Gyanmanjari Innovative University

Syllabus
Gyanmanjari Institute of Technology
Semester-5 (B. Tech)

Subject: Advance Mobile Application Development-BET1CE15322

Type of course: Professional Core

Prerequisite: Basic knowledge of mobile application development, fundamentals of Android application structure, programming fundamentals, object-oriented programming concepts, basic Java/Kotlin understanding, logic and problem-solving skills, basic database concepts, and familiarity with Android Studio.

Rationale:

Advance Mobile Application Development enables students to build modern, scalable, and industry-oriented Android applications using Kotlin and advanced Android development practices. This course builds upon basic mobile application development and introduces Kotlin programming, modern UI design, Android Jetpack components, MVVM architecture, local data storage, API integration, asynchronous programming, Firebase services, and application deployment. It focuses on practical skills required to design, develop, test, and deploy real-world mobile applications with clean architecture, efficient data handling, responsive user interfaces, and backend connectivity. The skills gained through this course will prepare students for industry projects, internships, freelancing, startup-based applications, and advanced learning in cloud-connected and enterprise mobile solutions.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
2	0	4	4	100	50	150

Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation.

Course Content:

Sr. No	Course Content	Hrs.	% Weightage
1	**Kotlin Programming Fundamentals** ***Theory Topics:*** Introduction to Kotlin programming language and its importance in modern Android application development, comparison between Java and Kotlin, Kotlin project structure in Android Studio, variables and data types, type inference, operators and expressions, input and output basics, control flow statements, if, when, loops, functions, default arguments, named arguments, lambda expressions, higher-order functions, null safety, nullable and non-nullable types, safe call operator, Elvis operator, non-null assertion operator, classes and objects, primary and secondary constructors, properties and methods, inheritance, interfaces, method overriding, data classes, enum classes, sealed classes, object declaration, companion object, extension functions, Kotlin collections, exception handling, and Kotlin coding practices for Android application development. ***Practical:*** 1. Install and Configure Android Studio for Kotlin Development – Install Android Studio, required SDK components, Kotlin plugin support, and verify the development environment. 2. Create a Kotlin-Based Android Project – Create a new Android project using Kotlin as the programming language and understand the project structure. 3. Execute Basic Kotlin Program – Write and execute Kotlin code using variables, data types, operators, and basic output statements. 4. Implement Decision-Making Statements – Develop Kotlin programs using if, if-else, nested if, and when expression for logical decision-making. 5. Apply Looping Statements – Use for, while, and do-while loops to solve iterative programming problems. 6. Create Functions in Kotlin – Write Kotlin functions using parameters, return values, default arguments, and named arguments. 7. Implement Null Safety Concepts – Demonstrate nullable variables, safe call operator, Elvis operator, and non-null assertion operator.	18	20%

8. Create Classes and Objects – Define Kotlin classes with properties and methods and create objects to access class members.
9. Implement Constructors in Kotlin – Use primary and secondary constructors to initialize object data.
10. Apply Inheritance and Method Overriding – Create parent and child classes and implement method overriding using Kotlin.
11. Implement Interfaces in Kotlin – Define and implement interfaces to achieve abstraction and structured programming.
12. Use Data Classes and Enum Classes – Create data classes to store structured data and enum classes to represent fixed values.
13. Apply Kotlin Collections – Use List, MutableList, Set, Map, and related operations to manage multiple records.
14. Develop a Kotlin-Based Mini Utility App – Create a simple Android application such as calculator, grade calculator, BMI calculator, or unit converter using Kotlin fundamentals.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Code Acceleration :** Optimize and improve given Kotlin code using appropriate functions, collections, null-safety techniques, and efficient programming practices.	15	-
2	**Difference and Comparison Questions:** Compare related Kotlin concepts and explain their differences using suitable technical points and examples.	05	-
3	**Kotlin Logic Implementation and Debugging:** Develop, test, and debug a Kotlin solution by applying control statements, functions, null safety, object-oriented concepts, and collections.	-	10
	Total	20	10

	Examination Style: **1. Code Acceleration: (15 Marks):** • This section consists of three questions, each carrying 05 marks, focusing on Kotlin code optimization and logic improvement. Students will be expected to improve the given Kotlin code by reducing unnecessary statements, improving readability, applying proper functions or collections, handling null safety correctly, and minimizing time and/or space complexity without changing the actual output. The questions will be framed by the faculty to assess the student's ability to write clean, efficient, optimized, and Android-ready Kotlin code. 2. **Difference and Comparison Questions (05 Marks):** • The difference and comparison question carries 05 marks and will be asked from the topics covered in Module 1. Students will be required to clearly differentiate or compare two related Kotlin concepts, terms, or techniques. This task is designed to test conceptual clarity and the ability to highlight precise distinctions between concepts such as Java vs Kotlin, nullable vs non-nullable types, safe call operator vs Elvis operator, class vs object, abstract class vs interface, list vs mutable list, data class vs regular class, and inheritance vs interface implementation. 3. **Kotlin Logic Implementation and Debugging (10 Marks):** • This activity carries 10 marks and focuses on the practical implementation, testing, and debugging of Kotlin programs. Students will be given a problem statement or partially completed Kotlin code and will be required to develop, complete, execute, and debug the solution using appropriate Kotlin concepts. The activity may include variables, data types, operators, control statements, loops, functions, default and named arguments, null safety, classes, objects, constructors, inheritance, interfaces, data classes, enum classes, collections, and exception handling. Students will be assessed on logical correctness, appropriate use of Kotlin features, code readability, error identification, debugging ability, and successful execution of the required output in Android Studio or a suitable Kotlin programming environment.		

2	**Modern Android UI Development** ***Theory Topics:*** Introduction to modern Android UI development using Kotlin, role of XML layouts in Android applications, overview of View system, Activity and Fragment-based UI handling, View Binding for safe access to UI components, basic concept of Data Binding, ConstraintLayout for responsive screen design, LinearLayout, RelativeLayout and FrameLayout revision, Material Design principles, Material Components, TextInputLayout, MaterialButton, CardView, FloatingActionButton, Toolbar and AppBar, RecyclerView, Adapter and ViewHolder pattern, list item layout design, dynamic data rendering, handling click events in RecyclerView, form design and input validation using Kotlin, Toast and Snackbar, AlertDialog and custom dialog, Options Menu and Popup Menu, Bottom Navigation, Navigation Drawer, theme and style management, color resources, drawable resources, dark mode support, responsive UI design for different screen sizes, basic animation in Android UI, and introduction to Jetpack Compose fundamentals. ***Practical:*** 15. Design Login Screen Using Kotlin – Create a login screen using XML layout, EditText, Button, and Kotlin-based validation. 16. Design Registration Form – Build a registration form using ConstraintLayout with proper alignment, input fields, and validation. 17. Implement View Binding – Enable View Binding and access UI components safely without using findViewById. 18. Create Material Design Form – Use Material Components such as TextInputLayout, MaterialButton, CardView, and FloatingActionButton. 19. Implement Event Handling in Kotlin – Handle button clicks, text changes, and user interactions using Kotlin. 20. Display Toast and Snackbar Messages – Show user feedback using Toast and Snackbar based on form validation or user action. 21. Create Custom Alert Dialog – Design and display AlertDialog and custom dialog for confirmation, warning, or user input. 22. Implement Options Menu and Popup Menu – Create menu-based actions using Options Menu and Popup Menu.	18	20%

	23. Design Toolbar and AppBar – Implement a customized Toolbar/AppBar with title, menu icon, and action items. 24. Create RecyclerView List – Display a dynamic list of records using RecyclerView, Adapter, and ViewHolder. 25. Design Card-Based RecyclerView UI – Use CardView with RecyclerView to display structured data such as students, products, courses, or events. 26. Handle RecyclerView Item Clicks – Implement item click events to show details, messages, or navigate to another screen. 27. Implement Bottom Navigation – Create an application with multiple screens using Bottom Navigation. 28. Implement Navigation Drawer – Design a navigation drawer with menu items and screen switching. 29. Apply Themes and Dark Mode Support – Configure color resources, themes, styles, and dark mode compatibility. 30. Develop a Responsive UI Screen – Create a screen that adjusts properly for different screen sizes and orientations. 31. Add Basic UI Animation – Apply basic animation such as fade, slide, scale, or transition effect on UI components. 32. Create a Mini UI-Based Android App – Develop a small Android application such as student profile app, course listing app, event listing app, or feedback form app using modern UI components.		

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**UI Debugging and Layout Correction:** Identify and correct alignment, constraint, spacing, validation, readability, and responsiveness issues in a given Android UI layout.	10	-
2	**Screen Recreation Challenge:** Recreate a given mobile application screen or wireframe using appropriate XML layouts, Kotlin, Material Components, and UI controls.	10	-
3	**Responsive UI Implementation and Testing:** Design, execute, and test a responsive Android UI screen on different screen sizes and orientations using Android Studio and AVD.	-	10
	Total	20	10

Examination Style:

1. **UI Debugging and Layout Correction (10 Marks):**
   - In this section, students will be given an Android UI screen or XML layout containing design issues such as improper alignment, overlapping views, incorrect constraints, poor spacing, missing validation messages, unreadable text, or broken responsiveness. Students must identify the UI problems and correct the layout using proper Android UI components, ConstraintLayout, Material Components, View Binding, and Kotlin-based event handling. This task will evaluate the student's ability to analyze, debug, and improve Android interface design.

2. **Screen Recreation Challenge (10 Marks):**
   - In this section, students will be provided with a reference mobile application screen, wireframe, or screenshot and will be required to recreate the screen in Android Studio

	using XML, Kotlin, Material Components, RecyclerView, Toolbar, menus, dialogs, or navigation components as applicable. The recreated screen should maintain proper alignment, color usage, spacing, responsiveness, and user interaction. This task will evaluate the student's ability to convert a UI reference into a working Android application screen. **3. Responsive UI Implementation and Testing (10 Marks):** • This activity carries 10 marks and focuses on designing, implementing, and testing a responsive Android application screen. Students will be given a UI requirement or partially designed Android screen and will be required to complete the interface using appropriate XML layouts, ConstraintLayout, Material Components, View Binding, Kotlin-based validation, event handling, navigation, and resource management. Students must execute and test the developed screen on different Android Virtual Devices, screen sizes, or orientations and verify layout alignment, spacing, visibility, scrolling behaviour, text readability, component accessibility, user interaction, and responsiveness. Students will be assessed on UI design accuracy, appropriate use of Android components, functional correctness, responsiveness, code organization, validation, and successful execution in Android Studio.		
3	**Local Data Management Using MVVM and Room Database** ***Theory Topics:*** Introduction to Android Jetpack and its importance in modern Android application development, limitations of traditional Android application structure, need for clean architecture, overview of MVC, MVP and MVVM architecture, MVVM architecture components, role of Model, View, ViewModel and Repository, separation of concerns in Android applications, Activity and Fragment lifecycle with architecture components, ViewModel for managing UI-related data, LiveData for observable data handling, StateFlow basics, lifecycle-aware UI updates, Repository pattern for data abstraction, Room Database as modern local database solution, SQLite vs Room comparison, Room components including Entity, DAO and Database class, CRUD operations using Room, relationship between Room and RecyclerView, suspend functions with Room, DataStore	18	20%

	Preferences for storing small app settings. SharedPreferences vs DataStore. dependency management for Jetpack libraries, handling configuration changes. and best practices for scalable, maintainable and testable Android applications. ***Practical:*** 33. Create MVVM-Based Android Project Structure – Organize an Android project using Model. View. ViewModel, Repository, and Data layer. 34. Configure Jetpack Dependencies – Add dependencies for ViewModel. LiveData. Room Database. Coroutines. and DataStore. 35. Implement ViewModel in Android App – Use ViewModel to store and manage UI-related data during configuration changes. 36. Create Repository Class – Implement Repository pattern to separate data handling logic from ViewModel and UI layer. 37. Create Room Entity, DAO and Database Class – Define Entity, DAO methods, and Room Database class for local data storage. 38. Insert Records Using Room Database – Add new records into local database using Room and Kotlin. 39. Retrieve Records from Room Database – Fetch stored records from Room Database and display them in RecyclerView. 40. Update Records Using Room Database – Modify existing records using Room DAO update operation.Super Keyword: Use super to call parent class method and constructor. 41. Delete Records Using Room Database – Delete selected records from Room Database and refresh the displayed list. 42. Implement Search in Room **Database** – Add search functionality to filter local database records. 43. Store User Preferences Using DataStore – Save small app settings such as username. login state. theme mode, or notification preference using DataStore. 44. Handle Configuration Changes Using ViewModel – Demonstrate how ViewModel preserves data during screen rotation. 45. Develop Notes Management App Using MVVM and Room – Build a mini application that performs CRUD operations using MVVM architecture. Room Database, RecyclerView, ViewModel, and Repository pattern. **Evaluation Method:**		

Sr. No.	Evaluation Methods	SEE	CCE
1	**Architecture Flow Development:** Design and implement an application architecture flow using View, ViewModel, Repository, Entity, DAO, and Room Database components.	10	-
2	**MVVM Debugging and Refactoring Task:** Identify architectural issues in a given Android code structure and refactor it by separating UI, business, and database responsibilities using MVVM components.	10	-
3	**Activity Learning Activity: Local Data Management App with CRUD Testing Report:** Develop a local data management Android application using MVVM, Room Database, RecyclerView, and CRUD operations, and submit the source code and testing report through the ERP portal.	-	10
	Total	20	10

Examination Style:

1. **Architecture Flow Development (10 Marks):**
   In this section, students will be given a simple application requirement and will be required to design and implement the basic architecture flow using MVVM components such as View, ViewModel, Repository, Entity, DAO, and Database class. Students must show proper separation of UI logic, business logic, and data layer, and demonstrate how data flows from the database to the user interface.

2. **MVVM Debugging and Refactoring Task (05 Marks):**
   In this section, students will be provided with a poorly structured Android code snippet or project flow where database logic, UI logic, and business logic are mixed together. Students will identify the architectural issues and refactor the given code or flow into proper MVVM structure using ViewModel, Repository, DAO, Entity, and Room Database components. This task will evaluate

	students' ability to analyze code structure, separate responsibilities, improve maintainability, and apply clean architecture principles in Android development. 3. **Active Learning Activity: Local Data Management App with CRUD Testing Report (10 Marks):** In this task, students will develop a small Android application that manages local data using MVVM architecture, ViewModel, Repository pattern, Room Database, and RecyclerView. Students must implement basic CRUD operations and prepare a brief testing report including objective, database table structure, test cases, expected output, actual output, pass/fail status, screenshots, and conclusion. The final source code, output screenshots, and testing report must be uploaded on the ERP portal within the given timeline.		
4	**API Integration and Asynchronous Programming** ***Theory Topics:*** Introduction to client-server communication in mobile applications, need for API integration in Android apps, REST API concepts, HTTP request and response cycle, HTTP methods such as GET, POST, PUT, PATCH and DELETE, JSON data structure, API testing using Postman, Retrofit library, OkHttp client basics, Gson/Moshi converter, API interface creation, data/model classes for API response, handling API response in Android UI, displaying API data using RecyclerView, form submission using POST API, handling loading, success and error states, internet permission, network security configuration, Kotlin Coroutines, suspend functions, Dispatchers, asynchronous programming in Android, background thread execution, exception handling in API calls, image loading from URL using Glide or Coil, search and filter on API data, basic authentication and token-based API communication, and best practices for secure and efficient API-based Android application development. ***Practical:*** 46. Test REST API Using Postman – Test GET, POST, PUT, and DELETE API endpoints and analyze request-response structure. 47. Configure Retrofit in Android Project – Add Retrofit, Gson/Moshi converter, OkHttp, and required internet permission in the Android project.	18	20%

	48. Create API Interface and Model Classes – Define API endpoints and create Kotlin data classes according to JSON response. 49. Fetch Data Using GET API – Consume a GET API using Retrofit and display the response data in the Android UI. 50. Display API Data in RecyclerView – Fetch list-based API data and display it using RecyclerView, Adapter, and ViewHolder. 51. Submit Data Using POST API – Create a form and submit user-entered data to the server using POST API. 52. Update Data Using PUT/PATCH API – Send updated data to the server using PUT or PATCH API request. 53. Delete Data Using DELETE API – Perform delete operation through API and update the UI after successful response. 54. Handle Loading, Success and Error States – Show ProgressBar, success message, error message, and retry option during API communication. 55. Implement Kotlin Coroutines with Retrofit – Use suspend functions and Dispatchers to perform API calls asynchronously. 56. Load Images from URL Using Glide or Coil – Display online images in ImageView or RecyclerView items using an image loading library 57. Implement Search and Filter on API Data – Add search functionality to filter API-based list data in the application 58. Implement Basic Token-Based API Call – Pass token or authorization header in API request and handle protected API response 59. Develop API-Based Mini Application – Build a mini application such as news app, product listing app, weather app, quote app, or student information app using Retrofit, RecyclerView, Coroutines, and proper error handling.		

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**API Response Handling and Data Transformation Challenge:** Analyse a complex API or nested JSON response, create suitable Kotlin data classes and Retrofit methods, and transform the received data for structured UI presentation.	10	-
2	**Asynchronous Flow Debugging Task:** Identify and correct coroutine, dispatcher, repeated API call, error-handling, UI-thread, and loading-state issues in a given asynchronous Android code flow.	10	-
3	**Active Learning Activity: API-Based Social Utility App with Response Testing Report:** Develop an API-based Android application using Retrofit, RecyclerView, Kotlin Coroutines, and loading/error handling, and submit the source code and response testing report through the ERP portal.	-	10
	Total	20	10

Examination Style:

1. **API Response Handling and Data Transformation Challenge (10 Marks):**
   - In this section, students will be given a complex API response scenario or nested JSON data structure and will be required to analyze the response, create suitable Kotlin data classes, define Retrofit interface methods, and transform the received data before displaying it in a proper Android UI. The task will include handling nested objects, arrays, nullable fields, missing values, error responses, loading states, and filtered or categorized output. Students must also apply proper response validation, meaningful error handling, and clean data presentation using RecyclerView or suitable UI components. This task will evaluate students' ability to

	process real-world API responses and convert raw backend data into useful, structured, and user-friendly mobile application output. 2. **Asynchronous Flow Debugging Task (10 Marks):** • In this section, students will be provided with an Android code flow containing problems related to blocking UI thread, improper coroutine usage, missing error handling, repeated API calls, wrong dispatcher usage, or poor loading-state management. Students must identify the issue and correct the flow using Kotlin Coroutines, suspend functions, Dispatchers, try-catch handling, and proper UI state updates. This task will evaluate students' ability to debug asynchronous API communication and improve application stability. 3. **Active Learning Activity ALA: API-Based Social Utility App with Response Testing Report (10 Marks):** • In this task, students will identify a small real-life or society-oriented problem and develop an API-based Android mini application using Retrofit, RecyclerView, Kotlin Coroutines, and proper loading/error handling. Students may create applications such as public information viewer, college notice app, weather awareness app, emergency contact listing app, health tips app, government scheme information app, or any useful API-based concept. Students must prepare a brief response testing report including API details, request type, test cases, expected response, actual response, UI output screenshots, pass/fail status, and conclusion. The final source code, screenshots, and testing report must be uploaded on the ERP portal within the given timeline.		
5	**Firebase Integration and Android Application Deployment** ***Theory Topics:*** Introduction to Firebase and Backend-as-a-Service in Android applications, Firebase project setup and Android Studio integration, Firebase Authentication using email and password, overview of Google Sign-In, Firebase Realtime Database and Cloud Firestore, difference between Realtime Database and Firestore, basic CRUD operations using Firebase database, Firebase Cloud Storage for image and file upload, Firebase Cloud Messaging and push notification basics, Android runtime permissions, camera and gallery integration, location services using Kotlin, foreground and background location overview, WorkManager for background task scheduling, advanced	18	20%

	notification handling, app security fundamentals, API key safety basics, ProGuard and R8 overview, app signing process, keystore generation, versionCode and versionName, debug build vs release build, signed APK/AAB generation, basic Play Store publishing process, and final project development guidelines for a complete Kotlin-based Android application. ***Practical:*** 60. Connect Android App with Firebase – Create a Firebase project and connect it with an Android Studio project using required configuration files. 61. Implement Firebase Authentication – Develop user registration and login functionality using Firebase email-password authentication. 62. Create User Profile Using Firebase – Store and retrieve user profile details using Firebase Realtime Database or Cloud Firestore. 63. Perform Firebase CRUD Operations – Insert, read, update, and delete records using Firebase database services. 64. Upload Image to Firebase Storage – Select an image from gallery and upload it to Firebase Cloud Storage. 65. Retrieve and Display Uploaded Image – Fetch uploaded image URL from Firebase Storage and display it in the Android application. 66. Implement Firebase Push Notification Basics – Configure Firebase Cloud Messaging and display a basic notification in the application. 67. Handle Runtime Permissions – Request and manage permissions such as camera, storage/media, notification, and location permissions. 68. Integrate Location Services – Fetch and display the user's current location using location APIs. 69. Integrate Camera and Gallery Feature – Capture an image using camera or select an image from gallery and display it in the app. 70. Use WorkManager for Background Task – Schedule a background task such as reminder, data sync, or periodic notification using WorkManager. 71. Generate Signed APK/AAB – Configure app version, keystore, build type, and generate a release-ready signed APK or AAB.		

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Feature Integration Challenge:** Integrate advanced Android features such as Firebase services, notifications, camera, location, or WorkManager with proper UI flow, validation, permission handling, and error management.	10	-
2	**Release Readiness and Security Review Task:** Review and correct application permissions, Firebase configuration, API key exposure, signing, version details, release settings, and build-security issues before deployment.	10	-
3	**Active Learning Activity: Firebase-Enabled Android App with Deployment and Testing Report:** Develop a Firebase-enabled Android application, test its integrated features, generate a signed APK/AAB, and upload the source code, report, screenshots, and deployment evidence through the ERP portal.	-	10
	Total	20	10

Examination Style:

1. **Feature Integration Challenge (10 Marks):**
   - In this section, students will be given a real-world mobile application requirement and will be required to integrate one or more advanced Android features such as Firebase Authentication, Firestore/Realtime Database, Firebase Storage, notification, camera/gallery access, location service, or WorkManager. Students must connect the selected feature with proper UI flow, permission handling, error messages, and data validation. This task will evaluate students' ability to combine multiple Android services into a functional and user-oriented application feature.

	2. **Release Readiness and Security Review Task (10 Marks):** • In this section, students will be given an Android project or release scenario and will be required to review the application before deployment. Students must identify issues related to app permissions, Firebase configuration, API key exposure, versionCode/versionName, debug mode, app signing, ProGuard/R8 configuration, unnecessary files, and release build generation. This task will evaluate students' ability to prepare an Android application for safe, professional, and deployment-ready release. 3. **ALA: Firebase-Enabled Android App with Deployment and Testing Report (10 Marks):** • In this task, students will develop a Firebase-enabled Android mini application that solves a small academic, social, or real-life problem by using authentication, database, storage, notification, or location-based features. Students must prepare a brief testing and deployment report including application objective, Firebase services used, feature screenshots, test cases, expected output, actual output, pass/fail status, signed APK/AAB details, and conclusion. The final source code, screenshots, testing report, and deployment evidence must be uploaded on the ERP portal within the given timeline.		

Suggested Specification table with Marks (Theory):100

Distribution of Theory Marks (Revised Bloom's Taxonomy)						
Level	**Remembrance (R)**	**Understanding (U)**	**Application (A)**	**Analyze (N)**	**Evaluate (E)**	**Create (C)**
Weightage %	10%	15%	30%	15%	10%	20%

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcome:

After learning the course, the students should be able to:	
CO1	Explain Kotlin basics for Android development.
CO2	Design responsive Android user interfaces.
CO3	Implement Jetpack, MVVM, Room, and DataStore.
CO4	Develop API-based apps using Retrofit and Coroutines.
CO5	Integrate Firebase, testing, and app deployment.

Instructional Method:

The course delivery method will depend upon the requirement of content and need of students. The teacher in addition to conventional teaching method by black board, may also use any of tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in laboratory.

Reference Books:

[1] Android Programming: The Big Nerd Ranch Guide, by Bill Phillips, Chris Stewart, Kristin Marsicano, and Brian Gardner, Big Nerd Ranch.
[2] Head First Android Development: A Learner's Guide to Building Android Apps with Kotlin, by Dawn Griffiths and David Griffiths, O'Reilly Media.
[3] Kotlin in Action, Second Edition, by Sebastian Aigner, Roman Elizarov, Svetlana Isakova, and Dmitry Jemerov, Manning Publications.
[4] Programming Android with Kotlin, by Pierre-Olivier Laurence, Amanda Hinchman-Dominguez, Mike Dunn, and G. Blake Meike, O'Reilly Media.
[5] Android Studio Development Essentials: Kotlin Edition, by Neil Smyth, Payload Media.
[6] Firebase Essentials – Android Edition, by Neil Smyth, Payload Media.

Suggested Assessment Guidelines:

Module-1: Kotlin Programming Fundamentals

Code Acceleration (15 Marks)		
Criteria	**Description**	**Marks**
Code Analysis	Correct implementation of assigned algorithms with proper logic and structure	05
Code Optimization	Clean, readable, and well-documented code with proper naming and indentation	05
Correctness and Code Quality	Correct execution showing expected input and output results for all algorithms	05
	Total	15

Difference and Comparison Questions (05 Marks)		
Criteria	**Description**	**Marks**
Conceptual Accuracy	Correct understanding and explanation of the given Kotlin concepts, terms, or techniques.	02
Comparison Points	Clear and technically accurate presentation of differences or similarities between the given concepts.	02
Examples and Presentation	Suitable examples and well-organized presentation using appropriate technical terminology.	01
	Total	05

Module-2: Modern Android UI Development

UI Debugging and Layout Correction (10 Marks)		
Criteria	**Description**	**Marks**
UI Issue Identification	Correct identification of alignment, constraint, spacing, readability, validation, and responsiveness issues.	3
Layout Correction	Appropriate correction using ConstraintLayout, Material Components, resources, and suitable UI properties.	3
Functional Validation	Proper functioning of event handling, input validation, user feedback, and UI components after correction.	2
UI and Code Quality	Clean XML and Kotlin code with consistent design and proper resource management.	2
	Total	10

Screen Recreation Challenge (10 Marks)		
Criteria	**Description**	**Marks**
Visual Accuracy	Accurate recreation of the reference screen with proper alignment, spacing, typography, colors, and component placement.	3
UI Component Implementation	Correct use of XML layouts, Material Components, RecyclerView, Toolbar, menus, dialogs, or navigation components.	3
Interaction and Responsiveness	Proper implementation of user interactions and adaptability across different screen sizes and orientations.	2
Code and Resource Management	Well-structured Kotlin and XML code with proper use of drawable, color, style, string, and dimension resources.	2
	Total	10

Module-3: Local Data Management Using MVVM and Room

Architecture Flow Development (10 Marks)		
Criteria	**Description**	**Marks**
Architecture Understanding	Correct identification and understanding of View, ViewModel, Repository, Entity, DAO, and Database components.	2
MVVM Component Development	Proper design and implementation of MVVM components with suitable responsibilities.	3
Data Flow Implementation	Correct demonstration of data flow between Room Database, Repository, ViewModel, and the user interface.	3
Separation and Code Structure	Clear separation of UI, business, and data-access logic with readable and maintainable code.	2
	Total	10

MVVM Debugging and Refactoring (10 Marks)		
Criteria	**Description**	**Marks**
Architectural Issue Identification	Correct identification of mixed responsibilities, direct database access, and improper component dependencies.	2
MVVM Refactoring	Effective restructuring of the given code using ViewModel, Repository, DAO, Entity, and Room Database components.	3
Responsibility Separation	Proper separation of UI logic, business logic, and data-access operations.	3
Correctness and Maintainability	Successful execution of the refactored solution with improved readability and maintainability.	2
	Total	10

Module-4: API Integration and Asynchronous Programming

API Response Handling and Data Transformation Challenge (10 Marks)		
Criteria	**Description**	**Marks**
API Response Analysis	Correct analysis of nested objects, arrays, nullable fields, missing values, and required response data.	2
Data Classes and Retrofit Implementation	Proper creation of Kotlin data classes and Retrofit interface methods for the given API requirement.	3
Data Transformation and UI Presentation	Correct filtering, categorization, transformation, and presentation of API data using suitable UI components.	3
Validation and Error Handling	Proper handling of loading states, missing values, unsuccessful responses, exceptions, and error messages.	2
	Total	10

Asynchronous Flow Debugging (10 Marks)		
Criteria	**Description**	**Marks**
Issue Identification	Correct identification of UI-thread blocking, repeated API calls, improper coroutine usage, and wrong dispatcher selection.	2
Coroutine Flow Correction	Proper use of suspend functions, coroutine scopes, Kotlin Coroutines, and suitable dispatchers.	3
State and Error Management	Effective handling of loading, success, empty, and error states using appropriate exception handling.	3
Stability and Execution	Successful execution without crashes, UI freezing, or unnecessary API calls.	2
	Total	10

Module-5: Firebase Integration and Application Deployment

Feature Integration Challenge (10 Marks)		
Criteria	**Description**	**Marks**
Requirement Analysis	Correct understanding of the application requirement and selection of suitable advanced Android features.	5
Feature Integration	Proper integration of Firebase services, notifications, camera, location, or WorkManager as applicable.	5
Permission, Validation and Error Handling	Correct handling of permissions, input validation, unsuccessful operations, and meaningful user messages.	5
UI Flow and Execution	Smooth connection of the integrated feature with the application UI and successful execution.	5
	Total	10

Release Readiness and Security Review (10 Marks)		
Criteria	**Description**	**Marks**
Security and Configuration Review	Correct identification and correction of permission issues, Firebase configuration problems, API key exposure, and debug settings.	3
Versioning, Signing and Build Configuration	Proper configuration of versionCode, versionName, application signing, ProGuard/R8, and release settings.	3
Project Cleanup and Optimization	Removal of unnecessary files, unused resources, debug code, logs, and insecure configuration details.	2
Release Build and Deployment Readiness	Successful generation and verification of a secure and deployment-ready signed APK or AAB.	2
	Total	20