CARPOOLING SYSTEM - USER MANUAL & PROJECT REPORT

Project Title: Carpooling System: A Smart and Privacy-Focused Ride-Sharing Solution

Developer Name: Lohith Marneni - AP22110011121

BTech, SRM University-AP

LinkedIn | GitHub

Table of Contents:

Introduction

The Carpooling System is a full-stack ride-sharing web application developed using the MERN (MongoDB, Express, React, Node) stack. It connects users who want to offer rides (**Drivers**) with those seeking rides (**Riders**), with built-in features for safety, privacy, and intelligent matching.

System Overview

The platform enables:

- Drivers to post ride details (time, route, seats).
- Riders to search, filter, and request to join rides.
- Intelligent matching based on routes, timing, and user preferences.
- Emergency and privacy features for secure ridesharing.

Modules and Functionalities

1. Authentication

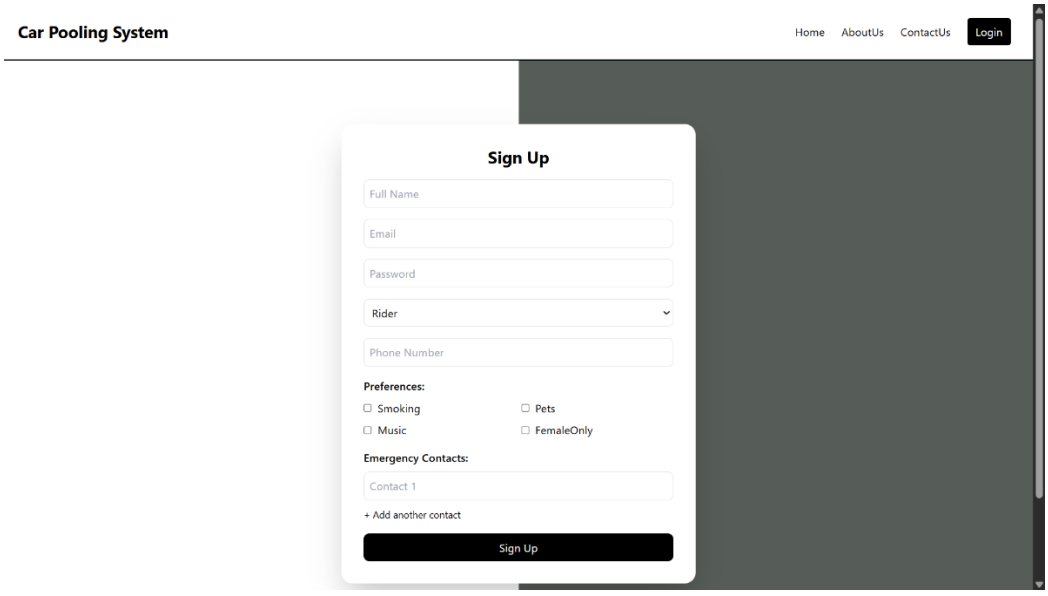

Figure 1: Register Functionality

- Registration and Login

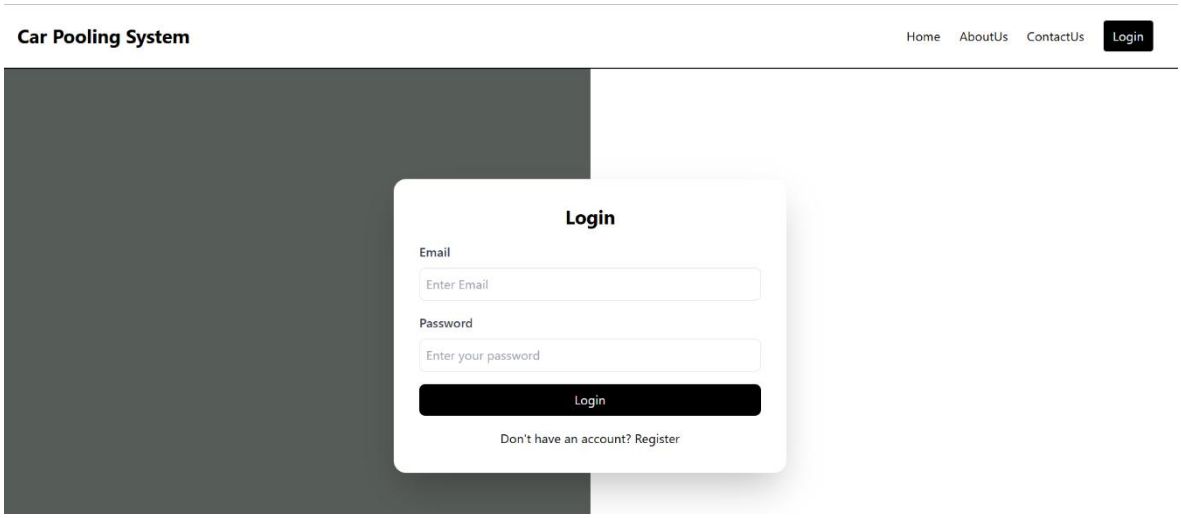

Figure 2: Login Funtionality

- JWT-based session management
- Role-based access (Driver vs. Rider)

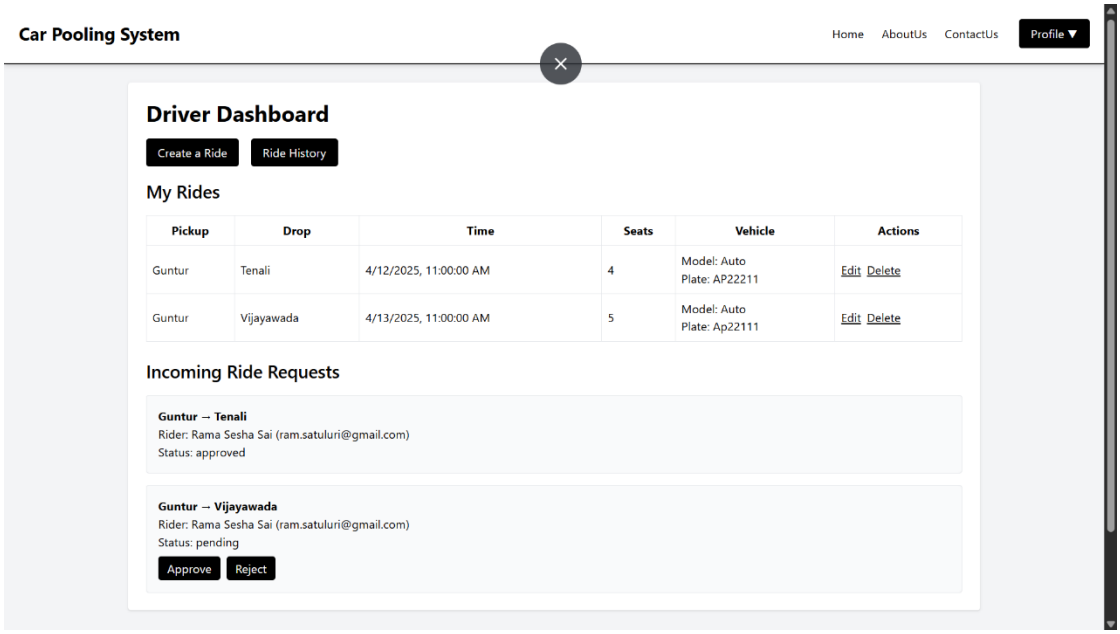

Figure 3: Driver Dashboard

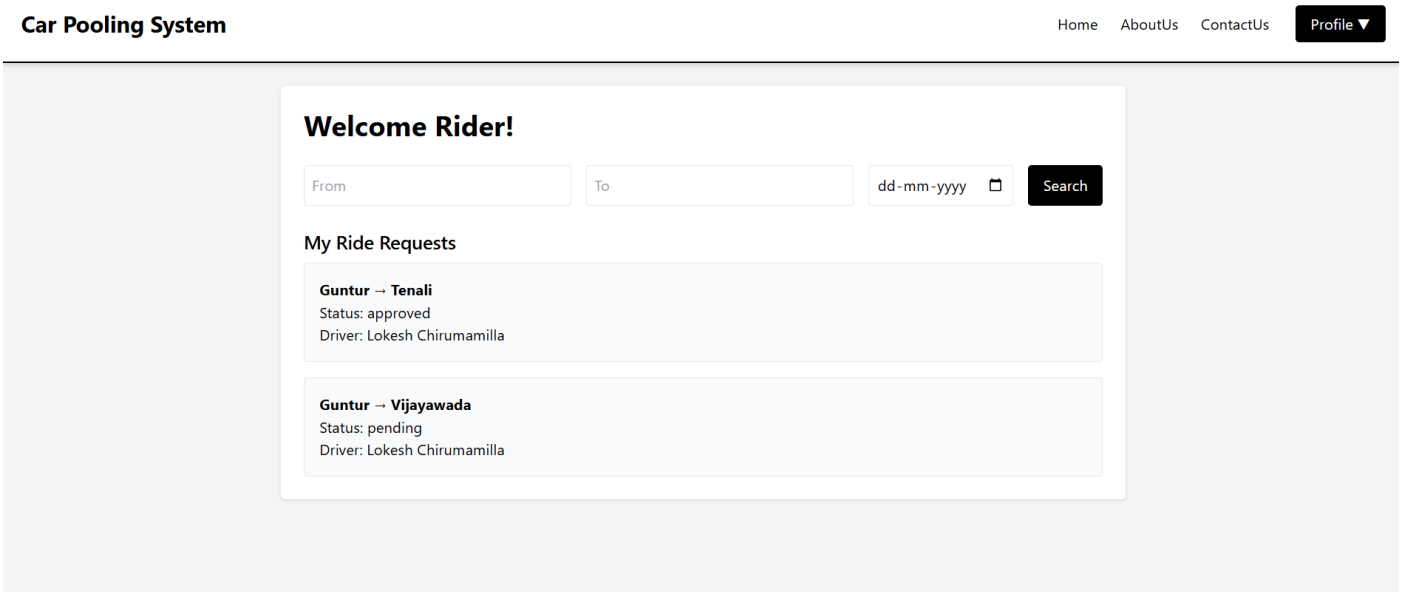

Figure 4: Rider Home After login

2. User Profile

- Emergency contact details
- Ride preferences (music, pets, smoking)
- Contact privacy settings

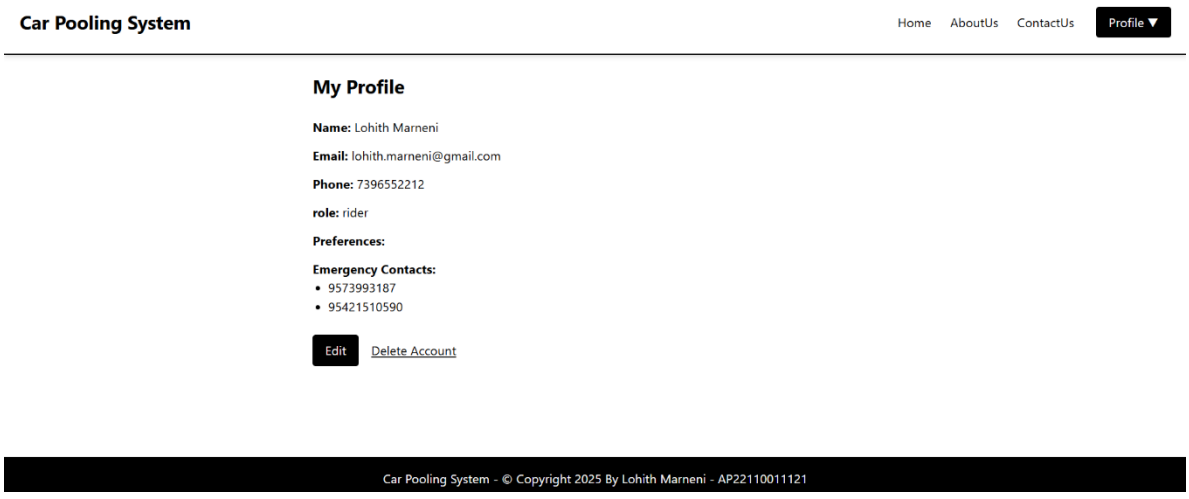

Figure 5: User Profile Update, Delete, Get

3. Driver Functionalities

- Create, edit, and delete rides
- Set vehicle details, preferences, rules
- View incoming ride requests
- Approve or reject ride requests

- View ride history

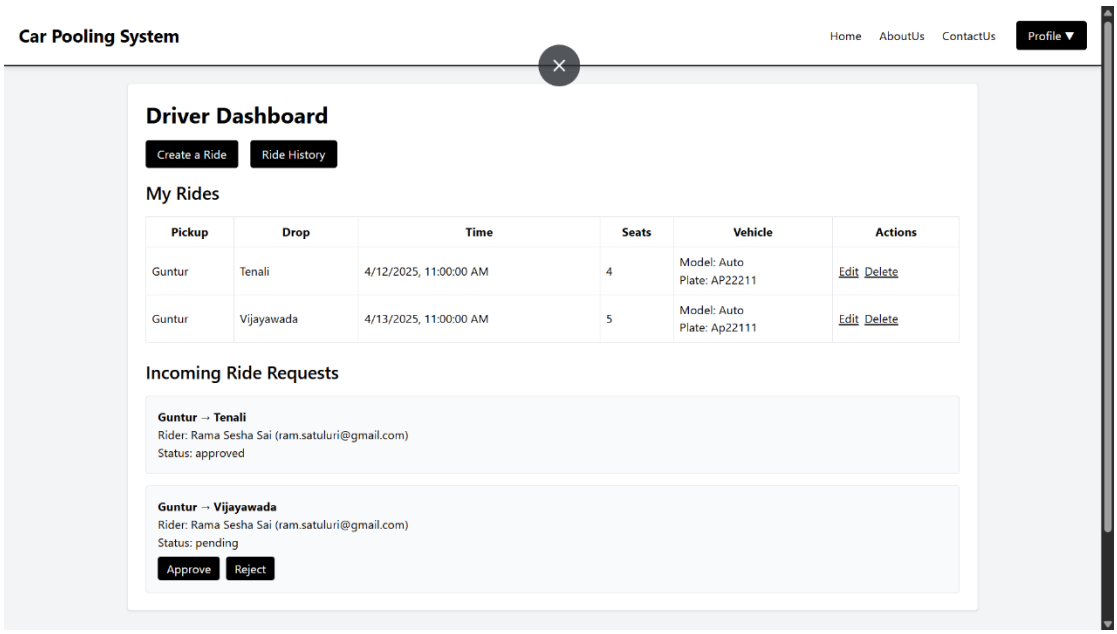

Figure 6: Driver Dashboard

4. Rider Functionalities

- Search rides by pickup, drop, and date
- View ride details and match percentage
- Request to join rides
- View request status

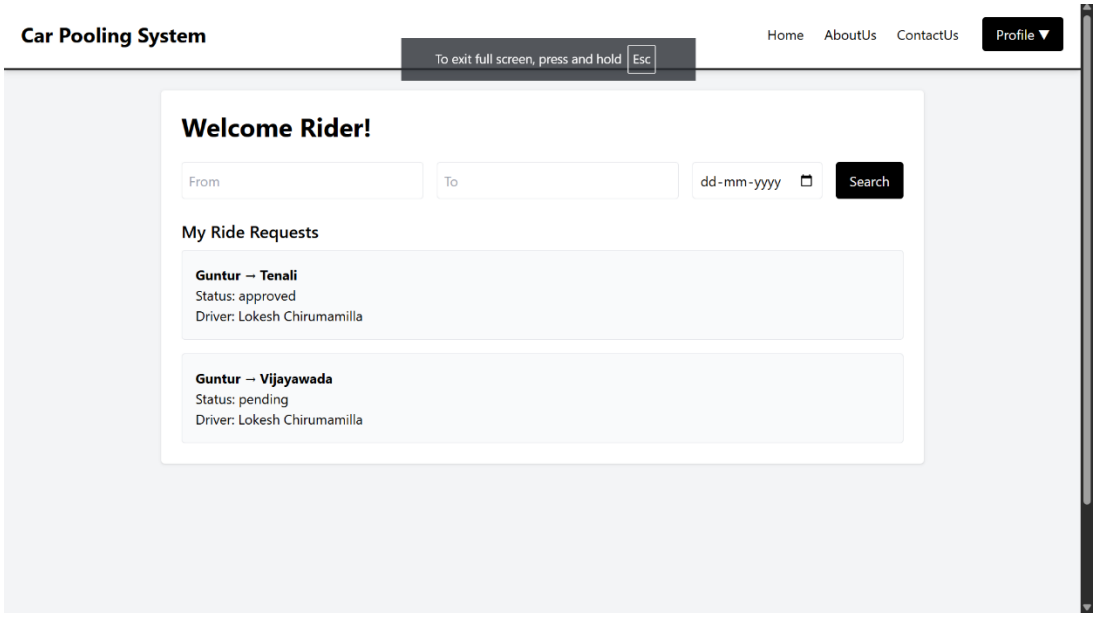

Figure 7: Rider Functionalities

System Architecture

1)

Frontend (React.js)

↕ Axios

Backend (Express.js)

↕ Mongoose

Database (MongoDB)

2)

+ Authentication: JWT + Refresh Token Flow

+ Role Management: Rider vs Driver separation

+ Middleware: Token Validation, Error Handling

Implementation Highlights

Criteria	Implementation
Authentication	JWT Auth with refresh token strategy
Time/Space Optimization	Efficient queries using Mongoose indexing and lean fetches
Fault Tolerance	Centralized error handler, async/await with try-catch
OOP Principles	Models for Users, Rides, RideRequests encapsulated in Mongoose
Caching	Potential Redis caching for hot data (future)
Monitoring	Logs with timestamps for request tracing
Error Handling	Clear responses with meaningful HTTP status codes

User Roles and Interface

Driver UI

- "Create Ride" form
- "My Rides" section to track past and upcoming rides
- "Incoming Requests" list with approve/reject options

Rider UI

- Search bar with filters (timing, gender, preferences)
- “Request Ride” button with match percentage
- “My Requests” page for status tracking

Security & Privacy

- All user passwords hashed using bcryptjs
- Access tokens and refresh tokens used for session security
- Phone numbers and full names hidden unless necessary
- Ride requests only processed after explicit approval

Setup Instructions

Backend

```
cd backend
npm install
# .env file
PORT=5000
MONGO_URI=<your_mongo_uri>
ACCESS_TOKEN_SECRET=your_secret
REFRESH_TOKEN_SECRET=your_secret
npm run dev
```

Frontend

```
cd frontend
npm install
npm run dev
```

System Design Decisions & Trade-offs

Decision	Trade-off
Manual ride approval	Sacrifices speed for control and safety
Separate models for Ride and RideRequest	Slight complexity increase, but improves data integrity
Match score instead of auto-matching	Gives user freedom over AI-based automation
No external API for location	Simpler development; may affect route accuracy (for now)

Future Enhancements

- Real-time notifications via WebSocket
- Integration with Google Maps for route validation
- Live location tracking with SOS
- AI-based song/music preference during rides
- Shareable invite links via WhatsApp/email

Appendix: Technologies Used

Layer	Tools
Frontend	React.js, Tailwind CSS, Axios, Vite
Backend	Node.js, Express.js, Mongoose
Authentication	JWT, bcrypt.js
Database	MongoDB