

Back End Engineering-I

Project Report

Semester-IV (Batch-2023)

Cabbiez



Supervised by :

Dr. Kiran Deep Singh

Associate Professor

Submitted by:

Aniket Singla, 2310992100, G24

Ankit Bansal, 2310992106, G24

Ketan Sharma, 2310992120, G24

Vihaan Chawla, 2310991319, G24

**Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab**

ABSTRACT

Cabbiez is a full stack web application built using Node.js, Express.js, EJS, and MongoDB Compass, designed to simplify urban transportation through an efficient cab booking system. The platform serves both riders, enabling quick ride bookings, real-time cab tracking, and seamless ride management via a user-friendly interface. It features secure user authentication, CRUD operations for rides and profiles, and responsive design for accessibility across devices. Leveraging MongoDB for scalable data storage and Express middleware for efficient request handling. Cabbiez delivers fast, reliable performance. This project addresses key challenges in urban mobility by providing a centralized, scalable, and accessible solution, demonstrating essential web development and backend integration concepts while offering a strong foundation for future enhancements such as in-app payments, real-time notifications, and advanced route optimization.

TABLE OF CONTENTS

	Page No
Abstract	i
Table of Contents	ii
List of Figures	iii
1 Introduction	1
2 Problem Definition and Requirements	3
3 Proposed Design / Methodology	5
4 Results	10
5 Conclusion	13
References	14

LIST OF FIGURES

Figure No.	Figure Caption	Page No.
Fig. 4.1	Sign In Page	10
Fig. 4.2	Sign Up Page	10
Fig. 4.3	Home Page	11
Fig. 4.4	Book A Cab	11
Fig. 4.5	Check Fair and All Users	12
Fig. 4.6	Cab History	12

1. Introduction

1.1 Background

In today's digital era, the need for fast and reliable transportation has driven the rise of online cab booking platforms. Traditional methods of hailing cabs are being replaced by digital solutions that offer real-time tracking, transparent pricing, and ease of use.

Cabbiez is a modern web based cab booking platform designed to meet these expectations. It allows users to book rides, track cabs in real time, view available drivers, and get fare estimates through a simple, intuitive interface. Drivers can easily manage ride requests, navigate routes, and monitor earnings.

Built with Node.js, Express.js, MongoDB, and EJS, Cabbiez offers a responsive, scalable, and maintainable solution tailored to today's urban mobility needs.

1.2 Objectives

The key objectives of the Cabbiez cab booking platform are:

1. To build a full-stack web application that allows users to book cabs and drivers to manage ride requests through a streamlined interface.
2. To implement a robust backend using Express.js, managing routes for booking, user management, and ride tracking efficiently.
3. To integrate middleware for handling authentication, input validation, and error management to ensure secure and smooth operation.
4. To design a responsive frontend using EJS templates and modern CSS, offering an intuitive experience across devices.
5. To enable data persistence through MongoDB Compass, storing information such as user profiles, bookings, and ride history.
6. To utilize Node.js's non-blocking architecture for handling concurrent requests, ensuring the system remains fast and scalable under load.

1.3 Significance

The significance of the Cabbiez project extends beyond its practical use in urban transportation:

1. **Industry Relevance:** Cabbiez addresses a real-world need by offering a digital solution for quick, reliable, and convenient cab booking.
2. **Technical Implementation:** The project showcases the effective use of modern web technologies, including Node.js, Express.js, and MongoDB, in building scalable applications.
3. **Scalability:** With MongoDB Compass, the system can handle increasing user activity and data without performance degradation.
4. **User Experience:** A responsive, intuitive interface ensures smooth usability across devices, enhancing user satisfaction.
5. **Educational Value:** Cabbiez demonstrates core full stack development principles in a real-world context, making it a strong learning project for aspiring developers.

1.4 Scope

The Cabbiez project covers the development of a responsive web-based cab booking system with the following features:

1. User and driver authentication with profile management.
2. Ride booking, tracking, and history for users.
3. Backend API development using Express.js.
4. Data storage and management using MongoDB Compass.
5. Responsive UI built with EJS and CSS.
6. Basic error handling and user feedback support.

Designed for browser access, the system focuses on meeting urban transportation needs, especially within the Indian market.

2. Problem Definition and Requirements

2.1 Problem Statement

Urban commuters and drivers face several challenges in traditional and existing cab booking systems:

1. **Availability:** Users often struggle to find nearby cabs quickly, especially during peak hours.
2. **User Experience:** Many platforms have cluttered interfaces that hinder smooth booking and ride management.
3. **Accessibility:** Not all services are optimized for use across different devices and screen sizes.
4. **Data Handling:** Managing ride history, driver profiles, and real-time availability requires efficient backend systems.

Cabbiez aims to address these issues by offering a centralized, real-time, and user-friendly web platform tailored for the Indian urban mobility market.

2.2 Software Requirements

The development of Cabbiez uses the following core software components:

1. **Node.js** – Server-side JavaScript runtime.
2. **Express.js** – Web framework for routing and middleware.
3. **EJS** – Templating engine for dynamic HTML generation.
4. **MongoDB & MongoDB Compass** – NoSQL database and its cloud service for storing users, drivers, and booking data.
5. **Key npm packages:**
 - body-parser – Parses incoming request bodies.
 - mongoose – MongoDB object modeling.

- CORS – Allows secure cross-origin communication between the frontend and backend.
- Express – Handles backend routing and request processing in a lightweight Node.js framework
- NodeMailer – Enables the application to send emails such as confirmations or notifications.

2.3 Data Processing Requirements

The Cabbiez system must efficiently manage the following data processes:

1. **User Authentication:** Secure login and registration for riders and drivers.
2. **Session Management:** Maintaining active user sessions during interactions.
3. **CRUD Operations:** Handling creation, retrieval, updates, and deletion of users, drivers, and ride bookings.
4. **Search Functionality:** Efficient querying of available cabs based on location and availability.
5. **Data Validation:** Ensuring accuracy and validity of user inputs like booking details.

2.4 Performance Requirements

Given the real-time and interactive nature of cab booking, Cabbiez must meet these performance criteria:

1. **Response Time:** Pages and ride updates should load within 2-3 seconds under normal load.
2. **Scalability:** The system must handle high user traffic during peak hours smoothly.
3. **Concurrency:** Support multiple simultaneous users, including riders and drivers.
4. **Availability:** Ensure high uptime with minimal downtime or interruptions.
5. **Mobile Responsiveness:** Interface should adapt seamlessly to various devices and screen sizes.

3. Proposed Design / Methodology

3.1 System Architecture

Cabbiez follows the Model-View-Controller (MVC) architecture, ensuring a clear separation of concerns:

1. **Model:** MongoDB schemas defining users, drivers, rides, and bookings.
2. **View:** EJS templates rendering responsive user interfaces.
3. **Controller:** Express.js route handlers managing requests and application logic.

Key Components:

- **Client-Side Interface:** EJS with CSS for responsive design.
- **Express Application Core:** Main server setup and configuration.
- **Routing Layer:** Routes directing requests to controllers.
- **Middleware:** Handles authentication, request parsing, and error management.
- **Controllers:** Implements business logic for ride booking and management.
- **Data Models:** MongoDB schemas modeling data entities.
- **Database Layer:** Connection and operations with MongoDB Compass.

3.2 Express.js Routing Implementation

Routing in Cabbiez uses Express.js to handle HTTP requests efficiently:

1. **Route Methods:**
 - **GET:** Retrieve available cabs, ride status, and user profiles.
 - **POST:** Create new ride bookings and register users.
 - **PUT:** Update ride details or user profiles.
 - **DELETE:** Cancel bookings or remove user data.

2. **Route Paths:** Clear URL structures for resources such as:
 - /rides — Ride-related operations
 - /auth — User authentication
 - /drivers — Driver-specific features
 - /profile — User and driver profile management
3. **Route Parameters:** Dynamic URL segments for resource identification, e.g.,
 - /rides/:id — Access specific ride details
 - /drivers/:id/accept — Driver accepting a ride request
4. **Route Handlers:** Controller functions implementing the business logic for each route.

3.3 Middleware Implementation

Cabbiez uses middleware to manage various tasks efficiently:

1. **Application-Level Middleware:**
 - Parsing request bodies with body-parser
 - Managing user sessions via express-session
 - Serving static files using express.static
2. **Router-Level Middleware:**
 - Authentication and authorization checks
 - Input validation for secure and accurate data
 - Route-specific preprocessing as needed
3. **Error-Handling Middleware:**
 - Custom handlers for different error types
 - User-friendly error messages
 - Logging errors for troubleshooting

4. **Third-Party Middleware:**

- MongoDB session store using connect-mongodb-session
- Email service integration using NodeMailer for sending confirmations.

3.4 Request Processing Flow

The request lifecycle in Cabbiez using Express.js follows these steps:

1. **Request Reception:** The server receives an incoming HTTP request.
2. **Middleware Processing:** The request passes through middleware for tasks like parsing, authentication, and validation.
3. **Route Matching:** Express matches the request URL and method to the correct route handler.
4. **Controller Execution:** The handler processes the request, interacting with data models as needed.
5. **View Rendering:** Dynamic pages are rendered using EJS templates with the processed data.
6. **Response Generation:** The final response is sent back to the client.

This flow ensures organized and efficient handling of all user actions.

3.5 Non-Blocking Code Implementation

The Cabbiez Cab Booking Platform uses Node.js's non-blocking, event-driven architecture to ensure high performance and responsiveness:

1. **Asynchronous Database Operations:** MongoDB operations like ride bookings, user data, and trip history are handled using `async/await` or promises to prevent blocking the main thread.
2. **Concurrent Request Handling:** The system handles multiple ride requests, and fare calculations in parallel without blocking the event loop.

3. **Callback Patterns:** Asynchronous callbacks and event listeners are used where needed helping maintain real-time interactivity.

This approach keeps cabbiez responsive and scalable even during peak traffic hours..

3.6 Database Design

The Cabbiez cab booking system uses both a local and MongoDB database for efficient data handling:

Local Database:

Stores essential user credentials for quick access and authentication:

Fields: name, email, password, phone

MongoDB Collections:

1) Users :-

Stores complete user profiles.

Fields: name, age, email, phone, gender, password

2) CabDetails :-

Stores ride booking and cab information.

Fields: cabName, cabPrice, cabType, userEmail, bookingTime, cabSeats.

3.7 User Interface Design

The user interface is designed with a focus on usability and responsiveness:

1. **Responsive Layout:** Adapts to various screen sizes using CSS media queries.
2. **Interactive Elements:** Includes updates via mail and dynamic content.
3. **Form Validation:** Client-side validation complements server side checks.
4. **Feedback Mechanisms:** Clear notifications for successful operations and errors.

3.8 File Structure

```
EVALUATION1/  
├─ node_modules/  
├─ data.ejs  
├─ index.js  
├─ package-lock.json  
├─ package.json  
├─ register.ejs  
├─ script.js  
├─ server.js  
├─ styles.css  
└─ User.json
```

Evaluation 1 File Structure

```
project-root/  
├─ .idea/  
├─ client/  
│   ├── .vercel/  
│   ├── node_modules/  
│   ├── public/  
│   │   ├── favicon.ico  
│   │   ├── index.html  
│   │   ├── logo192.png  
│   │   ├── logo512.png  
│   │   ├── manifest.json  
│   │   └─ robots.txt  
│   └─ src/  
│       ├── components/  
│       ├── App.js  
│       ├── App.module.css  
│       ├── App.test.js  
│       ├── index.css  
│       ├── reportWebVitals.js  
│       └─ setupTests.js  
├─ .gitignore  
├─ package-lock.json  
├─ package.json  
├─ README.md  
└─ server/  
    ├── connect/  
    ├── models/  
    ├── node_modules/  
    ├── routes/  
    ├── .env  
    ├── .gitignore  
    ├── index.js  
    ├── package-lock.json  
    ├── package.json  
    ├── README.md  
    ├── ShortestPath.py  
    ├── vercel.json  
    └─ Cab System scaler.docx
```

Cab-Booking File Structure

4. Results

The Cabbiez platform successfully allows users to book cabs, view ride details, and track bookings in real-time through a responsive and user-friendly interface. It showcases efficient use of full-stack technologies and secure, streamlined ride management.

4.1 Sign In Page

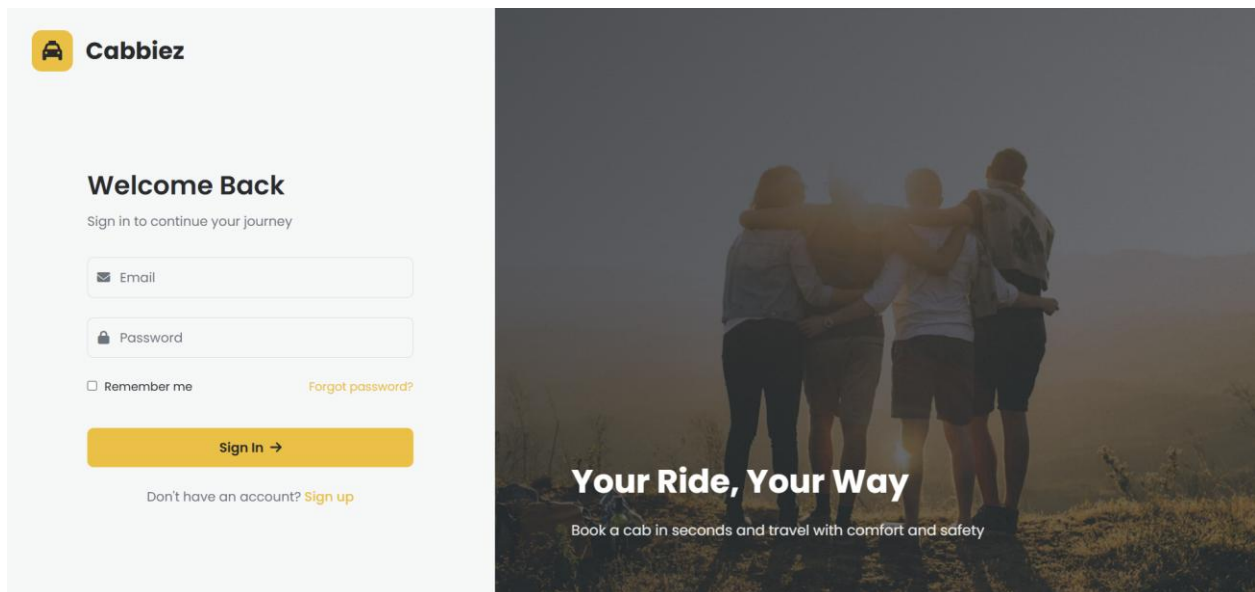


Fig 4.1 Sign In Page

4.2 Sign Up Page

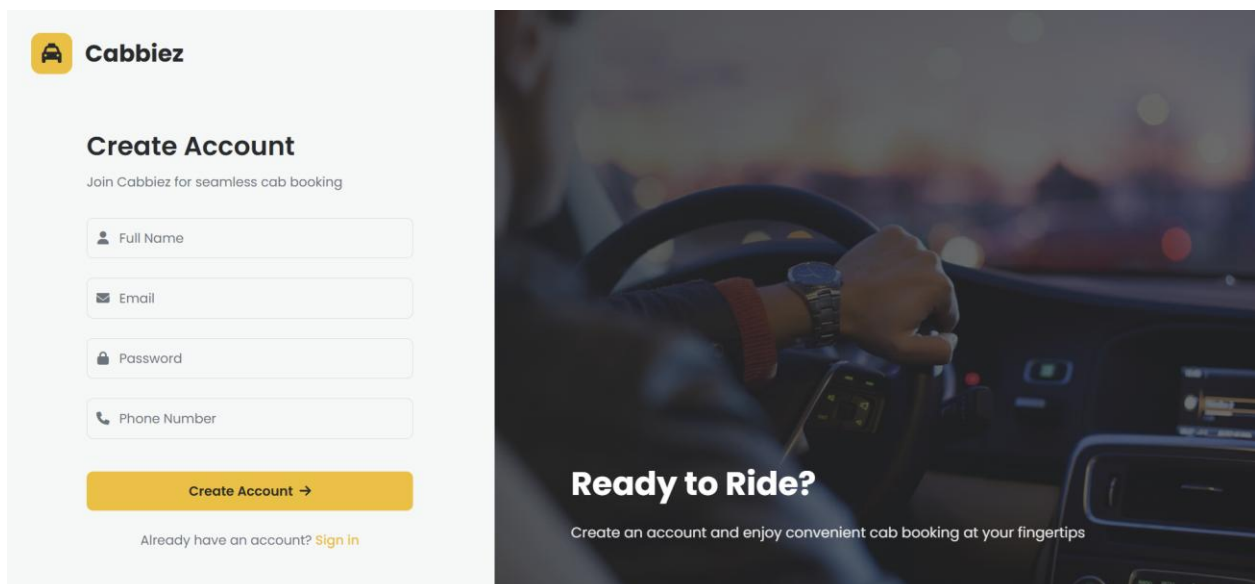


Fig 4.2 Sign Up Page

4.3 Home Page

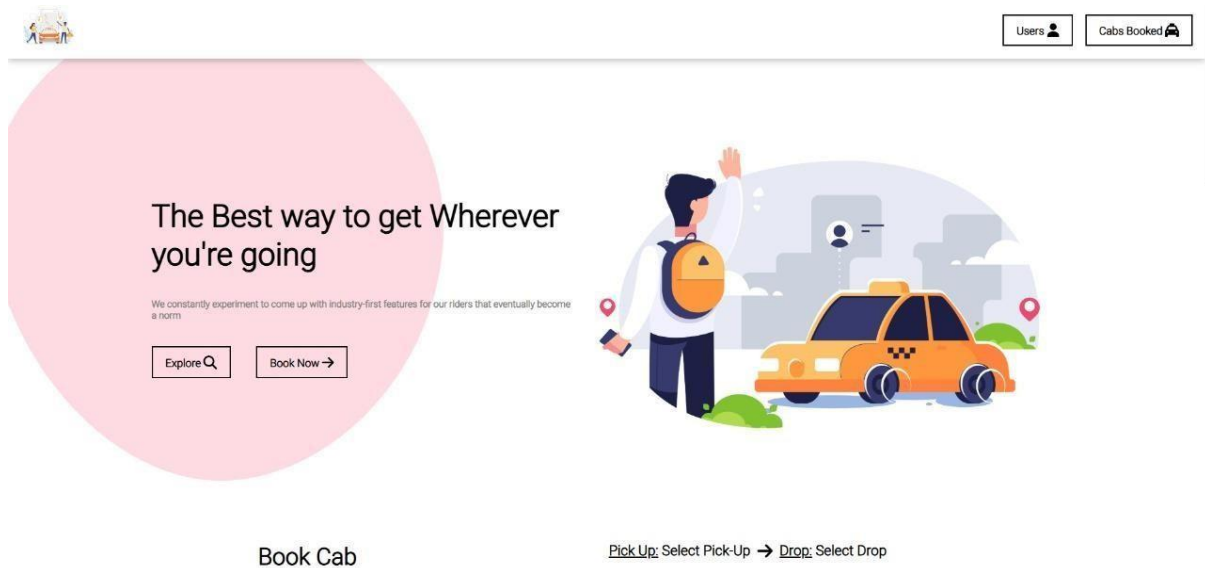


Fig 4.3 Cabbiez Home Page

4.4 Book A Cab

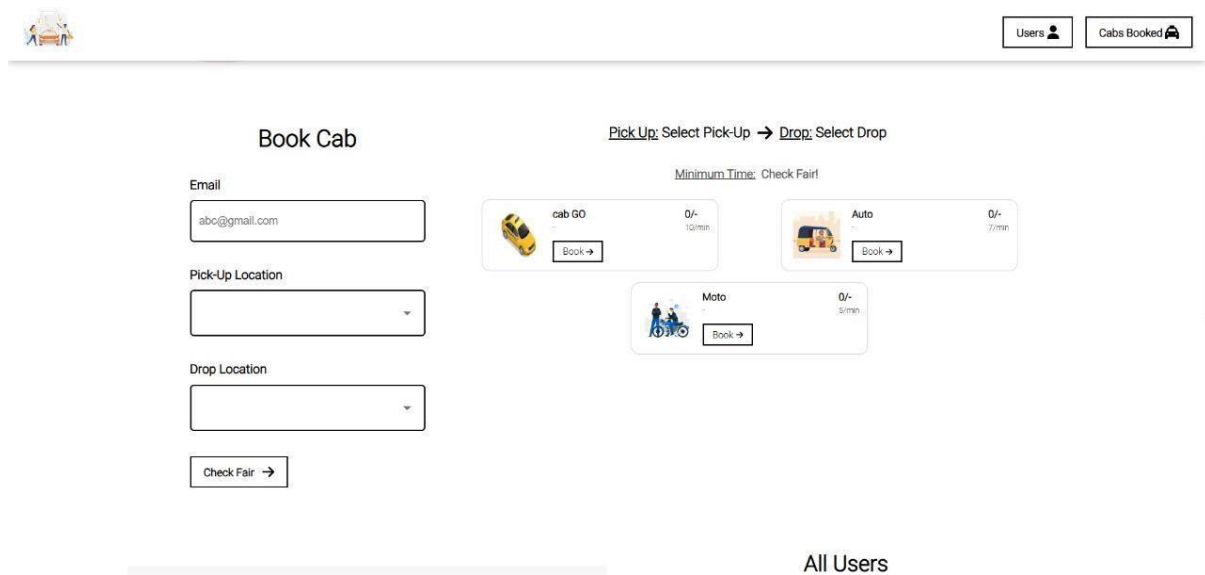





Fig 4.4 Book A Cab


4.5 Check Fair and All Users



Users 

Cabs Booked 

Check Fair →





All Users


SNo.	User Email	Bookings
→	adityakshk499@gmail.com	1
→	aniketsingla05@gmail.com	1
→	abhi19ay@gmail.com	1
→	bansalankit1575@gmail.com	1
→	ketan2120.be23@chitkara.edu.in	1

Fig 4.5 Check Fair and All Users

4.6 Cab History



Users 

Cabs Booked 

All Cabs

SNo.	Cab Name	Email
94456	cab GO	adityakshk499@gmail.com
d6b44	cab GO	aniketsingla05@gmail.com
d6ad5	cab GO	abhi19ay@gmail.com
acd21	cab GO	bansalankit1575@gmail.com
763ed	Moto	ketan2120.be23@chitkara.edu.in
76116	cash GO	animeshsharma42002@gmail.com




Fig 4.6 Cab History

5. Conclusion

The Cabbiez web application showcases a fully functional full-stack solution built using Node.js, Express.js, EJS, and MongoDB. It addresses common challenges in online cab booking by offering a smooth, user-friendly platform for booking rides efficiently.

Key achievements of the project include:

1. **Robust Backend:** Developed using Express.js, the backend ensures efficient routing, middleware use, and structured error handling for smooth server-side operations.
2. **Responsive Frontend:** EJS templates with modern CSS create an intuitive and adaptive user interface.
3. **Data Persistence:** Integration with MongoDB Compass ensures secure and scalable data storage for user profiles, cab details, and bookings.
4. **User Experience:** Both riders and administrators are offered tailored features such as cab selection, real-time booking status, and ride history, improving usability and accessibility.
5. **Scalable Architecture:** Leveraging Node.js's non-blocking I/O and modular code structure, the system is ready for scaling and future feature integration.

Future Enhancements:

- Real-time cab tracking using Web Sockets
- Driver-side dashboard with location and ride status
- Mobile app version for on-the-go booking
- Payment gateway integration for fare processing
- Admin analytics for trip monitoring and user insights

The Cabbiez project stands as a practical, scalable solution in the transportation sector, while also exemplifying modern web development techniques, including modular Express routing, EJS templating, and efficient use of MongoDB.

References

Express.js Documentation. (2023). Express.js Guide.

<https://expressjs.com/en/guide/routing.html>

MongoDB Documentation. (2023). MongoDB Manual.

<https://docs.mongodb.com/manual/>

EJS Documentation. (2023). Embedded JavaScript templating. <https://ejs.co/#docs>

Node.js Documentation. (2023). Node.js v16 Documentation.

<https://nodejs.org/docs/latest-v16.x/api/>

Brown, E. (2019). Web Development with Node and Express: Leveraging the JavaScript Stack. O'Reilly Media.

Dayley, B. (2018). Node.js, MongoDB and Angular Web Development. Addison-Wesley Professional.

Stack Overflow Community-driven solutions and troubleshooting tips for Node.js, Express, and MongoDB issues.

<https://stackoverflow.com/>