**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