PROJECT ABSTRACT

# Personal Vehicle Maintenance Log Android App with Service Reminders and Fuel Cost Analytics

*A native Android app that keeps a complete maintenance diary for each vehicle — fuel fill-ups with automatic mileage and cost-per-km math, service history with next-due tracking, odometer-plus-date reminders, and canvas-drawn fuel cost analytics. Delivered built-to-order with source code, APK, report, PPT and viva kit.*

## 01. Title

Personal Vehicle Maintenance Log Android App with Service Reminders and Fuel Cost Analytics

## 02. Introduction / Background

Vehicle owners routinely miss service intervals and have no real record of what their vehicle costs to run. Manufacturer service books get lost, fuel bills are thrown away, and the true per-kilometre running cost — the number that should drive every buy/sell and maintenance decision — is never computed. Existing maintenance apps typically demand accounts, show ads, or store data in the cloud, while desktop spreadsheets cannot capture a fill-up at the petrol pump. A native Android app with offline-first local storage fills this gap: it is always in the owner's pocket at the fuel station and the service centre, it computes mileage and cost analytics automatically from odometer readings, and it derives reminders from both distance driven and calendar time — the two dimensions every real service schedule uses.

## 03. Problem Statement

Students building this project solve a concrete, everyday problem: two data streams that vehicle owners generate constantly — fuel fill-ups (date, odometer, litres, price) and service events (oil, filters, brakes, PUC, insurance) — are today either not recorded at all or recorded in ways that yield no insight. The project addresses this by building a multi-vehicle Android app that (1) logs fill-ups and derives km/l, total spend and cost/km from odometer deltas, (2) logs service events with next-due targets in kilometres and/or dates, (3) computes a live reminder list such as “Engine oil due in 1,200 km / 45 days”, and (4) renders fuel-cost trend charts on-device — all stored locally in a Room database with no account or network required.

## 04. Objectives

- Maintain multiple vehicle profiles (name, make, type, fuel, current odometer) with per-vehicle data isolation.
- Log fuel fill-ups (date, odometer, litres, rate) and auto-compute per-fill mileage (km/l), total cost and running cost/km.
- Log service events with free-text notes and configurable next-due targets in kilometres, dates, or both.

- Derive a reminder engine that converts next-due targets into “due in X km / Y days” states with overdue / due-soon / on-track urgency.
- Render on-device analytics: spend-per-fill bar chart, mileage-per-fill trend line, and dashboard aggregates (average mileage, cost/km, total spend).
- Deliver the complete student kit: Kotlin source, debug APK, report, PPT and viva Q&A.;

## 05. Proposed Methodology

The app is developed as a native Android project in Kotlin following MVVM. (1) Data layer: a Room database with three entities — Vehicle, FuelEntry and ServiceEntry — plus DAO queries that return per-vehicle histories ordered by odometer. (2) Domain logic: a MileageCalculator derives km/l from consecutive odometer readings; a ReminderEngine maps each service's next-due kilometre/date pair to remaining distance and remaining days, then classifies urgency with configurable thresholds (expected defaults: due-soon within 500 km or 30 days). (3) Presentation: four tabs — Dashboard, Fuel, Service, Reminders — rendered with RecyclerViews and custom canvas-drawn charts; a floating action button opens bottom-sheet forms for fuel, service and vehicle entry with input validation (odometer monotonicity, positive litres/rate). (4) Notifications: WorkManager schedules daily checks that raise local notifications for reminders crossing the due-soon threshold.

## 06. System Architecture / Working

The user selects a vehicle; the Dashboard ViewModel loads its aggregates (average mileage over the last three fill-ups, lifetime cost/km, total fuel spend) and the top upcoming reminders. Each new fill-up inserts a FuelEntry and triggers recomputation of per-fill mileage from the odometer delta with the previous entry. Each service entry stores optional nextKm and nextDate; the ReminderEngine subtracts the vehicle's current odometer and today's date to produce “due in X km” and “in Y days” figures, sorted by urgency. The Fuel tab draws two charts from the same query results: a bar chart of rupees per fill-up and a line chart of km/l per fill-up. All writes go through the Room DAO on a background dispatcher; the UI observes Flow streams so lists and charts update live.

## 07. Hardware / Software Requirements

- Android Studio (Hedgehog or newer) with Kotlin plugin
- Kotlin, Android SDK (minSdk 26, targetSdk 34 — design targets)
- Room persistence library, Lifecycle ViewModel, Kotlin Coroutines/Flow
- WorkManager for scheduled reminder checks
- Custom Canvas / MPAndroidChart-class chart rendering (no paid SDKs)
- Android device or emulator (5.0+) for the demo build
- Git for source versioning

## 08. Expected Outcomes

- A working Android app managing multiple vehicles with isolated fuel and service histories.
- Automatic per-fill mileage, total fuel cost and running cost/km from odometer deltas.

- A reminder list derived from odometer-plus-date logic with urgency classification and local notifications.
- On-device spend and mileage trend charts plus dashboard aggregates.
- A seeded demo dataset (two Indian vehicles, 14 fuel entries) so every screen is demonstrable on first launch.
- Complete report, PPT and viva Q&A; covering Room schema design, the reminder engine and chart rendering.

## 09. Applications

- Personal vehicle expense tracking for two-wheeler and car owners.
- Teaching Android persistence (Room), MVVM, background work and custom chart drawing.
- A portfolio-grade native Android project demonstrating offline-first design.
- Basis for fleet-lite extensions (driver-wise logs) as student future work.

## 10. Future Scope

- Cloud backup and multi-device sync with user consent (optional extension).
- OCR-based fuel-bill scanning to pre-fill litres, rate and total.
- Service-centre locator and cost-comparison module.
- Export of logs and analytics to PDF/CSV for resale documentation.
- Wear-OS glanceable reminder tiles.