ASSIGNMENT

LEVEL 5

Mobile App Development 2 - COM2461-SENG2461 [Student Name] [Student ID]

INSTRUCTION TO CANDIDATES

1. Late submission will be awarded zero (0) unless extenuating circumstances (EC) are upheld.
2. Cases of plagiarism will be penalized.
3. The assignment should be submitted as softcopy via LMS
4. All evidences related to the sprint implementation in group assignment must be show cased in the final documentation.

Table of Contents

1. Introduction
2. Target Users
3. App Features
4. Folder Structure
5. External Packages Used
6. Database Usage Overview (MySQL & SQLite)
7. Use of Provider State Management in the Application
8. Legendary Motors REST API Documentation
   - Authentication Endpoints
   - Vehicle Inventory APIs
   - Order & Allocation APIs
   - Weather API (External)
9. Mobile Sensor Integration in the Legendary Motors App
   - Biometric Authentication
   - Geolocation & Weather
   - Battery Status Monitoring
10. Test Cases
11. User Interface Design and Screens
12. LinkedIn Learning Certification
13. GitHub Repository
14. Reference

Introduction

Legendary Motors is a premium mobile application designed for a luxury car dealership, offering an exclusive platform for high-net-worth clients to browse, reserve, and manage orders for high-performance vehicles. The application serves as a digital showroom, providing users with a sophisticated interface to explore a curated collection of supercars and hypercars.

The app integrates advanced features such as real-time weather updates based on user location to suggest driving conditions, and a seamless checkout process for vehicle allocations. By bridging the gap between physical luxury and digital convenience, Legendary Motors ensures that the experience of buying a car is as premium as the vehicle itself.

The application connects to a centralized Laravel backend for real-time inventory and order management, while utilizing local storage (SQLite) to ensure performance and offline capability.

Target Users

Legendary Motors is designed for: • **High-Net-Worth Individuals (HNWIs)**: Individuals seeking to purchase or reserve exclusive luxury vehicles. • **Car Collectors**: Enthusiasts who want to track the latest arrivals and manage their vehicle allocations. • **Automotive Connoisseurs**: Users who appreciate detailed specifications, high-quality imagery, and a premium digital experience. • **Existing Clients**: Customers who need a dedicated portal to view their order status and communicate with the dealership.

App Features

2. **Dynamic Home Dashboard** • Real-time weather updates based on GPS location. • Featured "Car of the Day" and latest arrivals.

3. **Inventory Management** • High-fidelity image galleries and detailed specifications for each vehicle. • Filter and search capabilities to find specific models. • Caching system to view inventory even when offline.

4. **Order & Allocation Tracking** • Users can view their current vehicle orders and status. • Digital garage view of owned or ordered vehicles.

5. **Battery-Aware Settings** • Monitors device battery level in real-time. • Adjusts UI elements or provides warnings when battery is low during resource-intensive tasks.

6. **Location-Based Services** • Automatic detection of user location to provide local weather context (e.g., "Perfect day for a convertible"). • Showroom locator to find the nearest dealership.

7. **Eye Protection System** • Uses the device's Ambient Light Sensor (PWA compatible). • Automatically switches to High-Contrast Light Mode in bright environments and Dark Mode in low light to reduce eye strain.

8. **Google Identity Integration** • Seamless sign-in using Google accounts for quick onboarding.

9. **Admin Dashboard (Role-Based)** • Special access for administrators to manage fleet and users directly from the app.

Folder Structure

```
lib/
├── api/                 # HTTP client and API constants
├── providers/           # State management (Provider)
│   ├── auth_provider.dart
│   ├── orders_provider.dart
│   ├── weather_provider.dart
│   └── ...
├── screens/             # UI Screens
│   ├── auth/
│   ├── home/
│   ├── inventory/
│   ├── settings/
│   └── ...
├── services/            # Business logic and external services
│   ├── auth_service.dart
│   ├── biometric_service.dart
│   ├── car_service.dart
│   ├── database_service.dart (SQLite)
│   └── ...
├── shared/              # Reusable widgets and models
│   ├── models/
│   ├── widgets/
│   └── utils/
└── main.dart            # Entry point
```

External Packages Used

Package	Usage
`dio`	Used to perform REST API requests (HTTP client) to the Laravel backend.
`provider`	Used for state management across the application.
`sqflite`	Used for local SQLite database to cache cars and orders.
`go_router`	Handles navigation and routing between screens.

| `geolocator` | Accesses the device's GPS to fetch current location for weather data. | | `battery_plus` | Monitors battery status to display level and adjust UI in settings. | | `shared_preferences` | Stores simple key-value data like auth tokens and theme preferences. | | `flutter_stripe` | Integration for secure payment processing (Reservations). | | `google_fonts` | Provides premium typography (Inter/Outfit) for the UI. | | `dart:html` | Web-specific API for Ambient Light Sensor (PWA). |

Database Usage Overview (MySQL & SQLite)

MySQL (Remote): MySQL acts as the primary source of truth, hosted on the backend server (Laravel). The mobile app interacts with this database via a RESTful API layer.

- **User Persistence**: Manages secure user profiles, roles (Admin/Member), and OAuth (Google) mappings.
- **Master Inventory**: Stores the complete vehicle fleet, including high-resolution images, year, brand, model, and nested JSON specifications for power and performance.
- **Transaction Logs**: Records vehicle allocations and payment confirmations of the $5,000 reservation deposits.

SQLite (Local): SQLite is implemented via the `sqflite` package to provide an "offline-first" experience, critical for a luxury app where users may browse in various connectivity environments.

- **Smart Caching**: The `DatabaseService` implements methods like `cacheCars()` and `cacheAllocations()`. When the API is successful, data is mirrored to SQLite.
- **High Performance**: On app launch, the `OrdersProvider` and `InventoryProvider` immediately load data from SQLite to prevent blank screens, while fetching updates in the background.
- **Offline Access**: If network requests fail (e.g. in a garage with poor signal), the services fall back to retrieved local data using `getCachedCars()` and `getCachedAllocations()`.

Use of Provider State Management in the Application

The application utilizes a multi-provider architecture to manage complex states efficiently:

- **`AuthProvider`**: Manages the complete lifecycle of a user session, including token persistence via `SharedPreferences`.
- **`WeatherProvider`**: A specialized reactive provider that combines `Geolocator` data with the OpenWeatherMap API. It updates the UI dynamically based on the user's current city and temperature to personalize the car discovery experience.
- **`OrdersProvider`**: Handles the synchronization between the remote `/orders` endpoint and the local SQLite database. It manages loading states and error handling for the user's private collection.
- **`InventoryProvider`**: Manages the searchable and filterable car list, ensuring smooth UI transitions and efficient data handling.

Legendary Motors REST API Documentation

Base URL: `http://10.0.2.2:8000/api` (Android Emulator) / `localhost:8000/api` (Web)

Authentication Endpoints

POST /login

- **Description**: Authenticates a user and returns a Laravel Sanctum token.
- **Body**: `{ "email": "...", "password": "..." }`
- **Success**: `200 OK` with `{ "user": {...}, "access_token": "..." }`

POST /auth/google

- **Description**: Authenticates via Google Identity.
- **Body**: `{ "token": "...", "provider": "google" }`

Vehicle Inventory APIs

GET /products

- **Description**: Fetches list of all available luxury vehicles for the inventory.
- **Response**: Returns an array of `Car` objects including brand, model, year, and performance specs.

POST /admin/cars (Admin Authorized)

- **Description**: Allows dealership admins to dynamically add new units to the digital showroom.

Order & Allocation APIs

GET /orders (Authenticated)

- **Description**: Retrieves the list of vehicles reserved or purchased by the authenticated user.
- **Logic**: Results are cached into SQLite by the `OrdersProvider`.

Mobile Sensor Integration in the Legendary Motors App

The Legendary Motors app leverages native mobile hardware to bridge the gap between digital interface and physical context.

Geolocation & Weather Context (`geolocator` + OpenWeatherMap)

- **Implementation**: On startup, the `WeatherProvider` fetches coordinates.
- **Feature**: The app dashboard greets users with localized weather (e.g., "It's 28°C in London - perfect day to drive your 911 Turbo"). This creates an immersive, contextual experience that feels "alive".

Battery Health Monitoring (`battery_plus`)

- **Implementation**: A real-time listener in the `SettingsScreen` monitors percentage and charging status.
- **Feature**: Displays a dynamic battery icon (1-bar to full-bar) in the "Device Health" section. This mimics a car's instrument cluster and ensures users are aware of their status while viewing high-energy 4K media content.

Secure Payments (`flutter_stripe`)

- **Implementation**: Integrated with Stripe's native Payment Sheet.
- **Feature**: Allows users to pay a $5,000 reservation fee instantly via credit card, Apple Pay, or Google Pay. The app handles the `clientSecret` flow to ensure PCI-compliant security.

Ambient Light Sensor (Generic Sensor API)

- **Implementation**: Uses `AmbientLightSensor` in PWA through JS interop.
- **Feature**: Provides "Eye Protection Mode" which dynamically adjusts the theme based on environmental lux levels.

- **Bright Light (>500 lux)**: Forces Light Mode for better readability in sun.
- **Dim Light (<50 lux)**: Forces Dark Mode to protect eyes from glare.

Test Cases

Test ID	Name	Description	Expected Outcome	Status
T01	Login Success	Valid credentials	Token stored & navigate to Home	Pass

| T03 | Offline Garage | Kill internet -> Open Garage | Cars load from SQLite cache | Pass | | T04 | GPS Greeting | Enable Location permissions | Header shows correct city and temp | Pass | | T05 | Stripe Checkout | Proceed to pay deposit | Stripe Payment Sheet opens successfully | Pass | | T06 | Battery Level | Check Settings UI | Icon changes based on system battery % | Pass | | T07 | Admin Access | Login with admin email | 'Administration' section appears in settings | Pass | | T08 | Google Auth | Tap Google Login | Google Sign-in flow completes | Pass |

User Interface Design and Screens

1. **Home Screen**: Features a glassmorphic weather card at the top, followed by horizontal scrollable list of "New Arrivals".
2. **Inventory Screen**: A grid view of high-quality car cards. Each card acts as a "PremiumListTile" with Hero animations to the detail view.
3. **Profile/Settings Screen**: A split-panel design (tablet compatible) or scrollable list (mobile) showing user details and the battery monitor. Uses `GoogleFonts.inter` for a clean look.
4. **Garage Screen**: Shows the user's purchased/ordered vehicles with status indicators (e.g., "In Transit", "Delivered").

LinkedIn Learning Certification

Note: Include your specific certification details here. This project utilized concepts from "Flutter Essential Training: Build for Multiple Platforms" and "Flutter: State Management with Provider" to implement the complex architectural patterns found in the app.

GitHub Repository

The complete source code is available at: **https://github.com/NethanK7/LegendaryMotors-App**

Reference

• Google (2024) Flutter documentation. Available at: https://docs.flutter.dev • OpenWeatherMap (2024) API documentation. • Legendary Motors Internal API Docs.