Android Final Year Project Ideas PDF

A good Android final year project is a Kotlin app built with modern tools (Jetpack Compose, ViewModel, Room) that talks to a real back end such as Firebase or a REST API, uses at least one device feature such as camera, GPS or sensors, and works offline where it makes sense. Campus apps, health trackers, delivery tracking and on-device ML apps are strong choices.

Key takeaways

- Use Kotlin and Jetpack Compose; Google recommends them for new Android apps.
- Follow a clear architecture (MVVM with a repository layer).
- Use one device feature: camera, GPS, sensors or notifications.
- Test on a real phone, not only the emulator.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, BCA and MCA students building Android apps. Each plan lists the app's users, the stack (Kotlin, Jetpack Compose, Room, Retrofit, Firebase), modules a team can split (UI, data layer, back end, device features, testing), and viva talking points. Java is also fine if your course uses it.

How to choose your project

- **Solve a campus or local problem** you can test with real users.
- **Plan the back end**: Firebase for speed, or your own REST API for more depth.
- **Decide offline needs** early; it changes the data layer.
- **Check permissions** (location, camera) and explain why each is needed.

Standard level projects

1. Campus Companion App

- **Problem:** Timetable, notices, events and lost-and-found in one app.
- **Data and services:** Your college's public notices and timetable; Firebase.
- **Tech stack:** Kotlin, Jetpack Compose, Firebase Auth, Firestore, Firebase Cloud Messaging.
- **Main modules:** Login, timetable, notices with push, events, lost-and-found posts.
- **Steps:** Design screens, set up Firestore collections, add push for new notices, cache for offline.
- **Show in the viva:** Firestore security rules.

- **Extensions:** Add a canteen pre-order feature.

2. Personal Health and Fitness Tracker

- **Problem:** Track steps, water, sleep and workouts.
- **Data and services:** Phone sensors; Health Connect API where available.
- **Tech stack:** Kotlin, Compose, Room, WorkManager, Health Connect.
- **Main modules:** Step counter, logs, reminders, weekly charts, goals.
- **Steps:** Read step data, store logs in Room, schedule reminders, plot weekly charts.
- **Show in the viva:** How WorkManager runs background tasks.
- **Extensions:** Add cloud backup.

3. Expense Splitter for Groups

- **Problem:** Split shared expenses among friends.
- **Data and services:** No external data; Firebase or your own API.
- **Tech stack:** Kotlin, Compose, Room, Firebase.
- **Main modules:** Groups, expenses, settle-up calculation, reminders, history.
- **Steps:** Model groups and expenses, compute minimal settlements, sync between members.
- **Show in the viva:** The settle-up algorithm.
- **Extensions:** Add UPI deep links for payments.

4. Blood Donor Finder

- **Problem:** Find nearby donors by blood group.
- **Data and services:** Registered users' data with consent; Google Maps SDK.
- **Tech stack:** Kotlin, Compose, Firebase, Google Maps SDK, Location services.
- **Main modules:** Donor registration, search by group and distance, request alerts, privacy controls.
- **Steps:** Store donor locations coarsely, query by distance, send request notifications.
- **Show in the viva:** How you protect donor phone numbers and location.
- **Extensions:** Add donation eligibility reminders.

Advanced level projects

5. Food Delivery Tracking App

- **Problem:** Order food and track delivery on a map in real time.
- **Data and services:** Sample restaurant data; Google Maps SDK.
- **Tech stack:** Kotlin, Compose, Retrofit, Node.js or Spring Boot back end, WebSockets, Maps SDK.
- **Main modules:** Customer app, delivery partner app, order API, live location, notifications.
- **Steps:** Build the API, stream rider location, show it on the customer's map, update order states.
- **Show in the viva:** How live location updates reach the customer.
- **Extensions:** Add estimated arrival time.

6. Offline-first Field Survey App

- **Problem:** Collect survey data in places without internet.

- **Data and services:** Survey forms you design.
- **Tech stack:** Kotlin, Compose, Room, WorkManager, a REST back end.
- **Main modules:** Dynamic forms, photo and GPS capture, local storage, background sync, conflict handling.
- **Steps:** Store responses in Room, queue uploads, sync with retries when online.
- **Show in the viva:** Your sync and retry strategy.
- **Extensions:** Add an admin web dashboard.

7. Plant Disease Detector App

- **Problem:** Diagnose leaf diseases from a phone photo.
- **Data and services:** PlantVillage dataset (Kaggle) for training the model.
- **Tech stack:** Kotlin, CameraX, TensorFlow Lite.
- **Main modules:** Camera capture, on-device model, results with tips, history.
- **Steps:** Train or reuse a MobileNet model, convert to TFLite, run on CameraX frames.
- **Show in the viva:** Why on-device inference helps farmers with poor signal.
- **Extensions:** Add Hindi voice output.

8. Women Safety SOS App

- **Problem:** Send location and alerts to trusted contacts in an emergency.
- **Data and services:** No external data; phone GPS and SMS.
- **Tech stack:** Kotlin, Compose, Location services, SMS, foreground service.
- **Main modules:** Trusted contacts, SOS trigger (button or shake), live location sharing, audio recording.
- **Steps:** Run a foreground service, send SMS with location link, share updates every minute.
- **Show in the viva:** Battery and permission handling.
- **Extensions:** Add a fake call feature.

Research-oriented projects

9. On-device Text Recognition and Translation

- **Problem:** Read printed text from the camera and translate it offline.
- **Data and services:** No dataset needed; Google ML Kit models.
- **Tech stack:** Kotlin, CameraX, ML Kit Text Recognition and Translation.
- **Main modules:** Camera, text recognition, language detection, translation, accuracy test.
- **Steps:** Recognise text, detect language, translate, measure accuracy on sample signboards.
- **Show in the viva:** On-device versus cloud trade-offs.
- **Extensions:** Add Indian scripts supported by ML Kit.

10. App Performance and Battery Study

- **Problem:** Compare battery and memory use of different design choices.
- **Data and services:** One of your apps.
- **Tech stack:** Kotlin, Android Studio Profiler, Macrobenchmark, Baseline Profiles.
- **Main modules:** Baseline measurements, optimisation changes, benchmark runs, results.

- **Steps:** Measure start-up time and frame drops, apply Baseline Profiles and fixes, measure again.
- **Show in the viva:** Before and after results.
- **Extensions:** Compare Compose and View-based screens.

11. Accessibility-first Android App

- **Problem:** Build an app usable with TalkBack and large fonts.
- **Data and services:** Any simple app such as a bus timetable.
- **Tech stack:** Kotlin, Compose, Accessibility Scanner.
- **Main modules:** Content descriptions, touch targets, contrast, TalkBack testing, user feedback.
- **Steps:** Build the app, test with Accessibility Scanner and TalkBack, fix issues, test with users.
- **Show in the viva:** Issues found and fixed.
- **Extensions:** Add voice commands.

12. Secure Notes App with Encryption

- **Problem:** Store notes encrypted on the device.
- **Data and services:** No external data.
- **Tech stack:** Kotlin, Android Keystore, Jetpack Security or encrypted storage, BiometricPrompt.
- **Main modules:** Biometric unlock, encrypted storage, auto-lock, backup option.
- **Steps:** Create keys in the Keystore, encrypt notes, unlock with biometrics.
- **Show in the viva:** Why keys stay in the Keystore.
- **Extensions:** Add encrypted cloud sync.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the app architecture (MVVM layers), screen flow, database schema and API list.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** screenshots, test cases, device and Android version tested, and performance notes.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Putting all logic in Activities or composables with no architecture.
- Asking for permissions the app does not need.
- Testing only on the emulator.

- Leaving API keys inside the APK without restrictions.

Frequently asked questions

Which Android project is best for a final year?

An app that solves a real campus or local problem, uses a back end and one device feature. Campus apps, delivery tracking and on-device ML apps work well.

Kotlin or Java for an Android final year project?

Kotlin is Google's recommended language for Android. Java is still accepted if your course uses it.

Firebase or my own back end?

Firebase is faster to build with. Your own REST API shows more back-end skill. Some teams use Firebase Auth with their own API.

Do I need to publish the app on Play Store?

No. A signed APK and a demo on a real phone are enough for most colleges.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- Android Developers: Jetpack Compose
- Android Developers: Guide to app architecture
- Firebase documentation
- Google ML Kit

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550