management

Table 1: Products

Structure:

Column Name	Data Type	Constraints
product_id	INT	PRIMARY KEY
product_name	VARCHAR(50)	NOT NULL
price	DECIMAL(10,2)	CHECK(price > 0)
stock_qty	INT	CHECK(stock_qty >= 0)

Sample Data:

product_id	product_name	price	stock_qty
1	Pen	10.00	100
2	Notebook	50.00	200
3	Stapler	120.00	50
4	Marker	25.00	80
5	File Folder	60.00	150

Table 2: Suppliers

Structure:

Column Name	Data Type	Constraints
supplier_id	INT	PRIMARY KEY
supplier_name	VARCHAR(50)	NOT NULL
contact_no	VARCHAR(20)	UNIQUE
product_id	INT	FOREIGN KEY REFERENCES Products(product_id)

Sample Data:

supplier_id	supplier_name	contact_no	product_id
101	StationeryMart	9876543210	1

102	PaperWorld	9876500000	2
103	OfficeSupplies	9876512345	3
104	MarkerHub	9876522222	4
105	FileDepot	9876533333	5

Section A: DDL (Data Definition Language)

1. Create a database called InventoryDB.
2. Create a table Products and table Suppliers with the specified columns and constraints:

Section B: DML (Data Manipulation Language)

4. Insert at least 5 rows into the Products table.
5. Insert at least 5 rows into the Suppliers table.
6. Update the stock quantity of product 'Pen' to 120.
7. Delete a supplier with a specific supplier_id.
8. Write a query to rename 'Notebook' to 'NoteBook A4'

Section C: DQL (SELECT Queries)

9. Display all records from the Products table.
10. Display only product_name and price of all products.
11. List all products that have a stock quantity less than 100.
12. Show all products between 20 and 100 price range.
13. Find all suppliers whose contact number starts with '98765'.
14. Find the average price of products.
15. Display the total number of products in the inventory.
16. Show the maximum and minimum stock quantities.
17. Count how many suppliers supply each product.
18. Show all products where price > 50 AND stock_qty > 100.
19. Show all products where price < 20 OR stock_qty < 80.
20. Display suppliers whose supplier_name contains the word 'Mart'
21. List all suppliers along with the product they supply (use INNER JOIN).
22. Display suppliers whose name starts with 'S'.
23. Find products whose name has exactly 5 characters
24. Find suppliers who supply products costing more than 100.

Experiment 2 : ONLINE BOOKSTORE DB

An online book store wants to implement a BOOKSTORE DB for managing their online transactions by using the following tables.

Authors Table

Column Name	Data Type	Constraints
author_id	INTEGER	PRIMARY KEY

first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
nationality	VARCHAR	NULL allowed

Books Table

Column Name	Data Type	Constraints
book_id	INTEGER	PRIMARY KEY
Title	VARCHAR	NOT NULL
author_id	INTEGER	FOREIGN KEY REFERENCES Authors
publication_year	INTEGER	
Price	DECIMAL	

Customers Table

Column Name	Data Type	Constraints
customer_id	INTEGER	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
Email	VARCHAR	UNIQUE, NOT NULL
Address	VARCHAR	NOT NULL

Orders Table

Column Name	Data Type	Constraints
order_id	INTEGER	PRIMARY KEY
customer_id	INTEGER	FOREIGN KEY REFERENCES Customers
book_id	INTEGER	FOREIGN KEY REFERENCES Books
order_date	DATE	NOT NULL
quantity	INTEGER	NOT NULL

SAMPLE DATA SET for BOOKSTORE DB

Authors Table

author_id	first_name	last_name	nationality
1	Jane	Austen	British
2	George	Orwell	British
3	Gabriel	Garcia Marquez	Colombian

4	Toni	Morrison	American
5	Mark	Twain	American
6	Harper	Lee	American
7	Fyodor	Dostoevsky	Russian

Books Table

book_id	Title	author_id	publication_year	price
101	Pride and Prejudice	1	1813	12.99
102	1984	2	1949	9.50
103	One Hundred Years of Solitude	3	1967	15.00
104	Beloved	4	1987	11.25
105	Animal Farm	2	1945	8.75
106	Adventures of Huckleberry Finn	5	1884	10.50
107	To Kill a Mockingbird	6	1960	14.00

Customers Table

customer_id	first_name	last_name	Email	address
201	Alice	Smith	alice.s@example.com	12 Oak St, London
202	Bob	Johnson	bob.j@example.com	45 Pine Ave, Oxford
203	Charlie	Brown	charlie.b@example.com	78 Maple Rd, Bristol
204	Diana	Prince	diana.p@example.com	34 Queen St, York
205	Edward	Norton	edward.n@example.com	22 River Ln, Leeds
206	Fiona	Hall	fiona.h@example.com	56 Lake Dr, Bath
207	Greg	Miller	greg.m@example.com	89 Park Ave, Glasgow

Orders Table

order_id	customer_id	book_id	order_date	Quantity
301	201	101	2025-07-20	1
302	202	102	2025-07-21	2
303	201	105	2025-07-22	1
304	203	103	2025-07-23	1
305	204	106	2025-07-24	1
306	205	107	2025-07-25	3
307	206	104	2025-07-26	2

Section A: DDL (Schema Design & Constraints)

1. Write SQL statements to create all 4 tables (Authors, Books, Customers, Orders) with:
    - Primary Keys
    - Foreign Keys
    - Appropriate data types
    - NOT NULL constraints where necessary.
2. Alter the Books table to add a constraint that price must be greater than 0.
3. Add a new column phone_number to the Customers table (VARCHAR(15)) and ensure it is unique.
4. Drop the phone_number column from the Customers table.

Section B: DML (Data Manipulation)

5. Insert at least 7 records for each table (use sample dataset above).
6. Update the price of the book titled *Animal Farm* by increasing it by 10%.
7. Delete all orders made before 2025-07-21.
8. Change the nationality of Gabriel Garcia Marquez to "Latino-American".

Section C: SELECT Queries (Data Querying)

9. List all books published between 1900 and 2000.
10. Find all customers whose email contains "example.com".
11. Retrieve books whose price is between 10 and 15 and published before 1950.
12. Show authors who are either 'British' or 'American'.
13. Find books that have a price less than 10 or are published after 1980.
14. Display all orders placed after 2025-07-22.
15. List all books written by author with author_id = 2.
16. Find customers whose last name starts with B.
17. Show all books with a price NOT between 9 and 13.
18. Display books whose publication_year is in (1813, 1945, 1987).
19. Find authors whose nationality is NOT 'British'.
20. List customers whose address contains the word Park.
21. Show all books sorted by price in descending order.
22. List authors in alphabetical order by last_name.
23. Display orders sorted by order_date (latest first).

Use of Date Functions

24. Show all orders placed in July 2025.
25. Show all orders with an estimated delivery date (5 days after order date).
26. Show customers who placed an order on a weekend.
27. Calculate how many days have passed since the last order was placed.

Aggregate Functions (COUNT, SUM, AVG, MIN, MAX)

28. Count the total number of books in the database.
29. Find the average price of all books.
30. Show the highest-priced book.

31. Count how many orders each customer has placed.
32. Calculate the total sales (price × quantity) for each customer.

GROUP BY and HAVING

33. Count how many books are written by each author.
34. Group orders by customer_id and display total quantity ordered.
35. Show customers who have ordered more than 2 books in total (use HAVING).
36. Find the total number of books sold per author (GROUP BY author).

Experiment 3: EMPLOYEE DB

An enterprise wants to automate its employee management process by implementing an Employee Database.The goal is to replace manual record-keeping with a centralized system that stores employee, department, and project details. Use the following table structures and data set to implement Employee DB.

EmployeeDB - Table Structures

1. Departments Table

Column	Type	Constraints
dept_id	INT	PRIMARY KEY
dept_name	VARCHAR	UNIQUE, NOT NULL
location	VARCHAR	NOT NULL

2. Employees Table

Column	Type	Constraints
emp_id	INT	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
email	VARCHAR	UNIQUE, NOT NULL
phone	VARCHAR	CHECK (phone LIKE '--____')
hire_date	DATE	NOT NULL
job_title	VARCHAR	NOT NULL
salary	DECIMAL	CHECK (salary > 0)
dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)
manager_id	INT	FOREIGN KEY REFERENCES Employees(emp_id) (self-referential)

3. Projects Table

Column	Type	Constraints
project_id	INT	PRIMARY KEY

project_name	VARCHAR	NOT NULL
start_date	DATE	NOT NULL
end_date	DATE	NULL
dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)

4. Employee_Project Table (Many-to-Many)

Column	Type	Constraints
emp_id	INT	FOREIGN KEY REFERENCES Employees(emp_id), PRIMARY KEY(emp_id, project_id)
project_id	INT	FOREIGN KEY REFERENCES Projects(project_id)
hours_allocated	INT	CHECK (hours_allocated > 0)

Sample Data Set

Departments Table

dept_id	dept_name	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago
4	Marketing	Boston
5	Operations	Seattle
6	Legal	Washington D.C.
7	Sales	Dallas
8	R&D	Austin
9	Procurement	Denver
10	Customer Care	Miami

2. Employees Table

emp_id	first_name	last_name	Email	phone	hire_date	job_title	salary	dept_id	manager_id
101	Alice	Johnson	alice.j@corp.com	123-456-7890	2020-03-15	HR Manager	75000	1	NULL
102	Bob	Smith	bob.s@corp.com	234-567-8901	2019-05-20	IT Analyst	65000	2	104

103	Charlie	Brown	charlie.b@corp.com	345-678-9012	2021-01-10	Finance Executive	58000	3	106
104	Diana	Prince	diana.p@corp.com	456-789-0123	2018-07-12	IT Manager	90000	2	NULL
105	Ethan	Hunt	ethan.h@corp.com	567-890-1234	2022-02-25	Marketing Lead	62000	4	NULL
106	Fiona	Hall	fiona.h@corp.com	678-901-2345	2017-11-01	Finance Manager	85000	3	NULL
107	Greg	Miles	greg.m@corp.com	789-012-3456	2023-04-15	IT Support	45000	2	104
108	Hannah	White	hannah.w@corp.com	890-123-4567	2021-09-05	HR Executive	50000	1	101
109	Ian	Scott	ian.s@corp.com	901-234-5678	2020-11-20	Operations Analyst	56000	5	NULL
110	Julia	Adams	julia.a@corp.com	012-345-6789	2019-12-18	Legal Advisor	70000	6	NULL

3. Projects Table

project_id	project_name	start_date	end_date	dept_id
201	Payroll System	2023-01-01	NULL	3
202	Website Upgrade	2023-02-10	NULL	2
203	Recruitment Drive	2023-03-05	NULL	1
204	Ad Campaign	2023-05-20	NULL	4
205	New CRM Tool	2023-04-15	NULL	7
206	Compliance Portal	2023-06-10	NULL	6
207	Inventory System	2023-07-01	NULL	5
208	AI Research	2023-08-05	NULL	8
209	Customer Feedback	2023-09-10	NULL	10
210	Procurement System	2023-10-01	NULL	9

4. Employee_Project Table

emp_id	project_id	hours_allocated
102	202	120
104	202	80
103	201	100

106	201	150
101	203	50
105	204	70
107	202	60
109	207	90
110	206	110
108	203	40

Section A: DDL (Schema Creation & Modification)

1. Write SQL statements to create the above tables with the specified constraints
2. Alter the Employees table to add a column bonus DECIMAL(8,2) with default value0.
3. Drop the column bonus from Employees.

Section B: DML (Insert, Update, Delete)

4. Insert at least 10 rows into Departments, Employees, Projects, and Employee_Project.(use the above data set)
5. Try inserting an employee with a negative salary (should fail due to CHECK constraint).
6. Update the salary of the employee with emp_id = 103 by 15%.
7. Delete an employee record who has resigned (choose any emp_id).
8. Increase all employees' salaries in the IT department by 5%.
9. Change the department of an employee to "Research".(should fail due to FK constraint)

Section C: DQL (Select Queries)

10. List all employees and their details.
11. Show all employees in the "HR" department.
12. Find employees with salaries between 50,000 and 80,000.
13. Retrieve employees hired after 2020.
14. Show employees who are in either the IT or Finance department.
15. Find employees whose email ends with "@corp.com".
16. List all employees with salary > 60,000 AND located in "New York".
17. Display employees in descending order of salary.
18. Count the number of employees in each department.
19. Show the average salary of employees department-wise.
20. Display departments where the average salary is greater than 70,000.
21. Find the number of employees in each project.
22. Display departments with more than 3 employees.
23. Show the sum of all salaries department-wise.
24. List all distinct department IDs from the Employees table.

25. Show employee names with the year they were hired.
26. Show employees grouped by the year of hire.
27. List employees hired in the last 90 days.
28. List the no of years of experience of all the employees

Section D: Joins

29. List all employees with their department names (INNER JOIN).
30. Display all departments along with employees, including those departments without employees (LEFT JOIN).
31. Show employees and the projects they are working on (JOIN 3 tables: Employees, Employee_Project, Projects).
32. List projects along with total hours allocated by employees.
33. Write a query to find employees who are working on more than one project.
34. Show all projects handled by the 'Finance' department.

Section E: PL/SQL Programming

1. Write a procedure GetEmpInfo that takes emp_id as input and displays name, salary, and department.
2. Write a PL/SQL block that checks if an employee's salary is above 50,000.
   If yes, print "High Salary" ;Otherwise print "Standard Salary".
3. Write a PL/SQL program to display the top 10 rows in the Emp table based on their job and salary
4. Write a stored procedure GiveBonus that takes department ID and a designation as input, along with a bonus amount, and updates the salary of all employees in that department who have the specified designation by adding the bonus amount to their current salary.
5. Create a trigger to prevent inserting employees with a salary less than 30,000.
6. Create a trigger to avoid any transactions(insert, update, delete) on the EMP table on Saturday & Sunday.

C6: Data Science with R:

Course Objectives

1. Introduce the data science process, lifecycle, and applications in real-world domains.
2. Build proficiency in R programming for data manipulation, exploration, and visualization.
3. Train students in handling structured, unstructured, and time-based data effectively.
4. Familiarize with basic machine learning and statistical modeling using R.
5. Develop awareness of ethical, interpretability, and responsible use of data science.

Course Outcomes

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

1. Explain the Data Science process and perform EDA (Exploratory Data Analysis).
2. Write R programs using variables, functions, loops, and packages for basic analytics.
3. Perform data wrangling, cleaning, and visualization with R libraries (dplyr, tidyr, ggplot2).
4. Build and evaluate basic machine learning models such as regression and clustering.
5. Apply data science techniques to practical case studies.

Unit 1. Introduction to Data Science Process:

Introduction- Definition - Data Science in various fields - Examples - Impact of Data Science - Data Analytics Life Cycle - Data Science Toolkit - Data Scientist - Data Science Team, Exploratory Data Analysis (EDA), Feature Engineering & Data Transformation

Unit 2. Basics of R Programming:

Introduction to R and RStudio, Data Types, Variables, Operators, Control Structures (if, loops, apply), Functions and Packages, Data Input/Output (CSV, Excel, XML, JSON).

Unit 3. Data Handling & Visualization in R:

Data Frames, Lists, Matrices, Data Wrangling with dplyr and tidyr, Handling Missing Data, Working with Date/Time in R. Visualization with ggplot2: grammar of graphics, aesthetics, geometries, scales.

Faceting and layering techniques,Visualizing categorical and numerical data, Customizing and exporting plots

Unit 4. Applications & Case Studies in Data Science:

Simple Linear Regression, Multiple Regression

Model Evaluation Method: Accuracy, Confusion Matrix, ROC.

K-Means Clustering, Text Mining & Word Clouds, Recommender Systems Basics, Ethical Issues in Data Science

Unit 5. Advanced Topics in Data Science with R :

Introduction to Time Series Analysis in R (ARIMA basics)- Concept of time series (trend, seasonality, noise), Time series objects in R (ts, zoo, xts), Plotting and decomposing time series, Stationarity and differencing, Autocorrelation & Partial Autocorrelation (ACF/PACF), AR, MA, ARIMA model basics, Forecasting using forecast package

Creating interactive visualizations with plotly packages-Converting ggplot2 plots to interactive plots

Animations and sliders in plotly

R Shiny: Building interactive web applications-Introduction to Shiny framework, UI and server functions, Reactive expressions and reactivity in Shiny, Input and output widgets (sliders, dropdowns, text), Layouts and dashboard design

Textbooks

1. An Introduction to Statistical Learning with Applications in R, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Springer, 2nd Edition, 2021
2. R for Data Science, Hadley Wickham and Garrett Grolemund, O'Reilly Media, 2017.

Reference Books

1. The Art of R Programming, Norman Matloff,, No Starch Press, 2011.
2. Modern Applied Statistics with S, W.N. Venables & B.D. Ripley, Springer, 2002.
3. Introduction to Data Science: Data Analysis and Prediction Algorithms with R, Rafael A. Irizarry, CRC Press, 2020.
4. Data Science from Scratch: First Principles with Python (for conceptual clarity only), Joel Grus,

Activities:

Outcome: Explain the Data Science process and perform EDA (Exploratory Data Analysis).

Activity: Use a real-world dataset (e.g., Titanic or COVID data) to:

- Outline the steps of the Data Science workflow

- Perform EDA using summary statistics and visualizations (histograms, boxplots, scatterplots)

Evaluation Method: Presentation and checklist (10-point scale):

- Clear explanation of workflow stages
- Quality of EDA insights
- Use of appropriate plots and summaries

Outcome: Write R programs using variables, functions, loops, and packages for basic analytics.

Activity: Write an R script that:

- Reads a CSV file
- Uses if, for, and while loops
- Defines and calls custom functions with arguments and return values

Evaluation Method: Code review and execution test to verify (10-point scale):

- Correctness of the syntax and logic
- Functionality of control structures
- Output accuracy and modularity

Outcome: Perform data wrangling, cleaning, and visualization with R libraries (dplyr, tidyr, ggplot2).

Activity: Clean a messy dataset using:

- dplyr for filtering, selecting, and mutating
- tidyr for reshaping and handling missing values
- Time-based operations (e.g., filling gaps, formatting dates)

Evaluation Method: Before-and-after comparison (10 point score):

- Completeness of cleaning steps
- Use of appropriate functions
- Handling of missing/time data

Outcome: Implement basic machine learning models and evaluate performance using appropriate metrics and visual tools.

Activity: Build a simple classification model (e.g., logistic regression or decision tree) using R:

- Train/test split

- Predict outcomes
- Evaluate using confusion matrix, accuracy, precision, recall

Evaluation Method: Model report and demo (10 point scale):

- Correct implementation of model
- Use of evaluation metrics

C6P: Data Science with R Lab:

List of Practicals:

1. Compute Mean, Median, Mode, Variance, and Standard Deviation
2. Visualize Binomial, Normal, and Poisson Distributions
3. Perform t-test and Chi-Square Test in R
4. Calculate Correlation and Build a Simple Linear Regression Model
5. Conduct Exploratory Data Analysis (EDA) on a Real-World Dataset
6. Apply Feature Engineering: Scaling, Normalization, and Encoding
7. Practice R Programming: Variables, Control Structures, and Functions
8. Read and Write Data from CSV, Excel, JSON, and XML Files
9. Use dplyr and tidyr for Data Wrangling Tasks
10. Handle Missing Data and Detect Outliers
11. Work with Dates and Times in R
12. Visualize Data Using ggplot2 (Bar, Scatter, Histogram, Boxplot)
13. Perform K-Means Clustering and Visualize Clusters
14. Evaluate Models Using Confusion Matrix, Accuracy, and ROC Curve
15. Perform Text Mining and Create a Word Cloud
16. Time Series Forecasting with ARIMA on a real dateset (e.g., monthly airline passengers, stock prices, or temperature data).
17. Create interactive bar, line, and scatter plots using plotly. On a real dataset (e.g., COVID-19 cases, sales data, or student marks).
18. Develop a Shiny app that lets users upload a CSV file.

C7: Object Oriented Programming Using JAVA

Course Objectives

1. Introduce core OOP principles and contrast procedural and object-oriented paradigms within the Java ecosystem.
2. Equip learners with foundational and advanced Java syntax including variables, control statements, arrays, strings, and classes.
3. Enable the use of inheritance, polymorphism, interfaces, and exception handling to create maintainable and reusable code.
4. Train students in concurrent programming and stream-based I/O operations including file management and serialization.
5. Empower learners to design, build, and manage GUI programs using Swing components, layout managers, and event handling techniques.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications.
2. Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively.
3. Construct modular code leveraging interfaces, abstract classes, and package hierarchies.
4. Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction.
5. Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming.

Unit 1. OOPs Concepts and Java Programming:

Introduction to Object-Oriented concepts, procedural and object-oriented programming paradigm Java programming: An Overview of Java, Java Environment, Data types, Variables, constants, scope and life time of variables, operators, type conversion and casting, Accepting Input from

the Keyboard, Reading Input with Java.util.Scanner Class, Displaying Output, Displaying Formatted Output with String.format(), Control Statements

Unit 2. Arrays and OOP Constructs:

Arrays, Command Line Arguments, Strings-String Class Methods
Classes & Objects: Creating Classes, declaring objects, Methods, parameter passing, static fields and methods, Constructors, and 'this' keyword, overloading methods and access Inheritance: Inheritance hierarchies, super and subclasses, member access rules, 'super' keyword, preventing inheritance: final classes and methods, the object class and its methods; Polymorphism: Dynamic binding, method overriding, abstract classes and methods;

Unit 3. Interfaces, Packages & Exception Handling:

Interfaces VS Abstract classes, defining an interface, implement interfaces, accessing implementations through interface references, extending interface;
Packages: Defining, creating and accessing a package, importing packages.
Exception Handling: Benefits of exception handling, the classification of exceptions, exception hierarchy, checked exceptions and unchecked exceptions, usage of try, catch, throw, throws and finally, rethrowing exceptions, exception specification, built in exceptions, creating own exception sub classes.

Unit 4. Multithreading & Stream based I/O:

Differences between multiple processes and multiple threads, thread states, thread life cycle, creating threads, interrupting threads, thread priorities, synchronizing threads, inter thread communication.
Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, The Console class, Serialization

Unit 5. GUI Programming with Swing:

Introduction, MVC architecture, components, containers. Understanding Layout Managers - Flow Layout, Border Layout, Grid Layout, Card Layout, GridBag Layout. Event Handling- The

Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling mouse and keyboard events, Adapter classes.

Text Books:

1. Java The complete reference, Herbert Schildt, 9th edition, McGraw Hill.
2. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press.

Reference Books:

1. Programming with JAVA - A Primer, E Balaguruswamy, 3rd Edition, McGraw Hill
2. Head First Java: A Brain-Friendly Guide , Katty Sierra, Bert Bates, 2nd Edition, O'Reilly

Activities:

Outcome: Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications

Activity: Develop a class hierarchy for a zoo management system using inheritance and polymorphism (e.g., Animal → Mammal → Dog). Implement encapsulation through private fields and public getters/setters.

Evaluation Method: Code review and oral explanation focusing on class relationships, method overriding, and encapsulation practices.

Outcome: Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively

Activity: Create a console-based student grade calculator using loops, conditionals, arrays, and modular methods.

Evaluation Method: Practical test with debugging tasks and output validation across multiple input scenarios.

Outcome: Construct modular code leveraging interfaces, abstract classes, and package hierarchies **Activity:** Design a payment processing system with abstract classes for Payment, interfaces for Taxable, and organize classes into packages (e.g., com.billing, com.tax).

Evaluation Method: Project submission assessed for modularity, interface implementation, abstraction usage, and package structure.

Outcome: Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction

Activity: Build a multithreaded logger that reads input from the console and writes to a file using synchronized threads and buffered streams.

Evaluation Method: Lab demonstration with thread state tracing and file output verification under concurrent input.

Outcome: Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming

Activity: Create a GUI-based quiz application using Swing components (JFrame, JButton, JTextField) with event listeners for mouse clicks and key presses.

Evaluation Method: Live demo and rubric-based assessment of UI responsiveness, event handling accuracy, and layout design.

C7P: Object Oriented Programming Using JAVA lab

List of Experiments

1. Write a Java program to print Fibonacci series.
2. Write a Java program to calculate multiplication of 2 matrices.
3. Write a Java program for sorting a given list of names in ascending order.
4. Create a class Rectangle. The class has attributes length and width. It should have methods that calculate the perimeter and area of the rectangle. It should have readAttributes() method to read length and width from the user.
5. Write a Java program that implements method overloading.
6. Write a Java program to implement various types of inheritance

i. Single ii. Multi-Level iii. Hierarchical iv. Hybrid

7. Write a java program to implement runtime polymorphism.
8. Write a Java program which accepts withdrawal amount from the user and throws an exception In Sufficient Funds when withdrawal amount is more than available amount.

9. Write a Java program to create three threads and that displays good morning, for every one second, hello for every 2 seconds and welcome for every 3 seconds by using extending Thread class.

10. Write a Java program that creates three threads. The first thread displays OOPS, the second thread displays Through and the third thread displays JAVA by using Runnable interface.

11. Write a Java program that displays the number of characters, lines and words in a text file.

12. Implement a Java program for handling mouse events when the mouse entered, exited, clicked, pressed, released, dragged and moved in the client area.

13. Implement a Java program for handling key events when the key board is pressed, released, typed.

14. Write a Java swing program that reads two numbers from two separate text fields and displays the sum of two numbers in the third text field when the button add is pressed.

15. Write a Java program to design student registration form using Swing Controls. The form must have the following fields and button SAVE

Form Fields are: Name, RNO, Mailid, Gender, Branch, Address.

Semester-IV

C8: Data Mining and Data Warehousing

Course Objectives:

- Provide an understanding of data warehousing concepts, architecture, and OLAP operations for effective storage, modeling, and analysis.
- Develop knowledge of data mining fundamentals, tasks, and preprocessing techniques to prepare data for mining.
- Introduce students to association rule mining algorithms for discovering hidden patterns and relationships in large datasets.
- Enable learners to apply classification techniques (decision trees, Bayesian, nearest neighbor, rule-based) for predictive modeling.

- Equip students with knowledge of clustering paradigms and algorithms (partitioning, hierarchical, density-based, categorical) for data grouping and pattern discovery.

Course Outcomes:

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

1. Explain the architecture, schemas, and OLAP operations of data warehousing and distinguish it from traditional database systems.
2. Apply preprocessing techniques (data cleaning, dimensionality reduction, feature selection, transformation, and similarity measures) to prepare raw data for analysis.
3. Implement various association rule mining algorithms (Apriori, Partition, FP-Growth, etc.) to uncover meaningful relationships within large datasets.
4. Build and evaluate classification models using decision tree algorithms (ID3, C4.5, CART), rule-based classifiers, Bayesian classifiers, and nearest-neighbor methods.
5. Analyze and implement clustering techniques such as K-Means, K-Medoid, DBSCAN, BIRCH, and categorical clustering methods (STIRR, ROCK, CACTUS) for grouping and pattern discovery in different types of datasets.

Unit-1: Data Warehousing:

Introduction to Data Ware House, Differences between Database systems and Data Ware House, Data Ware House characteristics, Data Ware House Architecture and its components, Data Modeling, Schema Design, star and snow-Flake Schema, Fact Constellation, Fact Table, OLAP cube, OLAP Operations.

Unit-2: Data Mining:

What is Data Mining? Data Mining: Definitions, KDD vs Data Mining, Data Mining Tasks, Data Preprocessing- Data Cleaning, Missing Data, Dimensionality Reduction, Feature Subset Selection, Discretization and Binarization, Data Transformation; Measures of similarity and Dissimilarity-Basics.

Issues and Challenges in DM, DM Applications- Case Studies

Unit-3: Association Analysis:

Association Rules: What is an Association Rule?, Methods to Discover Association Rules, A Priori Algorithm, Partition Algorithm, Pincer-Search Algorithm, Dynamic Itemset Counting Algorithms, FP-Tree Growth Algorithm, Generalized Association Rule, Association Rules with Item Constraints

Unit-4: Classification:

Introduction to Classification and Predictive Modeling, Decision Tree Concepts:, Basic Terminology (Root, Node, Leaf), Tree Construction Principles (Splitting, Pruning), Information Gain, Gini Index (Basic conceptual), Decision Tree Algorithms: CART, Overview of Other Classifiers: Naïve Bayes, k-Nearest Neighbor (k-NN), Rule-Based Classifiers (concepts)

Introduction to Ensemble Learning (Conceptual): What is Ensemble Learning?, Motivation: Improving Accuracy through Combination, Bagging and Random Forest (concepts only), Boosting, Voting Classifiers (concept only), Model Evaluation: Confusion Matrix, Accuracy, Precision, Recall

Unit-5: Clustering Techniques:

Clustering Paradigms, Partitioning Algorithms (K-Means), k-Medoid Algorithms, Hierarchical Clustering: DBSCAN, BIRCH, Categorical Clustering Algorithms: STIRR, ROCK, CACTUS

Textbooks:

1. Data Mining Techniques, Arun K Pujari, 3rd Edition, Universities Press
2. Data Mining: Concepts and Techniques, Jiawei Han, Micheline Kamber, Jian Pei, 3rd Edition, Morgan Kaufmann Publishers

Reference Books:

1. K.P. Soman , Shyam Diwakar, V.Ajay ,2006, Insight into Data Mining Theory and Practice, Prentice Hall of India Pvt. Ltd - New Delhi.

2. Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, 2nd edition,

Activities:

Outcome: Explain the architecture, schemas, and OLAP operations of data warehousing and distinguish it from traditional database systems.

Activity: Students will design a **conceptual schema** for a data warehouse using **star or snowflake schema** for a given business case (e.g., sales, hospital, or university records). They will also **demonstrate OLAP operations** (roll-up, drill-down, slice, dice) on a sample dataset using a spreadsheet or OLAP tool.

Evaluation Method: Assessment will be based on correctness and completeness of schema design, appropriateness of fact/dimension tables, and accuracy of OLAP operation outputs. A short viva/quiz will be used to test conceptual understanding.

Outcome 2: Apply preprocessing techniques (data cleaning, dimensionality reduction, feature selection, transformation, and similarity measures) to prepare raw data for analysis.

Activity: Students will be given a **real-world noisy dataset** (with missing values, outliers, redundant features). They will perform:

- Data cleaning (handling missing values, outliers)
- Dimensionality reduction (e.g., PCA)
- Feature selection (e.g., filter/wrapper methods)
- Similarity/dissimilarity measure calculations.

Evaluation Method: Evaluation will consider correctness of preprocessing steps, justification of chosen methods, and clarity of intermediate outputs. Students will submit a short report with before/after comparisons.

Outcome 3: Implement various association rule mining algorithms (Apriori, Partition, FP-Growth, etc.) to uncover meaningful relationships within large datasets.

Activity: Using a **transaction dataset** (e.g., market basket data), students will implement Apriori/FP-Growth in **Python (mlxtend library)** or **Weka**. They will generate frequent itemsets, derive association rules, and interpret them under given **support, confidence, and lift thresholds.**

Evaluation Method: Submissions will be graded on correctness of generated rules, clarity of code/parameters, quality of interpretation, and ability to link discovered rules to practical business insights.

Outcome 4: Build and evaluate classification models using decision tree algorithms (ID3, C4.5, CART), rule-based classifiers, Bayesian classifiers, and nearest-neighbor methods.

Activity: Students will implement at least **two classification algorithms** (e.g., Decision Tree + Naïve Bayes or KNN) on a dataset like **Iris, Titanic, or Student Performance.** They will evaluate models using **confusion matrix, accuracy, precision, recall, and F1-score** and compare results.

Evaluation Method: Assessment will consider correctness of implementation, clarity in performance comparison, and visualization of decision trees/rules. Students must explain why certain models performed better.

Outcome: Analyze and implement clustering techniques such as K-Means, K-Medoid, DBSCAN, BIRCH, and categorical clustering methods (STIRR, ROCK, CACTUS).

Activity: Students will implement at least two clustering algorithms (e.g., K-Means and DBSCAN or BIRCH) on real datasets (e.g., customer segmentation, text data, student groups). They will compare clusters using Silhouette Score, Davies-Bouldin Index, and visualize results using scatter plots/dendrograms.

Evaluation Method: Evaluation will be based on accuracy of clustering implementation, choice of parameters (e.g., k in K-Means, eps in DBSCAN), visualization quality, and clarity in interpretation of clusters.

C8P: Data Mining and Data Warehousing Lab

List of Experiments:

Recommended datasets: **weather.arff, iris.arff, supermarket.arff, vote.arff, contact-lenses.arff**, or custom CSV datasets.

1. Load datasets in WEKA and explore data formats (ARFF/CSV)
2. Perform data cleaning and handle missing values using filters
3. Apply normalization and discretization on numeric attributes
4. Reduce data using attribute selection and PCA
5. Summarize and visualize data using statistical tools and class-wise comparison in WEKA.
6. Generate association rules using the Apriori algorithm
7. Apply multilevel association rule mining using hierarchical attributes
8. Apply K-means clustering and interpret the cluster outputs.
9. Perform hierarchical clustering and visualize results using dendrograms.
10. Apply Expectation-Maximization (EM) clustering and analyze cluster summaries.
11. Build a decision tree classifier using J48 and evaluate its performance.
12. Perform Naive Bayes classification and compare with decision tree results.
13. Apply rule-based classification using PART or JRip algorithms.
14. Compare classifiers using confusion matrix, accuracy, and ROC curves.

C9: Exploratory Data Analysis & Data Visualization

Course Objectives

1. To introduce the concepts, importance, and workflow of exploratory data analysis.
2. To impart practical skills in data preprocessing, cleaning, and manipulation using Python.
3. To equip students with techniques in univariate, bivariate, and multivariate data analysis.
4. To develop proficiency in data visualization and interpretation using standard Python libraries.
5. To enable students to build reports and dashboards to communicate data insights effectively.

Course Outcomes

After completing this course, students will be able to:

1. Explain the importance and process of EDA in the data science pipeline.
2. Perform robust data preprocessing including missing value handling, outlier detection, transformation, and encoding.
3. Manipulate, filter, and reshape datasets using Python libraries like Pandas and NumPy.
4. Conduct data analysis and visualize results using Matplotlib, Seaborn, and Plotly.
5. Design interactive visualizations and dashboards to communicate findings in real-world scenarios.

Unit 1: Introduction to Exploratory Data Analysis (EDA)

Significance and objectives of EDA, Types and scales of data (nominal, ordinal, interval, ratio), Concepts of variability and central tendency, Role of EDA in the data science pipeline, Introduction to data quality and data issues, Differences between EDA, classical, and Bayesian analysis, Basic terminology and concepts

Unit 2: Data Preprocessing and Cleaning

Overview of data preprocessing, Handling missing types of missingness (MCAR, MAR, MNAR), Techniques for missing data imputation (mean, median, mode, forward/backward fill), Detection and treatment of duplicates, Outlier detection methods (Z-score, IQR method), Data transformation: normalization and standardization, Encoding categorical variables: label encoding, one-hot encoding, Data integration and reduction techniques, Importance of preprocessing for machine learning models

Unit 3: Data Manipulation using Python with Pandas and NumPy

NumPy Array Creation: Use `np.array()`, `np.zeros()`, `np.ones()` to create arrays of various dimensions. **Array Manipulation:** Reshape, transpose, flatten arrays for flexible data formats. Element-wise Arithmetic Operations, Statistical Functions on NumPy Arrays

Indexing & Slicing: Access array elements via slicing, masking, and fancy indexing for selective data analysis.

Pandas DataFrames: Series and Data frames, Create from dictionaries, Series, CSV, Excel, and JSON files using Dataframe or reader functions

Data Frame Manipulation: Indexing & Selection - Label-based and Position-based indexing; conditional filtering with boolean masks, Aggregation & Grouping- Group data by columns with `.groupby()`, summarize using `.sum()`, `.mean()`, `.count()`.

Data Reshaping & Pivoting and stack/unstack hierarchical index data.

Merging & Joining Data Frames

Datetime & Categorical Data: Handle and transform temporal data with `pd.to_datetime()`, categorical conversions with `.astype('category')`.

Unit 4: Basic Data Analysis and Visualization

What is visualization, what is it's importance? Types: histograms, bar charts, Boxplots and violin plots for data distribution, Scatter plots and correlation analysis (Pearson, Spearman), Cross-tabulation and contingency tables, Analyzing relationships: correlation vs causation, Visualizing categorical vs numerical data

Unit 5: Advanced Visualization and Reporting

Principles of effective graphical representation, Introduction to visualization libraries: Matplotlib, Seaborn, Plotly, Advanced plots: heatmaps, pair plots, joint plots, Interactive visualizations and dashboards, Geographical data visualization basics, Storytelling with using visualization for communication, Introduction to reporting tools and best practices

Activities

Unit 1: Introduction to EDA

Activity:

Analyze the Titanic dataset to explore data types, central tendency, and variability.

Outcome:

Develop a foundational understanding of data structures and EDA significance.

Evaluation Method:

Lab notebook submission and oral quiz.

Unit 2: Data Preprocessing and Cleaning

Activity:

Clean a health survey dataset: identify/impute missing values, standardize columns, encode categories, remove duplicates/outliers.

Outcome:

Demonstrate proficiency in data cleaning and preprocessing techniques in Python.

Evaluation Method:

Script submission and report on preprocessing steps with before/after summary.

Unit 3: Data Manipulation using Python

Activity:

Manipulate a retail transaction dataset by indexing, filtering, grouping, merging, and handling date/time columns.

Outcome:

Apply multiple Pandas/NumPy techniques for dataset transformation.

Evaluation Method:

Graded Jupyter notebook, peer code review.

Unit 4: Basic Data Analysis and Visualization

Activity:

Explore demographic data with descriptive statistics, create histograms, boxplots, scatter plots, and cross-tabulations.

Outcome:

Visualize and interpret univariate and bivariate data insights.

Evaluation Method:

Visualization notebook and critical analysis write-up.

Unit 5: Advanced Visualization and Reporting

Activity:

- Build a dashboard on sales or air quality data using Matplotlib/Seaborn/Plotly.
- Incorporate interactive and geospatial visualizations.Outcome:
- Develop professional dashboards and communicate insights through visual storytelling.

Evaluation Method:

Final project dashboard evaluation and presentation.

Preferred Textbooks

1. Suresh Kumar Mukhiya, Usman Ahmed, "Hands-On Exploratory Data Analysis with Python", Packt Publishing, 2020.
2. Jake Vander Plas, "Python Data Science Handbook: Essential Tools for Working with Data", O'Reilly, 2017.
3. Catherine Marsh, Jane Elliott, "Exploring Data: An Introduction to Data Analysis for Social Scientists", Wiley, 2008.

References

1. Eric Pimpler, "Data Visualization and Exploration with R", GeoSpatial Training Service, 2017.
2. Claus O. Wilke, "Fundamentals of Data Visualization", O'Reilly, 2019.
3. Matthew O. Ward, Georges Grinstein, Daniel Keim, "Interactive Data Visualization: Foundations, Techniques, and Applications", CRC Press, 2015.

C9P: Exploratory Data Analysis & Data Visualization Lab

Note: This lab has to be executed using interactive computing environments like Jupyter Notebook or Google Colab.

1. Importing Data and Basic Exploration
   - Load datasets from CSV/Excel/JSON
   - Use `.head()`, `.info()`, `.describe()`, `.shape` to examine structure and summary statistics
2. Examining Data Types and Missing Values
   - Identify data types for each column
   - Detect and quantify missing values with `.isnull()` and `.sum()`.
3. Handling Missing Data Techniques
   - Impute missing values using mean, median, mode, forward, and backward fill
   - Drop rows/columns with missing data and compare results
4. Dealing With Duplicates and Outliers

- Find and remove duplicate entries
- Detect outliers using Z-score or IQR methods
- Treat or remove outliers and compare distributions

5. Data Transformation and Scaling
   - Normalize, standardize, log-transform columns
   - Encode categorical variables (Label Encoding, One-Hot Encoding)
6. Indexing, Filtering, and Slicing DataFrames
   - Select specific rows/columns using index and Boolean filters
   - Conditional selections and advanced slicing
7. Aggregating, Grouping, and Pivoting Data
   - Use `.groupby()` for aggregate statistics
   - Create pivot tables for categorical summaries
8. Merging and Concatenating Multiple Datasets
   - Join, merge, concatenate DataFrames
   - Handle merging keys and missing arguments
9. Univariate Visualization Techniques
   - Create histograms, boxplots, violin plots for distributions
   - Visualize frequency and central tendency
10. Bivariate and Multivariate Visualization
    - Display scatter plots, correlation matrices, pair plots
    - Visualize relationships and patterns
11. Time Series Data Exploration
    - Parse and index datetime columns
    - Visualize trends and seasonality
12. Interactive Visualizations and Dashboards
    - Build simple interactive graphs with Plotly
    - Design a dashboard showing multiple aspects of a dataset

C10: SOFTWARE ENGINEERING

Course Objectives :

1. Understand the fundamental principles of software engineering, including software development life cycle models and their practical applications.
2. Analyze and document software requirements through feasibility studies and specification techniques.
3. Apply software design principles and development standards for building reliable and maintainable systems.
4. Evaluate software testing methodologies, including design and execution of test cases for various testing levels.
5. Examine software maintenance strategies, types, and metrics to sustain software performance over time.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.
2. Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.
3. Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.
4. Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.
5. Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Unit 1: Software Engineering Foundations and Requirements Engineering:

Definition of Software engineering, Software life cycle models: Waterfall, prototyping, Evolutionary, Spiral and Agile models. Comparison among development life cycles.

Unit 2: Requirement Analysis and Specification

Feasibility study, Requirements gathering, Functional and Non-functional requirements, Requirements analysis and specification, design of software requirement specification (SRS).

Unit 3: Software Design Principles and Development Practices

Introduction to software design, modularity, cohesion, coupling and layering, functional design, and solution design, use of DFD and Structure chart in software design, UML in software design, user interface design. Software Development Basics, Coding standards, Version Control and Code review techniques.

Unit 4: Software Testing

Fundamentals of testing (verification and validation), White-box, and black-box testing, unit testing, integration testing, system testing, acceptance testing (alpha testing and beta testing), test scenarios and test case design, automation testing and manual testing.

Unit 5: Software Maintenance

Introduction to software maintenance, corrective maintenance, perfective maintenance, adaptive maintenance, preventive maintenance, challenges in software maintenance, metrics related to software maintenance.

Textbooks:

1. Software Engineering: A Practitioner's Approach, Roger Pressman, McGraw Hill, 6th Edition
2. Software Engineering, Sommerville, Adison Wesley, 7th edition.

Reference Books:

1. Fundamentals of Software Engineering, Mall Rajib,PHI, Fifth Edition,
2. Fundamentals of Software Engineering, Hitesh, BPB Publications

Activities

Outcome: Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.

Activity: Create a comparison chart in groups showing key features, pros/cons, and use cases of Waterfall, Spiral, and Agile models. Include a real-world example for each.

Evaluation Method: Use a rubric to assess on a 10-point scale to check:

- Accuracy of model descriptions
- Clarity of comparison
- Relevance of examples
- Presentation skills (if shared in class)

Outcome: Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.

Activity: Analyze a simple system (e.g., online library or food ordering app). Identify 5 functional and 3 non-functional requirements. Draft a mini-SRS document using a template.

Evaluation Method: Checklist-based review on a 10-point scale:

- Correct classification of requirements
- Completeness of SRS sections
- Clarity and formatting
- Use of standard terminology

Outcome: Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.

Activity: Design a basic login system using DFD (Level 0 and Level 1) and a structure chart. Highlight modules and discuss cohesion and coupling in pairs.

Evaluation Method: Peer review and instructor feedback on a 10-point scale:

- Correct use of DFD symbols
- Logical flow and modularity
- Explanation of cohesion/coupling
- Neatness and labelling

Outcome: Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.

Activity: Write simple test cases for a calculator app (e.g., addition, division by zero). Perform manual unit testing and simulate black-box and white-box testing.

Evaluation Method: Observation and worksheet to assess on a 10-point scale:

- Correct identification of test types
- Clear test case steps and expected output
- Execution and result recording
- Understanding of manual vs automated testing

Outcome: Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Activity: Categorize sample maintenance tasks (e.g., fixing a bug, adding a feature) into corrective, adaptive, perfective, or preventive. Discuss sustainability strategies in small groups.

Evaluation Method: Group activity score:

- Correct classification of maintenance types
- Participation in discussion
- Suggestions for sustainability (e.g., documentation, modular design)
- Use of basic metrics (e.g., number of bugs fixed)

C10P: SOFTWARE ENGINEERING

Select domain of interest (e.g. College Management System) and identify multi-tier software applications to work on (e.g. Online Fee Collection). Analyze, design and develop this application:

1. Develop an SRS document. Also develop risk management and project plan (Gantt chart).
2. Understanding of System modeling: Data modeling using ER – Diagram-Draw an ER Diagram which also includes generalization, specialization and aggregation of specified problem statements.

3. Understanding of System modeling: Functional modeling: DFD Context and draw it

4. Understanding of System modeling: UML and draw it.

5. Develop a sample calculator program and perform Black-box Testing:

    a. Identify test scenarios without viewing the internal code.

    b. Write test cases for valid and invalid inputs.

    c. Execute the test cases and record outcomes.

6. Develop a sample login authentication page and perform White--box Testing:

    a. Analyze the code for all possible paths, conditions, and loops.

    b. Apply statement coverage and branch coverage techniques.

    c. Test internal functions with known inputs.

7. For the above login authentication page, perform the following:

    a. Simulate the following maintenance activities:

        • Corrective Maintenance: Fix an existing bug (e.g., wrong output, crash, miscalculation).

        • Perfective Maintenance: Add a new user-requested feature (e.g., sorting, filter, or improved UI).

        • Adaptive Maintenance: Modify the code to adapt to a new platform or library version.

        • Preventive Maintenance: Refactor the code to improve readability, performance, or security.

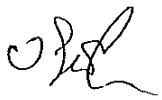
    b. Document each change with:

        • Problem description

        • Type of maintenance

        • Before and after screenshots

        • Change summary



Semester-V

C11: Business Intelligence Tools

Course Objectives:

1. Introduce foundational concepts of Business Intelligence (BI) and Decision Support Systems (DSS), including their scope, evolution, and organizational relevance.
2. Familiarize students with leading BI tools such as Power BI and Tableau, highlighting their ecosystems, interfaces, and comparative strengths.
3. Develop skills in data preparation and transformation, using Power Query and Tableau's data connection features to clean and model datasets.
4. Enable effective data visualization and storytelling, leveraging charts, dashboards, and advanced features to communicate insights.
5. Equip learners with data modeling techniques, including dimensional modeling, relationships, joins, and governance principles for robust BI solutions.

Course Outcomes:

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

1. Differentiate between BI, Data Analytics, and Data Science, and explain the BI lifecycle and its applications across functional domains.
2. Use Power BI and Tableau to prepare, transform, and visualize data, applying basic DAX functions and calculated fields for analysis.
3. Design and implement dimensional data models, including star and snowflake schemas, and apply relationships and joins in BI tools.
4. Create interactive dashboards and visualizations, incorporating parameters, slicers, filters, and drilldowns to enhance decision-making.
5. Build and publish complete BI dashboards, and effectively communicate business insights through storytelling and visualization best practices.

Unit-I: Introduction to Business Intelligence and Decision Support Systems

Business Intelligence: Definition, Scope, and Evolution, Business Intelligence vs. Data Analytics vs. Data Science, BI Lifecycle, **Applications of BI in Functional Domains:** Finance,