# 🚌 Green Bus App (Frontend Architecture & Roadmap)

This document describes the complete frontend structure, technologies, and development roadmap for the Green Bus App, built using Flutter. It supports both Passenger and Driver modules.

## ✅ Technologies Required (Frontend)

|  |  |
| --- | --- |
| Feature | Technology |
| UI Framework | Flutter |
| State Management | GetX |
| Navigation | GetX Routing |
| Real-Time Location Tracking | geolocator, google\_maps\_flutter |
| Bus Stop Map | Google Maps API |
| QR Code Ticket Verification | qr\_flutter, qr\_code\_scanner |
| Payment UI | flutter\_stripe or custom UI |
| Local Storage | GetStorage or shared\_preferences |
| Notifications | flutter\_local\_notifications |

## 📁 Folder Structure & Responsibilities

* lib/

→ Root directory of the Flutter frontend code.

* main.dart

→ Entry point of the Flutter app.

* app/routes/

→ Holds route names for navigation.

* app/bindings/

→ Initializes controllers for views/pages using GetX.

* app/controllers/

→ Handles business logic and state using GetX.

* app/models/

→ Contains data models like Bus, Ticket, User, etc.

* app/services/

→ For handling logic like location, API, QR scanning.

* app/themes/

→ Defines theme (colors, fonts, styles).

* app/widgets/

→ Reusable UI components like buttons, loaders.

* modules/passenger/

→ Contains views and controllers for passengers.

* modules/driver/

→ Contains views and controllers for drivers.

* utils/

→ Common helper functions (e.g., formatters).

* constants/

→ Static strings, image paths, and other constants.

## 🛣️ Frontend Development Roadmap

1. 1. Set up the Flutter project with GetX.
2. 2. Create folder structure as per design.
3. 3. Design UI for both Passenger and Driver apps.
4. 4. Implement GetX routing and controllers.
5. 5. Integrate Google Maps and show dummy bus locations.
6. 6. Implement ticket QR generation (passenger) and scanning (driver).
7. 7. Build the payment interface with mock flow.
8. 8. Show nearby bus stops using dummy data.
9. 9. Simulate bus capacity tracking (for testing).
10. 10. Prepare for backend integration by mocking APIs.
11. 11. Connect with backend services (in future).
12. 12. Final UI polish and testing.