Installation Guide

Roadside Assistance Platform

Version 1.0.0	Date April 24, 2025
Author Development Team	Last Updated April 24, 2025

Table of Contents

1. Introduction — Page 1
2. Prerequisites — Page 1
3. Client App (Jetpack Compose) — Page 2
4. Provider App (Jetpack Compose) — Page 3
5. Backend API (Express.js) — Page 4
6. Web Showcase (Next.js) — Page 5
7. Admin Dashboard (Next.js) — Page 6
8. Troubleshooting — Page 7

1 Introduction

This guide provides detailed instructions for setting up the Roadside Assistance Platform development environment. The platform consists of four repositories:

- **Client App** - Mobile application for customers built with Jetpack Compose
- **Provider App** - Mobile application for service providers built with Jetpack Compose
- **Backend API** - Server-side application built with Express.js and Prisma ORM
- **Web Showcase** - Marketing website built with Next.js

Follow the instructions in each section to set up the respective components of the platform.

2 Prerequisites

Before you begin, ensure you have the following tools and software installed on your development machine:

Component	Requirements
All Repositories	• Git • GitHub account with repository access
Mobile Apps	• Android Studio (latest version) • JDK 11 or higher • Kotlin 1.5.0 or higher • Mapbox account with access token
Backend API	• Node.js (v16 or higher) • npm or yarn • PostgreSQL (local or cloud instance) • Prisma CLI
Web Showcase	• Node.js (v16 or higher) • npm or yarn

3 Client App (Jetpack Compose)

Repository

github.com/roadside-assistance-platform/roadside-client-app

Clone the Repository

```
$ git clone https://github.com/roadside-assistance-platform/roadside-client-app.git
$ cd roadside-client-app
```

Setup Instructions

1. Open the project in Android Studio
   Launch Android Studio and select "Open an Existing Project". Navigate to the cloned repository folder and click "Open".

2. Configure local.properties
   Create a `local.properties` file in the project root with your Android SDK path:

   ```
   sdk.dir=/path/to/your/Android/Sdk
   ```

3. Configure Mapbox access token
   Update the `values.xml` file to include your Mapbox access token:

   ```
   <!-- app/src/main/res/values/values.xml -->
   <resources>
       <string name="mapbox_access_token">YOUR_MAPBOX_ACCESS_TOKEN</string>
       <!-- other values -->
   </resources>
   ```

4. Build the project
   In Android Studio, select "Build > Make Project" or use the keyboard shortcut (Ctrl+F9 on Windows/Linux, Cmd+F9 on macOS).

5. Run the app
   Connect an Android device or start an emulator, then click the "Run" button in Android Studio.

4 Provider App (Jetpack Compose)

Repository

github.com/roadside-assistance-platform/roadside-assistance-android-provider

Clone the Repository

```
$ git clone https://github.com/roadside-assistance-platform/roadside-assistance-android-provider.git
$ cd roadside-assistance-android-provider
```

Setup Instructions

1. Open the project in Android Studio
   Launch Android Studio and select "Open an Existing Project". Navigate to the cloned repository folder and click "Open".

2. Configure local.properties
   Create a `local.properties` file in the project root with your Android SDK path:

   ```
   sdk.dir=/path/to/your/Android/Sdk
   ```

3. Configure Mapbox access token
   Update the `values.xml` file to include your Mapbox access token:

   ```
   <!-- app/src/main/res/values/values.xml -->
   <resources>
       <string name="mapbox_access_token">YOUR_MAPBOX_ACCESS_TOKEN</string>
       <!-- other values -->
   </resources>
   ```

4. Build the project
   In Android Studio, select "Build > Make Project" or use the keyboard shortcut (Ctrl+F9 on Windows/Linux, Cmd+F9 on macOS).

5. Run the app
   Connect an Android device or start an emulator, then click the "Run" button in Android Studio.

5 Backend API (Express.js)

Repository

github.com/roadside-assistance-platform/roadside-assistance-backend

Clone the Repository

```
$ git clone https://github.com/roadside-assistance-platform/roadside-assistance-backend.git
$ cd roadside-assistance-backend
```

Setup Instructions

1. Install dependencies

```
$ npm install # or $ yarn install
```

2. Configure environment variables

Create a `.env` file in the project root:

```
$ touch .env
```

Edit the `.env` file with your configuration:

```
# Environment variables declared in this file are automatically made available to Prisma. # See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema # Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB. # See the documentation for all the connection string options: https://pris.ly/d/connection-strings # DATABASE_URL="postgresql://0xapma:0xapma_power@localhost:5432/roadside_assistance_platform?schema=public" # Connect to Supabase via connection pooling with Supavisor. DATABASE_URL="postgresql://postgres.ivuqwmcghhawiwquioxs:edvqN9KNj!MuL*H@aws-0-eu-west-3.pooler.supabase.com:6543/postgres?pgbouncer=true" # Direct connection to the database. Used for migrations. DIRECT_URL="postgresql://postgres.ivuqwmcghhawiwquioxs:edvqN9KNj!MuL*H@aws-0-eu-west-3.pooler.supabase.com:5432/postgres" PORT=8080 SESSION_SECRET="default_secret" GOOGLE_CLIENT_ID="469116526388-14fltok5uk1qo6gf4jup9eu57m558h6v.apps.googleusercontent.com" GOOGLE_CLIENT_SECRET="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" GOOGLE_CLIENT_CALLBACK_URL="http://localhost:8080/auth/google/client/callback" GOOGLE_PROVIDER_CALLBACK_URL="http://localhost:8080/auth/google/provider/callback" RABBITMQ_URL="amqps://xcikzbue:mEXrzOw8yh9B4vwXggUbhIhQj2XjPIES@rat.rmq2.cloudamqp.com/xcikzbue" EMAIL_USERNAME="nodemailer2cp@gmail.com" EMAIL_PASSWORD="zpbo igmr dvem bhoy"
```

3. ### Set up the database with Prisma

   Ensure PostgreSQL is running, then set up the database with Prisma:

   ```
   $ npx prisma migrate dev # or $ yarn prisma migrate dev
   ```

   This will create the database tables based on your Prisma schema and apply any pending migrations.

4. ### Seed the database (optional)

   To populate the database with initial data:

   ```
   $ npx prisma db seed # or $ yarn prisma db seed
   ```

5. ### Generate Prisma client

   Generate the Prisma client to interact with your database:

   ```
   $ npx prisma generate # or $ yarn prisma generate
   ```

6. ### Start the development server

   ```
   $ npm run dev # or $ yarn dev
   ```

   The API server will start on `http://localhost:3000`.

7. ### Verify the installation

   Open your browser or use a tool like curl or Postman to access:

   ```
   $ curl http://localhost:3000/home
   ```

   You should see a response indicating the API is running.

6 Web Showcase (Next.js)

Repository

github.com/roadside-assistance-platform/roadside-assistance-showcase-web

Clone the Repository

```
$ git clone https://github.com/roadside-assistance-platform/roadside-assistance-showcase-web.git $ cd roadside-assistance-showcase-web
```

Setup Instructions

1. Install dependencies

   ```
   $ npm install # or $ yarn install
   ```

2. Start the development server

   ```
   $ npm run dev # or $ yarn dev
   ```

   The web showcase will start on `http://localhost:3000`.

3. Build for production (optional)

   To create a production build:

   ```
   $ npm run build # or $ yarn build
   ```

   To start the production server:

   ```
   $ npm run start # or $ yarn start
   ```

7 Admin Dashboard (Next.js)

Repository

github.com/roadside-assistance-platform/AdminWeb

Clone the Repository

```
$ git clone https://github.com/roadside-assistance-platform/AdminWeb.git $ cd AdminWeb
```

Setup Instructions

1. Install dependencies

```
$ npm install # or $ yarn install
```

2. Configure environment variables

Create a `.env` file in the project root:

```
$ touch .env
```

Edit the `.env` file with your configuration:

```
NEXT_PUBLIC_ADMIN_API_URL="https://roadside-assistance-backend.onrender.com"
REACT_APP_API_URL="http://localhost:5000"
```

3. Start the development server

```
$ npm run dev # or $ yarn dev
```

The admin dashboard will start on `http://localhost:3000`.

4. Build for production (optional)

To create a production build:

```
$ npm run build # or $ yarn build
```

To start the production server:

```
$ npm run start # or $ yarn start
```

7 Troubleshooting

Common Issues

Android Studio build failures

If you encounter build failures in Android Studio:

1. Ensure you have the latest Android Studio version installed
2. Try "File > Invalidate Caches / Restart"
3. Check that your Gradle version is compatible with your Android Studio version
4. Verify that all dependencies in build.gradle are available and accessible

PostgreSQL connection issues

If the backend API cannot connect to PostgreSQL:

1. Verify PostgreSQL is running on your system or that your cloud PostgreSQL instance is accessible
2. Check that the DATABASE_URL in your .env file is correct
3. Ensure network settings allow connections to the PostgreSQL port (default 5432)
4. Check PostgreSQL logs for any authentication or access issues
5. Verify that the database specified in the connection string exists

Prisma migration issues

If you encounter issues with Prisma migrations:

1. Reset the database if you're in development and can afford to lose data:

```
$ npx prisma migrate reset
```

2. Check that your Prisma schema is valid:

```
$ npx prisma validate
```

3. Ensure you've generated the Prisma client after schema changes:

```
$ npx prisma generate
```

API connection from mobile apps

If the mobile apps cannot connect to the API:

1. Ensure the API is running and accessible from your device or emulator

2. For emulators, use 10.0.2.2 instead of localhost in your API URL
3. Check that the API URL in the app's config.json is correct
4. Verify network permissions in the Android manifest
5. Check for CORS issues if applicable

Mapbox integration issues

If you encounter issues with Mapbox integration:

1. Verify your Mapbox access token is valid and has the necessary permissions
2. Ensure the token is correctly set in values.xml for Android apps
3. Check that you've enabled the necessary Mapbox APIs in your Mapbox account
4. For web applications, ensure the token is properly set in environment variables
5. Check browser console for any CORS or API-related errors

Getting Help

If you continue to experience issues after trying the troubleshooting steps above, please:

- Check the repository issues section for similar problems and solutions
- Consult the project documentation in the docs/ folder of each repository
- Contact the development team via Slack or email at support@roadside-assistance.com

© 2025 Roadside Assistance Platform. All rights reserved.
Version 1.0.0 | Last updated: April 24, 2025