SEMESTER-VI

COURSE 14 A: MOBILE APPLICATION DEVELOPMENT

Theory **Credits: 3** **3 hrs/week**

Course Objectives

1. Understand core concepts of mobile app development and differentiate between native and cross-platform approaches.
2. Set up the Flutter and Dart development environment and apply foundational Dart programming constructs.
3. Design interactive and responsive UIs using Flutter widgets and implement custom design elements.
4. Develop multi-screen applications with effective state management and navigation techniques.
5. Integrate external data through APIs, manage local storage, and incorporate Firebase for cloud functionality.
6. Utilize advanced Flutter features, optimize performance, and deploy apps to the Google Play Store.

Course Outcomes

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

1. Configure the Flutter SDK and development tools, and write Dart programs using object-oriented principles and error-handling mechanisms.
2. Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.
3. Implement navigation between screens and manage application state using Provider and other built-in mechanisms.
4. Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.
5. Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms.

Unit 1. Introduction to Flutter and Dart:

Overview of mobile app development trends, Native vs. cross-platform development, Introduction to Flutter SDK and its architecture, Setting up Flutter & Dart environment (IDE, emulator, device) (Install Flutter SDK and set up IDE (VS Code / Android Studio), Dart syntax: variables, data types, control structures, Functions, classes, and object-oriented principles in Dart, Error handling and assertions.

Unit 2. Flutter Widgets and UI Design:
Stateless vs Stateful widgets, Layout widgets: Container, Row, Column, Stack, Input & selection widgets: TextField, Checkbox, Radio, Switch, Styling widgets: Padding, Margin, Fonts, Colors, Custom widgets and theming

Unit 3. Navigation and State Management:
Navigation: Navigator, routes, passing data between screens,

State management: setState, InheritedWidget, Provider, Dialogs, alerts, and snackbars, Forms and validation

Unit 4. Working with Data and APIs:
HTTP package for API calls, JSON parsing and model classes, Displaying data with ListView and GridView, Local storage: SharedPreferences, File handling, Firebase integration (basic setup & Firestore)

Unit 5. Advanced Features and Deployment:
Animations and custom transitions, Accessing device features: Camera, Location, Introduction to packages and plugins, Debugging and performance optimization, Building and releasing apps to Google Play Store

Textbooks:

1. Beginning Flutter: A Hands-On Guide to App Development, Marco L. Napoli, Wiley
2. Flutter for Beginners, Alessandro Biessek, Packt Publishing, 2nd Edition

Reference Books:

1. Flutter for Mobile Apps: Miguel Farmer, Rafael Sanders, Lincoln Publishers
2. Flutter Development Masterclass, E.M. Redwood, 2025

Activities:

Outcome: Configure the Flutter SDK and development tools and write Dart programs using object-oriented principles and error-handling mechanisms.

Activity: Set up Flutter SDK and create a Dart console app that:

- Uses object-oriented principles (classes, inheritance)
- Includes error-handling (try-catch-finally)
- Demonstrates basic input/output

Evaluation Method: Evaluate on a 10-point scale based on a checklist to verify:

- SDK and IDE properly configured
- Correct use of classes and inheritance
- Functional error-handling logic
- Output correctness and code readability

Outcome: Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.

Activity: Build a login screen using:

- Built-in widgets (TextField, Button, Column, etc.)
- Custom styling (colors, fonts, padding)
- Responsive layout

Evaluation Method: Rubric-based assessment on a 10-point scale:

- UI consistency and alignment
- Use of appropriate widgets
- Styling customization
- Responsiveness across screen sizes

Outcome: Implement navigation between screens and manage application state using Provider and other built-in mechanisms.

Activity: Create a multi-screen app with:

- Home, Profile, and Settings screens
- Navigation using Navigator
- State management using Provider (e.g., toggle dark mode)

Evaluation Method: Functional testing (10-point scale):

- Smooth navigation between screens
- Correct state updates via Provider
- Code structure and separation of concerns
- UI reflects state changes

Outcome: Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.

Activity: Build a news app that:

- Fetches articles from a public REST API
- Parses JSON response
- Displays data in ListView and GridView

Evaluation Method: Live demo and code inspection (10-point scale):

- API integration and error handling
- JSON parsing accuracy
- Dynamic UI rendering
- Performance and responsiveness

Outcome: Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms.

Activity: Create a photo journal app that:

- Captures images using device camera

- Retrieves current location
- Applies basic animations (e.g., fade-in)
- Packages and runs on Android device

Evaluation Method: Device-based testing to check (10-point scale):

- Camera and location permissions handled
- Feature functionality verified
- Animation smoothness
- Successful APK build and installation

SEMESTER-VI

COURSE 14 A: MOBILE APPLICATION DEVELOPMENT

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Write Basic Dart Programs Using Variables, Functions, and Classes
2. Design a UI Layout Using Container, Row, Column, and Stack Widgets
3. Create an Interactive Form Using TextField, Checkbox, Radio, and Switch
4. Implement Custom Widgets and Apply Theming (Colors, Fonts, Styles)
5. Navigate Between Screens and Pass Data Using Navigator and Routes
6. Manage State Using setState and Provider Package
7. Create and Validate a Registration Form Using TextFormField and Validators
8. Fetch and Display JSON Data from a Public API Using HTTP Package
9. Parse JSON into Model Classes and Display Using ListView
10. Use SharedPreferences to Store and Retrieve Local Data
11. Integrate Firebase Firestore: Add and Retrieve Data
12. Implement Basic Animations Using AnimatedContainer and Hero Widgets
13. Access Device Camera or Location Using Flutter Plugins
14. Debug and Optimize Flutter Apps Using DevTools

SEMESTER-VI

COURSE 14 B: DATA VISUALIZATION TOOLS

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

- Understand the foundations and effective principles of data visualization.
- Learn to build visualizations using Python's ecosystem.
- Gain proficiency in R's ggplot2 for rich and layered data visualization.
- Understand Tableau for building intuitive, shareable dashboards.
- Apply tools in real-world data scenarios for exploration and insight.

Course Outcomes:

At the end of the course the Students will be able to:

- Analyze and critique data visualizations for effectiveness and ethical integrity.
- Develop static and interactive plots to analyze datasets.
- Construct customized visualizations with ggplot2.
- Create interactive visual stories using Tableau.
- Demonstrate complete visualization workflows from raw data to insight.

Unit 1. Fundamentals of Data Visualization:
Importance of visualization in data science,Types of data (categorical, numerical, time-series, geospatial), Chart types: bar, line, histogram, box plot, heatmaps, maps, Design principles: clarity, simplicity, integrity, aesthetics, Misleading visualizations and ethical considerations

Unit 2. Data Visualization Using Python:
Introduction to matplotlib: plots, customization, styling, Seaborn for statistical data , visualization, Plotly for interactive charts and dashboards, Advanced plots: pairplots, violin plots, heatmaps, time-series plots, Integrating visualizations in Jupyter and web apps

Unit 3. Data Visualization Using R (ggplot2):
Introduction to ggplot2 grammar of graphics,Aesthetics, geometries, scales, themes,Faceting and layering techniques,Visualizing categorical and numerical data, Customizing and exporting plots

Unit 4. Basics of Tableau:
Tableau basics: interface, data connection, Charts, filters,

Aggregation, Calculated values and table calculations, Using the calculation dialog box to create, Building formulas using table calculations, Using table calculation functions.

Unit 5. Interactive Dashboards with Tableau:
Maps, tooltips, and trendlines, generating new data with forecasts, Storytelling with dashboards, providing self evidence adhoc analysis with parameters, Editing views in tableau Server, Publishing and sharing reports

Text Books

1. Data Visualization: A Practical Introduction, Kieran Healy, Princeton University Press, 2019.
2. Learning Tableau 2022, Ben Jones, Packt Publishing, 2022.
3. Python Data Science Handbook, Jake VanderPlas, 2nd Edition, O'Reilly, 2022.

Reference Books

1. Fundamentals of Data Visualization, Claus Wilke, O'Reilly, 2019.
2. Better Data Visualizations, Jonathan Schwabish, Columbia University Press, 2021.
3. ggplot2: Elegant Graphics for Data Analysis, Hadley Wickham, Springer, 3rd Edition, 2023.
4. Tableau eLearning & Public Gallery Resources (https://public.tableau.com)

Activities:

CO1: Analyze and critique data visualizations for effectiveness and ethical integrity

Activity:
Conduct a **visualization audit and critique assignment**. Students select a set of published visualizations (from news, dashboards, or social media) and analyze their design effectiveness, ethical considerations (e.g., misleading visuals, omitted context), and audience impact.

Evaluation Method:

Rubric-based assessment for depth of critique, identification of ethical flaws, clarity of reasoning, and suggestions for improvement - scored on a 10-point scale.

CO2: Develop static and interactive plots to analyze datasets

Activity:
Interactive lab + peer showcase - students create a combination of static (e.g., matplotlib/seaborn) and interactive (e.g., Plotly) plots for a real dataset. Then, participate in a **gallery walk** where students view each other's work and provide structured peer feedback using a guided rubric.
Evaluation Method:

Rubric includes technical accuracy, interactivity, design quality, peer feedback contribution, and clarity of explanation - scored out of 10.

CO3: Construct customized visualizations with ggplot2

Activity:
Mini project in R using ggplot2 - students work on a domain-based dataset (e.g., environmental, healthcare, or education) and apply advanced ggplot2 features like custom themes, annotations, faceting, scales, and coordinate systems to convey insights.

Evaluation Method:

Rubric-based code review and visualization quality evaluation (design, customization, clarity of message) - scored on a 10-point scale.

CO4: Create interactive visual stories using Tableau

Activity:
Storyboarding + Tableau design - students first sketch a **storyboard** outlining the flow of their data story (with objectives, key visuals, and user interaction paths), and then implement the story in Tableau using dashboards and interactive filters.

Evaluation Method:

Rubric evaluates storyboard planning, narrative flow, interactivity, design consistency, and insightfulness - scored on a 10-point scale.

CO5: Demonstrate complete visualization workflows from raw data to insight

Activity:
Capstone visualization project - students select a dataset of interest, perform data cleaning, transformation, EDA, and build a complete multi-tool workflow (e.g., preprocessing in Python, visualization in Tableau/R). They then **present their insights in a recorded video or classroom seminar**.
Evaluation Method:

Rubric includes workflow completeness, tool integration, clarity of insights, storytelling effectiveness, and presentation quality - evaluated on a 10-point scale.

SEMESTER-VI

COURSE 14 B: DATA VISUALIZATION TOOLS

Practical **Credits: 1** **2 hrs/week**

List of Practicals

1. Use matplotlib to generate line, bar, pie, and scatter graphs.
2. Use Seaborn on statistical data visualization.
3. Implement interactive charts and dashboards using Plotly.
4. Generate pairplots, violin plots, heatmaps, time-series plots for data visualization.
5. Generate scatter graph , bar graph, and histogram using ggplot2.
6. Visualize categorical and numerical data using R.
7. On a sample data set(eg., Supermarket / Showroom spread across the states),
    a. Create visualizations
        i. Bar chart showing Sales by Region.
        ii. Line chart of Profit over time.
        iii. Add filters (e.g., by Category or Year).
    b. Aggregations
        i. Use SUM, AVG, and COUNT aggregations.
        ii. Show totals and subtotals on the chart.
    c. Deliverables
        i. Dashboard with at least two charts and interactive filters.
8. Implement the following inTableau
    a. Create Calculated fields like profit ratio
    b. Use Table Calculation functions like RUNNING_SUM(), WINDOW_AVG(), or RANK().
    c. Worksheet showing
        i. Calculated fields in use
        ii. Table calculation in a visualization (e.g., trend chart)
        iii. Proper labels and tooltips
9. Mini-project: Create dashboard for Semester Results Analysis using Python or R.

SEMESTER-VI

COURSE 15 A: MERN STACK

Theory	**Credits: 3**	**3 hrs/week**

Course Objectives:

1. Understand full-stack architecture and the individual roles of the MERN components-MongoDB, Express.js, React.js, and Node.js.
2. Develop interactive front-end applications using React.js and master state management and routing techniques.
3. Model and manipulate NoSQL databases using MongoDB and Mongoose, including CRUD operations and schema validations.
4. Build RESTful APIs using Express.js and integrate server-side logic with frontend and database components.
5. Implement full-stack application features such as authentication, session management, and deployment using modern platforms.

Course Outcomes:

1. Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.
2. Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.
3. Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.
4. Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.
5. Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Unit 1. Introduction to MERN Stack & Node.js:

Introduction to Full Stack Web Development, Frontend vs Backend, What is the MERN stack? Architecture of MERN Applications

Introduction to Node.js, Installing Node.js & npm, Node.js fundamentals (Modules, Events, Streams), Asynchronous Programming & Event Loop, Node.js File System module, npm modules & custom modules, Setting up a basic server with Node.js

Unit 2. React.js (Frontend Framework):

Introduction to React, Functional Components & JSX, State & Props, Handling Events, useState, useEffect Hooks, Conditional Rendering & Lists, React Router (Routing), Forms in React (Controlled Components), Axios for HTTP requests

Unit 3. MongoDB with Mongoose:
Introduction to NoSQL Databases, MongoDB vs SQL Databases, Installing & using MongoDB (local & Atlas), CRUD Operations with MongoDB Shell & Compass, Mongoose ODM, Models, Schemas, Validation, Relationships (One-to-Many, Many-to-Many), Aggregation

Unit 4. Express.js (Backend Framework):
Introduction to Express.js, Creating RESTful APIs using Express, Routing (GET, POST, PUT, DELETE), Middleware in Express, Error Handling, Connecting to MongoDB using Mongoose, Environment variables and `.env` files

Unit 5. Full Stack Integration & Deployment:
Connecting Frontend (React) with Backend (Express), CORS and Proxy setup, Authentication with JWT, Protected Routes in Frontend, Role-based access control, Deployment: Deploying backend to Render/Heroku, Deploying frontend to Netlify/Vercel, Connecting MongoDB Atlas

Textbooks:

1. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node, Subramanian, Vasan, 2nd Edition, Apress,
2. Learning React: Functional Web Development with React and Redux, Alex Banks & Eve Porcello, O'Reilly
3. MongoDB: The Definitive Guide, 3rd Edition, Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, O'Reilly

Reference Books:

1. Ultimate Full-Stack Web Development with MERN, Nabendu Biswas, Orange Education Pvt Ltd.
2. Full Stack Development with MERN, Thompson Carter, Lincoln Publishers

Activities:

Outcome: Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.

Activity: Set up a basic Node.js project with Express and required modules (express, nodemon, dotenv). Create a simple server that responds with Hello MERN Stack on a browser.

Evaluation Method: Checklist-based review (10-point scale):

- Correct installation of Node.js and modules
- Functional server response
- Use of environment variables
- Folder structure and code clarity

Outcome: Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.

Activity: Build a multi-page React app (e.g., user profile and contact form) using:

- Functional components
- useState and useEffect hooks
- Controlled form inputs

- React Router for navigation

Evaluation Method: Live demo and rubric (10-point scale):

- Component structure and reusability
- Hook usage and state management
- Form validation and routing functionality
- UI responsiveness and layout

Outcome: Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.

Activity: Create a student database using MongoDB and Mongoose. Implement:

- Schema with validation rules
- CRUD operations (Create, Read, Update, Delete)
- Reference between collections (e.g., student and course)

Evaluation Method: Code walkthrough and test cases (10-point scale):

- Schema correctness and validation
- CRUD functionality
- Relational mapping using ref
- Console output and error handling

Outcome: Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.

Activity: Develop a REST API for a task manager app using Express.js. Include:

- Routes for task operations
- Middleware for logging and JSON parsing
- Error handling for invalid routes

Evaluation Method: API testing with Postman(10-point scale):

- Route functionality and status codes
- Middleware implementation
- Error response structure
- Code readability and modularity

Outcome: Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Activity: Build a login system with:

- JWT-based authentication
- Protected routes
- Frontend-backend integration
- Deploy frontend on Netlify/Vercel and backend on Render

Evaluation Method: Deployment demo and checklist (10-point scale):

- Token generation and validation
- Secure data flow between client and server
- Working login/logout flow
- Successful deployment and accessibility

SEMESTER-VI

COURSE 15 A: MERN STACK

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Install Node.js and npm; Create and Use Built-in & Custom Node.js Modules
2. Demonstrate Asynchronous Programming Using Callbacks and Promises
3. Implement File Read/Write Operations Using Node.js File System Module
4. Create a React App and Build Functional Components Using JSX
5. Handle Events and Use useState & useEffect Hooks in React
6. Implement Navigation Between Pages Using React Router
7. Create and Validate a Form Using Controlled Components in React
8. Install and Connect to MongoDB (Local and MongoDB Atlas)
9. Perform CRUD Operations Using MongoDB Shell and MongoDB Compass
10. Create Mongoose Schemas and Models, and Connect Them to a Node.js App
11. Create a RESTful API Using Express.js (GET, POST, PUT, DELETE)
12. Use Middleware and Error Handling in an Express App
13. Connect Express App to MongoDB Using Mongoose
14. Implement User Authentication Using JWT in Express and React
15. Create Protected Routes and Role-Based Access Control in React
16. Deploy Backend to Render/Heroku and Frontend to Netlify/Vercel

SEMESTER-VI

COURSE 15 B: MACHINE LEARNING

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. Understand fundamental concepts, types, and applications of machine learning.
2. Develop, evaluate, and optimize machine learning models through preprocessing, training, and feature engineering techniques.
3. Apply supervised and unsupervised learning algorithms to real-world problems using appropriate tools and methods.

Course Outcomes:

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

1. Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.
2. Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.
3. Implement and analyze supervised learning algorithms for regression and classification tasks.
4. Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains

Unit 1. Introduction to Machine Learning:
Introduction to Machine Learning: Types of human learning, What is machine learning?, Types of machine learning: supervised, unsupervised, semi-supervised and reinforcement learning, machine learning activities, applications of machine learning. Types of data in machine learning, Structure of data

Unit 2. Model Preparation, Evaluation and feature engineering:
Data pre-processing, Model selection and training (for supervised learning), Model representation and interpretability, Evaluating machine learning algorithms and performance enhancement of models. What is feature engineering?, Feature transformation, Feature subset selection. Principal component analysis.

Unit 3. Supervised Learning-Regression:
Regression: Introduction of regression, Regression algorithms: Simple linear regression, Multiple linear regression, Polynomial regression model, Logistic regression, Maximum likelihood estimation.

Unit 4. Supervised Learning- Classification:
Introduction of supervised learning, Classification model and learning steps, Classification algorithms: Naïve Bayes classifier, k-Nearest Neighbour (kNN), Decision tree, Support vector machines, Random Forest.

Unit 5. Unsupervised Learning:
Introduction of unsupervised learning, Unsupervised vs supervised learning, Application of unsupervised learning, Clustering and its types, Partitioning method: k-Means and KMedoids, Hierarchical clustering, Density-based methods – DBSCAN.

Case-study of ML applications: Image recognition, speech recognition, Email spam filtering, Online fraud detection and other.

Textbooks:

1. Introduction to Machine Learning, Ethem Alpaydin, MIT Press, Fourth Edition, 2020.
2. Machine Learning: Theory and Practice, M N Murthy, V.S Ananthanarayana, Universities press
3. Machine Learning, S. Sridhar, M. Vijayalakshmi, Second Edition, Oxford University Press

Reference Books:

1. Machine Learning: An Algorithmic Perspective, Second Edition, Stephen Marsland, CRC Press, 2014
2. Machine Learning, Tom Mitchell, McGraw Hill, 3rd Edition.
3. Python Machine Learning, Sebastain Raschka, Vahid Mirjalili , Packt publishing 3rd Edition, 2019.

Activities:

Outcome: Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.

Activity: Prepare a detailed comparative report/chart explaining supervised, unsupervised, and reinforcement learning paradigms, data types, and step-by-step machine learning pipeline stages.
Evaluation Method: Rubric-based assessment evaluating completeness, clarity, correctness, and presentation quality - scored on a 10-point scale.

Outcome: Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.

Activity: Conduct a hands-on lab exercise using a real dataset to perform data cleaning, normalization, feature extraction/selection, and evaluate model performance using metrics like accuracy, precision, recall, and F1-score.

Evaluation Method: Practical assessment including code correctness, applied techniques, and interpretation of evaluation metrics; assessed with a rubric out of 10.

Outcome: Implement and analyze supervised learning algorithms for regression and classification tasks.

Activity: Implement at least two supervised learning algorithms (e.g., Linear Regression and Decision Trees) to solve prediction tasks, followed by comparative analysis of their performance on test datasets.

Evaluation Method: Code and report evaluation focusing on implementation accuracy, performance comparison, and analysis depth; scored on a 10-point rubric.

Outcome: Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains.

Activity: Perform clustering (e.g., K-Means, Hierarchical) on a given dataset and prepare a case study selecting and justifying machine learning methods suited for different application scenarios.
Evaluation Method: Lab practical combined with a written case study; assessed for correct algorithm application, cluster interpretation, and justification of approach - evaluated on a 10-point rubric.

SEMESTER-VI

COURSE 15 B: MACHINE LEARNING

Practical **Credits: 1** **2 hrs/week**

Lab Experiments:

1. Write a python program to import and export data using Pandas library functions.
2. Demonstrate various data pre-processing techniques for a given dataset
3. Implement Dimensionality reduction using the Principal Component Analysis (PCA) method.
4. Write a Python program to demonstrate various Data Visualization Techniques.
5. Implement MLE on a Dataset
6. Implement Simple and Multiple Linear Regression Models.
7. Develop Logistic Regression Model for a given dataset.
8. Develop Decision Tree Classification model for a given dataset and use it to classify a new sample.
9. Implement Naïve Bayes Classification in Python.
10. Develop K-Means for a Given Dataset
11. Build KNN Classification model for a given dataset.
12. Develop DBSCAN on a given Dataset