



System Management

Page by: mayache-

This is a full-stack prototype for a car rental management system designed to manage car rentals, rental locations, and user accounts with distinct roles (Admin and Normal User). The system supports web (Next.js with TypeScript) and desktop (Electron) applications, with a Django backend for data management. Key features include car availability checking, user management, and tracking user login times.

Features

- **User Authentication:** Secure login with JWT and NextAuth.js.
- **Role-Based Access:**
 - **Admin:** Manage users, cars, rentals, locations, and view login history.
 - **Normal User:** Book rentals, view personal bookings, and see available cars/locations.

- **Dashboard:** Role-based statistics (Admins see all data; Normal Users see their bookings).
- **Car Management:** Add, edit, delete cars with location assignments (Admin only).
- **Rental Management:** Create and manage rentals with availability checks (both roles).
- **Location Management:** Add, edit, delete rental locations (Admin only).
- **User Management:** Add, edit, delete users and track login history (Admin only).
- **Advanced Feature:** Check car availability based on rental date ranges.
- **Responsive Design:** Ensures usability across devices.
- **Desktop Support:** Packaged as a desktop app using Electron.

Implemented Technologies

- TypeScript
- Next.js
- Material-UI
- Tailwind
- NextAuth.js
- Python
- Django
- Django REST Framework
- PostgreSQL
- Electron
- Docker

Modules

Front-End

- ☐ Login Page
 - ☐ Username and Password Fields
 - ☐ JWT Authentication via NextAuth.js
 - ☐ Role Identification (**Owner/Employees**)
- ☐ Dashboard Page
 - ☐ **Owner:** Total Cars, Rentals, Users, Locations
 - ☐ **Employees:** Personal Bookings
 - ☐ Recent Activity
- ☐ Cars Page
 - ☐ List of Cars (Make, Model, Year, Location, Status)
 - ☐ Add Car Form (**Owner**)
 - ☐ Edit Car Form (**Owner**)
 - ☐ Delete Car Option (**Owner**)
 - ☐ View Available Cars (Both Roles)
- ☐ Rentals Page
 - ☐ List of Rentals (Customer, Car, Dates, Status)
 - ☐ Create Rental Form (Both Roles)
 - ☐ Availability Check for Selected Dates
 - ☐ **Owner:** View Own Rentals Only
- ☐ Locations Page
 - ☐ List of Locations
 - ☐ Add Location Form (**Owner**)
 - ☐ Edit Location Form (**Owner**)
 - ☐ Delete Location Option (**Owner**)
 - ☐ View Locations (Both Roles)

- ☐ Users Page (**Owner**)
 - ☐ List of Users (Username, Email, Role)
 - ☐ Add User Form
 - ☐ Edit User Form
 - ☐ Delete User Option
 - ☐ View Login History
- ☐ Profile Page (**Employees**)
 - ☐ View Personal Bookings
 - ☐ View Profile Details
- ☐ Navigation
 - ☐ Sidebar with Role-Based Links
 - ☐ Logo
 - ☐ User Profile (Username, Role, Logout)
- ☐ Single Page Navigation Using Next.js
- ☐ Handle Login Form TypeScript
- ☐ Handle Car Form TypeScript
- ☐ Handle Rental Form TypeScript
- ☐ Handle Location Form TypeScript
- ☐ Handle User Form TypeScript
- ☐ Ensure Responsiveness Across All Designs

Back-End

- ☐ Account Management
 - ☐ Handle Account Creation
 - ☐ Handle Account Deletion
 - ☐ Generate JWT for Each User

- ☐ Authenticate API Requests
- ☐ Role-Based Access Control
- ☐ API Development
 - ☐ REST API for Cars
 - ☐ REST API for Rentals
 - ☐ REST API for Locations
 - ☐ REST API for Users
 - ☐ REST API for Login History
- ☐ Rental Validation
 - ☐ Check Car Availability (No Overlapping Dates)
- ☐ Login Time Tracking
 - ☐ Store Login Timestamps
- ☐ Handle All Aspects Using Docker Containers

Database

Users Data

- ☐ User ID
- ☐ Username
- ☐ Email
- ☐ Hashed Password
- ☐ Role (Admin, Normal User)

Cars Data

- ☐ Car ID
- ☐ Make
- ☐ Model

- ☐ Year
- ☐ Location ID
- ☐ Status (Available, Rented)

Rentals Data

- ☐ Rental ID
- ☐ Customer ID (User)
- ☐ Car ID
- ☐ Start Date
- ☐ End Date
- ☐ Status (Active, Completed)

Locations Data

- ☐ Location ID
- ☐ Location Name
- ☐ Address

Login History Data

- ☐ History ID
- ☐ User ID
- ☐ Login Timestamp

Security

- ☐ Hash User Passwords
- ☐ JWT Authentication
- ☐ Role-Based Authorization
- ☐ Validate All Form Inputs
- ☐ Protect Against SQL Injections

Files Structure

```
car-rental-system/  
├── frontend/  
│   ├── pages/  
│   │   ├── index.tsx  
│   │   ├── login.tsx  
│   │   ├── dashboard.tsx  
│   │   ├── cars.tsx  
│   │   ├── rentals.tsx  
│   │   ├── locations.tsx  
│   │   ├── users.tsx  
│   │   └── profile.tsx  
│   ├── components/  
│   │   ├── Navbar.tsx  
│   │   ├── CarForm.tsx  
│   │   ├── RentalForm.tsx  
│   │   ├── LocationForm.tsx  
│   │   └── UserForm.tsx  
│   ├── public/  
│   │   └── logo.png  
│   ├── styles/  
│   │   └── globals.css  
│   ├── lib/  
│   │   ├── auth.ts  
│   │   └── api.ts  
│   ├── types/  
│   │   └── index.ts  
│   ├── Dockerfile  
│   ├── next.config.js  
│   ├── tsconfig.json  
│   └── package.json  
└── backend/
```

```
| |— car_rental/
| | |— __init__.py
| | |— settings.py
| | |— urls.py
| | |— asgi.py
| | |— wsgi.py
| |— api/
| | |— __init__.py
| | |— models.py
| | |— views.py
| | |— serializers.py
| | |— urls.py
| |— migrations/
| |— manage.py
| |— requirements.txt
|— config/
| |— docker-compose.yml
|— main.js
|— README.md
```

Usage

- **Login:** Use credentials created via the Users page (e.g., `admin` / `password` for an Admin user).
- **Admin:**
 - Dashboard: View all stats (cars, rentals, users, locations).
 - Cars: Add/edit/delete cars, assign to locations.
 - Rentals: Manage all rentals with availability checks.
 - Locations: Add/edit/delete locations.
 - Users: Add/edit/delete users, view login history.
- **Normal User:**

- Dashboard: View personal bookings.
- Rentals: Book rentals, view own bookings.
- Profile: View booking history and profile details.
- Cars/Locations: View available cars and locations.

Notes

- The prototype uses SQLite for simplicity; production should use PostgreSQL.
- Login history is stored in the `LoginHistory` model and accessible to Admins via the Users page.
- The availability check is handled by the Django backend to prevent overlapping rentals.
- TypeScript ensures type safety for frontend API interactions.
- The desktop version reuses the Next.js app, packaged with Electron.
- For production, enhance security with input validation, SQL injection protection, and XSS prevention.

Future Enhancements

- Add file uploads for car images and user profile pictures.
- Implement real-time rental status updates via WebSockets.
- Support multiple languages.
- Improve mobile responsiveness.