Dealership Management Application (Flutter / Node.js)

This is a full-stack application designed to manage operations within a car dealership, built using **Flutter** for the frontend and a **Node.js/Express** backend with a **MySQL** database for data persistence.

The application supports two primary user roles: **Admin** and **Normal User**, with access control enforced across different features.

🚀 Key Features

User Authentication & Management

- **Role-Based Access Control (RBAC):** Differentiates between `admin` and `user` roles at login.
- **User Management (Admin Only):** Admins can view, create, edit, and delete user accounts.
- **Profile Editing (All Users):** Users can update their name and password without logging out.

Dealership Operations

- **Car Management (`/cars`):**
  - **Admin:** Full CRUD (Create, Read, Update, Delete) access to the car inventory.
  - **Normal User:** Read-only access to view the available car listings.
- **Dealer Contacts (`/contacts`):**
  - **Admin:** Full CRUD access to manage contact information for dealers.
  - **Normal User:** Read-only access to view dealer contact information.
- **Sales Reporting (`/reports` - Admin Only):** Access to sales and inventory reports.

⚙️ Project Structure and Setup

1. Backend Setup (Express/MySQL)

The backend is built with Node.js and Express, connecting to a persistent MySQL database.

A. Prerequisites

1. Node.js (and npm) installed.
2. MySQL Server running (e.g., via XAMPP, Docker, or local installation).

B. Database Configuration

1. Connect to your MySQL instance.
2. Execute the script in `schema.sql` to create the `flutter_express_db` database and populate the `users` , `cars` , and `contacts` tables with initial data.

Note: The `server.js` file is configured for standard localhost access (`host: 'localhost'` , `user: 'root'` , `password: ''`). Adjust these settings in `server.js` if your MySQL environment requires different credentials.

C. Installation & Running the Server

1. In the backend directory (where `server.js` is located), install the necessary Node.js packages:

```
npm install express mysql2 cors body-parser
```

2. Start the Express server:

```
node server.js
```

The server will be running on `http://localhost:3000` .

2. Frontend Setup (Flutter)

A. Prerequisites

- Flutter SDK installed and configured.
- A running backend (see step 1).

B. Running the App

1. Navigate to the project root directory.
2. Run the app on your preferred device or emulator:

```
flutter run
```

Note: The Flutter app uses `http://10.0.2.2:3000` to connect to the backend running on the host machine when using the Android emulator.

🛠 Technology Stack

- **Frontend:** Flutter
- **Backend:** Node.js (Express)
- **Database:** MySQL
- **Networking:** `http` package for REST API communication.

🔑 Default Credentials

Role	Email	Password