Gyanmanjari Innovative University

Syllabus
Gyanmanjari Institute of Technology
Semester-4 (B. Tech.)

Subject: Basics of Mobile Application Development - BET1CE14314

Type of course: Multidisciplinary Open Professional Elective courses

Prerequisite: Basic knowledge of object-oriented programming (OOP) using Java

Rationale:

There is a growing number of people using smartphones and tablets, making mobile app development an important and far-reaching field. Android, being open source, offers a major advantage for developers. This course enables students to design and develop mobile applications using Android. As smartphones and mobile apps continue to gain popularity, the demand for skilled professionals in this area is rising. Android application development is highly relevant for computer engineering students as it allows them to apply programming knowledge to create real-world solutions. The course provides hands-on experience in developing Android applications and helps students understand the complete app development process. This practical exposure is essential for converting theoretical concepts into real implementations. Overall, the course builds key competencies in mobile app development, equipping students with the technical skills, practical experience, and career opportunities needed in today's rapidly evolving technology-driven industry.

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.

GYANMANJARI INNOVATIVE UNIVERSITY BHAVNAGAR

Course Content:

Sr. No	Course Content	Hrs.	% Weightage
1	**Introduction to Android:** ***Theory Topics:*** Introduction to Android and its ecosystem, Evolution of Android versions, Key features of Android OS, Android Architecture (Applications, Framework, Libraries, ART, Linux Kernel), Android App Components (Activities, Services, Broadcast Receivers, Content Providers), Basics of APK/AAB, Introduction to Gradle, Overview of Android Studio interface, Understanding Manifest file, Resource folders, and Project Structure. ***Practical:*** 1. Install Android Studio and Required SDK Components – Download and set up Android Studio, install essential SDK packages, and verify successful installation through the SDK Manager. 2. Configure an Android Virtual Device (AVD) – Create and set up an emulator with appropriate device specifications, then verify its functionality by launching it from Android Studio. 3. Initialize a New Android Project – Create a new Android project using the correct package naming convention (com.gmiumad.<appname>) and confirm successful project creation. 4. Explore the Android Project Structure – Navigate through key folders such as java, res, .gradle, and app to understand their purpose and document the role of each directory. 5. Inspect the AndroidManifest.xml File – Open the manifest file to examine app-level configurations such as permissions, activities, intent filters, and document your observations. 6. Execute a Basic "Hello World" Application – Run the default app on an emulator or physical device and verify successful output on the screen. 7. Monitor App Lifecycle Using Logcat – Use Logcat filters to observe activity lifecycle callbacks (onCreate, onStart, onResume, etc.) while running the application. 8. Identify the Purpose of Gradle Files – Examine build.gradle (project and module level) to understand dependencies, build configurations, and document how Gradle influences the Android build process. 9. Analyze Resource Folders – Explore resource directories such as drawable, layout, values, and explain how each folder contributes to the user interface and app functionality.	18	20%

10. Simulate App Behavior Across Devices – Use AVD settings to test your application on various device profiles, screen sizes, and API levels, and record differences in app behavior.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Interface Genesis:** Students will create a simple Android application using basic UI components such as TextView, EditText, and Button, run it on the emulator, and demonstrate the app.	20	
2	**Active Learning Assignment Concept Map Activity:** Students prepare a computer-based mind map on Android Architecture, highlighting its major layers and their interconnections in a clear and structured manner.		10
	Total	20	10

Examination Style:

Interface Genesis (20 Marks)
In this examination, the topic focuses on basic Android user interface development and emulator setup. Students are required to create a simple Android project using basic UI elements such as TextView, EditText, and Button. They must properly create the project with the correct structure, configure an Android Virtual Device (AVD), and successfully execute the application on the emulator. Students should also demonstrate an understanding of essential Android development components such as AVD, Gradle, and SDK tools. Evaluation will be based on the accuracy of the configuration, correctness of the project structure, successful execution of the application, and the student's understanding of the Android development environment.

Students must submit the complete Android project folder along with screenshots of the working application running on the emulator or device. Screenshots must be submitted in the prescribed image formats (JPG/PNG/JPEG). All required files must be compressed into a single ZIP file and submitted as instructed.

Concept Map Activity (10 Marks)
In this activity, students are required to create a computer-based concept map using any digital tool such as draw.io, Canva, Lucidchart, PowerPoint, or any mind-mapping software. The topic of the concept map is Android Architecture, and it must clearly represent its major layers, including Applications, Application Framework, Android Runtime (ART), Native Libraries, and Linux Kernel, along with their

	hierarchy and interrelationships. The concept map should be neat, well-structured, and clearly labeled to demonstrate the student's understanding of the Android architecture. Students must submit the completed concept map in PDF format and upload the PDF on the GMIU Web Portal. If the concept map is created using a tool that does not directly support PDF export, students are required to convert the file into PDF format before submission.		
2	**XML Layouts, UI Design & User Interaction:** ***Theory Topics:*** Introduction to XML in Android UI design, Structure of layout files, Views and ViewGroups, LinearLayout, RelativeLayout, and ConstraintLayout basics, Common UI widgets (TextView, EditText, Button, ImageView, CheckBox, RadioButton), IDs and View Binding, Screen sizes and densities (dp, sp, px), Basics of Themes, Styles, and Resources, Event handling using listeners, Introduction to Toast and Snackbar. ***Practical:*** 11. Design a Login Screen – Use LinearLayout and essential UI widgets (TextView, EditText, Button) to create a functional login interface. 12. Construct a Registration Form – Build a complete registration screen using ConstraintLayout with appropriate constraints applied to all UI components. 13. Arrange UI Components Using RelativeLayout – Position elements based on parent and sibling relationships inside a RelativeLayout. 14. Customize XML UI Components – Modify TextView, EditText, and Button properties such as padding, margin, textSize, colors, and background in XML. 15. Implement a Button Click Event – Write code that updates on-screen text dynamically when a button is clicked. 16. Display User Notifications – Show feedback messages using Toast or Snackbar based on user actions. 17. Group User Options – Use RadioGroup and CheckBox widgets to collect user choices and display the selected output. 18. Apply Custom Themes and Styles – Create and assign styles or themes in styles.xml to maintain consistent UI appearance across the app. 19. Adapt Layouts for Multiple Screen Sizes – Use dp, sp, and appropriate layout qualifiers to ensure proper UI scaling on different devices. 20. Inspect UI Using Layout Inspector – Visualize and analyze the structure, hierarchy, and properties of your layout with Android	18	20%

Studio's Layout Inspector tool.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Prototype Demonstration:** Students will design a complete UI screen using XML and implement event handling such as button clicks, CheckBox/RadioButton selection, Toast/Snackbar output, and updating UI. They must demonstrate layout correctness, component alignment, and working interactions.	20	
2	**Peer Teaching Activity:** Students will explain any ONE UI-related concept (such as ViewGroups, dp/sp units, Themes, Widgets, or Layouts) in depth to their classmates in a 5–10 minute presentation. They may use slides, diagrams, or live demonstrations.		10
	Total	20	10

Examination Style:

Prototype Demonstration (20 Marks)
In this examination, students are required to design an Android user interface screen using XML by applying proper layouts, alignment, and component placement. The topic focuses on UI design and basic interaction handling. Students must include basic widgets such as TextView, EditText, Button, CheckBox, and RadioButton, and implement simple event handling such as button click actions or selection responses. The application should be executed on the emulator to demonstrate functional interactions. Students will be evaluated based on layout correctness, UI structure, responsiveness, and the successful demonstration of functional interactions.

Students must submit the complete Android project folder along with screenshots of the working application running on the emulator or device in the prescribed image formats (JPG/PNG/JPEG). All required files must be compressed into a single ZIP file and submitted as instructed.

	Peer Teaching Activity (10 Marks) In this activity, students are required to deliver a 5–10 minute presentation on any one UI-related concept such as ViewGroups, layout types, dp/sp units, Themes, Styles, or commonly used Widgets. The presentation should explain the selected topic in depth and demonstrate the student's conceptual understanding. Students may use presentation slides, diagrams, or brief live demonstrations to support their explanation. This activity aims to assess clarity of communication, accuracy of the content, and the depth of technical explanation. Marks will be awarded based on correctness, presentation quality, and overall technical understanding. Students must submit their presentation material in PDF format after completing the presentation.		
3	**Activities, Intents, Fragments & Navigation:** ***Theory Topics:*** Introduction to Activities and Activity Lifecycle, Launch modes and Activity states, Explicit and Implicit Intents, Passing data between Activities, Introduction to Fragments and Fragment Lifecycle, Fragment-to-Activity and Fragment-to-Fragment communication, Basics of navigation patterns, Understanding Context, Event handling inside Activities and Fragments, Introduction to Splash Screen and basic app flow design. ***Practical:*** 21. Create a Counter Application – Use Increment and Decrement buttons to update and display a counter value on the screen. 22. Navigate Between Activities – Move from one screen to another using explicit intents. 23. Pass Data Between Activities – Transfer user data using intent extras. 24. Use Implicit Intents – Invoke system apps such as Dialer, Email, or Browser. 25. Integrate Fragments – Add two fragments inside a single activity using a proper layout. 26. Activity–Fragment Communication – Coordinate data exchange between an activity and a fragment. 27. Dynamic Fragment Replacement – Swap fragments at runtime using FragmentManager transitions. 28. Implement Splash Screen – Launch a splash screen activity with a timed delay. 29. Debug Navigation Code – Identify and fix errors in intent or fragment implementation. 30. Design App Navigation Flow – Outline a multi-screen navigation flow using activities and fragments.	18	25%

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Debugging Challenge:** Students will receive Android code containing errors related to Activities, Intents, or Fragments. They must identify the mistakes, correct the code, and make the app run properly. Marks are given for correct debugging, logical fixes, and understanding of the topic.	20	
2	**Active Learning Assignment Storyboard / Wireframe Design:** Students will create a simple navigation flow (Activity → Fragment → Activity) using a digital tool. They must show screens clearly and connect them with arrows. Marks are given for clarity, correctness, and neatness.		10
	Total	20	10

Examination Style:

Debugging Challenge (20 Marks)
In this examination, students will be provided with an Android application containing errors related to core components such as Activities, Intents, or Fragments. The topic focuses on identifying and resolving logical and implementation errors in Android applications. Students are required to analyze the given code, identify the issues, correct the faulty implementation, and ensure that the application runs properly on the emulator or device. During the demonstration of the working solution, students should clearly explain the cause of the errors and describe the logical steps taken to fix them. Evaluation will be based on debugging accuracy, understanding of Android component behavior, and the ability to apply correct programming logic to restore full application functionality.

Students must submit the complete Android project folder along with screenshots of the working application running on the device in appropriate image formats (JPG/PNG/JPEG). All required files must be compressed into a single ZIP file and submitted as instructed.

	Storyboard / Wireframe Design (10 Marks) In this activity, students are required to create a digital storyboard representing a simple navigation flow between one Activity and two Fragments. The topic focuses on understanding screen navigation and user flow in Android applications. Using tools such as draw.io, Wireframe, or Canva, students must design each screen clearly and connect them using arrows to indicate transitions and user actions. Each screen should be properly labeled to describe its purpose and the interaction that triggers the navigation. Students will be assessed based on the clarity, correctness, completeness of the navigation flow, and the overall visual organization of the wireframe. Students must prepare a report that includes the storyboard designs along with brief explanations of each screen and interaction. The final report must be submitted in PDF format and uploaded on the GMIU Web Portal as the final submission.		
4	**Data Storage, System Components, and UI Interaction in Android:** ***Theory Topics:*** Introduction to SharedPreferences for key-value storage, Basics of SQLite database and CRUD operations, SQLiteOpenHelper usage, Introduction to Room Database, Menus in Android including Options Menu, Context Menu, and Popup Menu, AlertDialog and Custom Dialog basics, Notifications and PendingIntent, Introduction to Services (Foreground & Background), Broadcast Receivers and system communication, AlarmManager for scheduled tasks, Basics of WorkManager for modern background work, Introduction to simple View and Property Animations. ***Practical:*** 31. Store User Preferences – Save application settings or user choices using SharedPreferences and verify that the data persists. 32. Retrieve Stored Preferences – Load previously saved SharedPreferences values across multiple app sessions and display them in the UI. 33. Create an SQLite Database – Use SQLiteOpenHelper to define a database schema and generate required tables programmatically. 34. Insert Records into SQLite – Perform CRUD operations to add new records into an SQLite table and confirm insertion using a query. 35. Update Database Records – Modify specific rows in an SQLite table by executing appropriate SQL update commands. 36. Delete Records from SQLite – Remove selected records from the database using structured SQL delete operations and verify the changes. 37. Query SQLite Data – Retrieve records from an SQLite table and	18	20%

display the results dynamically in the application UI (e.g., ListView/RecyclerView).

38. Implement App Menus – Add and display Options Menu, Context Menu, and Popup Menu within an activity to enhance interaction.
39. Render Dialogs – Create simple dialogs, confirmation dialogs, and custom-designed dialogs to interact with the user.
40. Schedule a Notification – Use NotificationManager along with PendingIntent to trigger a scheduled notification at a specified time or event.
41. Activate a BroadcastReceiver – Implement a BroadcastReceiver to detect system events such as device charging state and respond with appropriate actions.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Mini Project Sprint:** Build a small Android app that includes Menus, a Custom Dialog, one Notification, and a simple Animation. The app must run smoothly, and all features must work correctly. Marks are based on proper implementation, functionality, and overall app behavior.	20	
2	**Code Walkthrough:** Explain your SharedPreferences or SQLite CRUD code step-by-step and the code developed during the semester. Marks depend on clarity and correctness.		10
	Total	20	10

Examination Style:

Mini Project Sprint (20 Marks)
In this examination, students are required to develop a small Android application that incorporates features such as Menus, a Custom Dialog, a Notification, and at least one basic Animation. The topic focuses on integrating multiple Android UI and interaction components within a single application. Students must ensure that all implemented features function smoothly, the user interface is well organized, and the overall user experience remains consistent throughout the application. Students should also demonstrate the workflow of each feature and explain how different components interact within the app. Evaluation will be based on implementation accuracy, completeness of features, smooth execution, and clarity of the demonstration.

	Students must submit the complete Android project folder along with screenshots of the working application running on the emulator or device in appropriate image formats (JPG/PNG/JPEG). All required files must be compressed into a single ZIP file and submitted as instructed. **Code Walkthrough (10 Marks)** In this activity, students are required to explain their own SharedPreferences or SQLite CRUD implementation step by step, based on the code developed throughout the semester. The topic focuses on data storage and retrieval in Android applications. Students must clearly describe the purpose of each function, explain how data is stored and retrieved, and justify their coding decisions. This activity aims to assess the student's clarity of explanation, correctness of implementation, depth of understanding, and overall conceptual knowledge. Students must keep their project code ready for personal evaluation during the examination. Marks will be awarded based on the live explanation, correctness of the code, and the student's understanding of the implemented logic. Failure to adequately explain the code may result in loss of marks.		
5	**Google Services and Publishing Application:** ***Theory Topics:*** Working with Google Maps, implementing Geocoding and Reverse Geocoding to convert addresses into coordinates and vice versa, customizing map markers, handling user interactions, Signing the Android Application for secure deployment, managing Versioning to maintain updates, and Publishing the Android Application on platforms like Google Play Store, including preparing the release build and app listing. ***Practical:*** 42. Load Web Pages in WebView – Display a webpage inside a WebView component and include basic navigation controls such as Back and Forward buttons. 43. Refresh WebView Content – Implement a button or swipe-to-refresh gesture to reload the current webpage displayed in the WebView. 44. Embed Google Maps – Integrate Google Maps (free-tier API) into an Android activity and load the map successfully. 45. Add a Map Marker – Place a marker on Google Maps to highlight a fixed predefined location. 46. Enable Map Gestures – Allow users to zoom, pan, and interact with the map using standard touch gestures.	18	15%

47. Switch Map Types – Implement controls to change Google Maps between Normal, Hybrid, Satellite, and Terrain view modes.
48. Display Current Location – Plot the device's real-time GPS location on the Google Map using location services.
49. Integrate OpenStreetMap – Use OpenStreetMap as a free alternative mapping service and render map tiles in an Android activity.
50. Generate a Signed APK – Produce a release-ready signed APK by configuring versionCode, versionName, and keystore settings.
51. Publish an Android App – Prepare release build assets and export the final application package suitable for publishing.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Maps Integration Test (Free Services Only):** Integrate a free mapping service (Google Maps without billing or OpenStreetMap). The app must show the map properly, display at least one marker, and allow zoom/pan. Marks are based on correct integration, working marker, and smooth map interaction.	20	
2	**Active Learning Assignment Release Build Presentation:** Create a short tutorial video showing how to generate a Signed APK/AAB. Include steps like versioning, choosing build variant, generating the file, and locating the output. Marks depend on clarity and accuracy of the explanation.		10
	Total	20	10

Examination Style:

Maps Integration Test (20 Marks)
In this examination, students are required to integrate a completely free mapping service such as Google Maps without billing or OpenStreetMap into an Android application. The topic focuses on map integration and basic map interactions. Students must successfully display the map, place at least one marker, and enable user interactions such as zooming or panning. The application should run correctly on the emulator or device, demonstrating proper integration and

	functionality. Evaluation will be based on correct map integration, clean and logical code structure, and successful execution of the application. Students must submit the complete Android project folder along with screenshots of the working application running on the emulator or device in appropriate image formats (JPG/PNG/JPEG). All required files must be compressed into a single ZIP file and submitted as instructed. **Release Build Presentation (10 Marks)** In this activity, students are required to prepare a tutorial video demonstrating how to generate a signed APK or AAB locally in Android Studio. The topic focuses on the application release process. The tutorial must clearly cover key aspects such as app signing, versioning, build variants, and the output location of the generated APK or AAB file. Students should explain each step properly while demonstrating the process. Marks will be awarded based on clarity of explanation, technical accuracy, and completeness of the demonstrated steps. Students must submit the video file or a shareable video link as instructed. The submitted video should be clear, properly explained, and audible to ensure effective evaluation, and the video link must be uploaded on the GMIU Web Portal or the video file should be submitted as a ZIP file through the GMIU Portal.		

Suggested Specification Table:

Distribution of Marks (Revised Bloom's Taxonomy)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage %	5%	10%	30%	20%	15%	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 the above table.

Course Outcome:

After learning the course, the students should be able to:	
CO1	Explain Android architecture, core components, and the process of setting up the development environment.
CO2	Apply Activities, Intents, and Fragments to build functional multi-screen navigation flows.
CO3	Create responsive and well-structured user interfaces using XML layouts, ViewGroups, widgets, and resource management.
CO4	Analyze application requirements and implement storage, menus, dialogs, notifications, services, and background tasks.
CO5	Evaluate mapping and location-based features and create release-ready applications through signing, versioning, and packaging.

Instructional Method:

The course delivery method will depend upon the requirement of content and needs of students. The teacher, in addition to conventional teaching methods by black board, may also use any 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 the Active Learning Assignment.

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

Reference Books:

[1] Android Programming: The Big Nerd Ranch Guide, 5th Edition, Bill Phillips, Chris Stewart, Kristin Marsicano, Big Nerd Ranch.
[2] Professional Android, 4th Edition, Reto Meier, Ian Lake, Wrox.
[3] Android Programming for Beginners, 3rd Edition, John Horton, Packt Publishing.
[4] Head First Android Development, 2nd Edition, Dawn Griffiths, David Griffiths, O'Reilly Media.
[5] Mobile Application Development: All-in-One, Valentino Lee, Heather Schneider, Robbie Schell, Pearson.