UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FAKULTI TEKNOLOGI KEJURUTERAAN ELEKTRONIK DAN KOMPUTER
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ASSIGNMENT : MYTAXI SYSTEM
BERR 2243 : DATABASE AND CLOUD SYSTEM

LECTURER'S NAME :
PROFESSOR MADYA DR SOO YEW GUAN

	NAME	MATRIC NUMBER
1	FAREZ AMINUDDIN BIN MOHAMAD JOHARI	B122310574
2	GRAYSON GAN KA SENG	B122310664
3	MUHAMMAD SYAZWAN BIN MOHD ZURKHI	B122310155

Group Members: BENR2243 DATABASE AND CLOUD SYSTEM

ASSIGNMENT PROJECT EVALUATION FORM

Student Name	Student Number	Signature
FAREZ AMINUDDIN BIN MOHAMAD JOHARI	B122310574	
GRAYSON GAN KA SENG	B122310664	
MUHAMMAD SYAZWAN BIN MOHD ZURKHI	B122310155	

	Rubrics					CLO/PO	Marks
Cloud and database design	A complete application with clear and precise comment, main program functions were designed and documented.	A good application developed with most of the main program functions has been presented.	An average application with lacking few important API. Partial implementation of application security.	A basic application with only few APIs. No consideration on the application security.	No submission	CLO2/PO3	/10
	4	3	2	1	0		
Project Functionality	The system is hosted on the cloud and working properly. Detail development progress, track records and team works can be seen on Github. All test procedures are provided in Postman.	The design of system is carried out, but partially working. Important development details can be seen on Github, but lacking team working records. Most test procedures are provided in Postman.	The design of system is carried out, but partially working. Not much development progress can be seen on Github. No procedures are provided.	The design of system is carried out, but not working accordingly. No development progress can be seen on Github. No procedures are provided.	No submission	CLO3/PO5	/10
	4	3	2	1	0		
Total (x/20)							

1.0 INTRODUCTION

The rapid advancement of digital technologies has significantly transformed the way transportation services operate today. Ride-hailing platforms such as Grab and Uber have reshaped commuting by providing users with seamless, on-demand mobility solutions. Taking inspiration from these services, MyTaxi has been developed as a campus-oriented ride-booking system aimed at improving transportation for students at Universiti Teknikal Malaysia Melaka (UteM).

MyTaxi offers a convenient platform where students can easily request rides using a mobile or online interface. What makes this system unique is its dual functionality: students can sign up either as passengers or as drivers. This not only enhances accessibility across the campus but also allows student drivers to earn additional income by offering transportation to their peers. The platform thus encourages a collaborative and self-sustaining environment within the university community.

Transportation challenges are common in university settings, especially where public transit options may be limited or inefficient. MyTaxi addresses these issues by providing an accessible and reliable ride-booking solution. Whether it's getting to class, heading to the library, or traveling to nearby facilities, the system enables students to manage their mobility more effectively, reducing the inconvenience of long walks or unreliable transport.

Beyond solving practical commuting needs, MyTaxi represents a step forward in embedding digital solutions into daily student life. By promoting student-to-student interaction through a trusted platform, it helps build a stronger sense of community and encourages responsible use of technology. The system is also aligned with the broader trend of digital integration in education, showcasing how smart services can support and enhance the student experience.

Overall, MyTaxi is more than just a transportation service it is a digital initiative that reflects the evolving needs of a connected campus. It introduces a modern solution to everyday student challenges, fostering both convenience and economic opportunity within a familiar, student-driven ecosystem.

2.0 OBJECTIVES

The main goal of this report is to explore the development and functionality of a campus-based ride-booking platform, MyTaxi, designed to serve students at Universiti Teknikal Malaysia Melaka (UTeM). The system aims to enhance mobility, promote peer-to-peer transportation, and introduce a digital solution to common commuting challenges faced by students. This report is structured around three key objectives :

1. **To understand the conceptual design of a ride-booking platform and its relevance in a campus environment.**

   This involves studying how a digital ride-hailing system can be adapted to a university setting where both drivers and passengers are students. It also includes identifying the essential features such as ride requests, ride status tracking, and user interaction, which form the foundation of the MyTaxi platform.

2. **To design and model a secure and scalable backend architecture that supports core functionalities of the MyTaxi platform.**

   This includes the development of a structured data model to handle user accounts, ride information, and real-time booking processes. Special attention is given to implementing secure authentication practices and creating a system that supports efficient data retrieval and management.

3. **To demonstrate system functionality through implementation of essential backend services, enabling interaction between clients and the database.**

   This involves the deployment of RESTful APIs that allow students to register, log in, request rides, and respond to ride requests. The objective also covers the testing and simulation of various user scenarios to verify the system's ability to manage real-time booking operations and reflect updates consistently across all users.

Through these objectives, the project aims to showcase a practical and meaningful application of backend development skills in solving real-world issues. MyTaxi not only introduces a convenient transportation solution but also represents how digital systems can foster community-driven services within academic environments.

3.0 METHODOLOGY

The development of the MyTaxi system followed a systematic and iterative process that incorporated a range of modern tools and technologies. This methodology was designed to ensure that the backend system would be secure, reliable, and responsive to the functional demands of a real-time ride-booking application. The workflow was organized into several key stages, each leveraging specific platforms and utilities to support implementation, testing, and version management.

The primary development work was conducted using Visual Studio Code (VS Code), a versatile and lightweight code editor ideal for backend development. Its integrated support for JavaScript and Node.js provided an efficient environment for writing and organizing the server-side code. All backend components including API routing, middleware, and controller logic were developed using the Node.js runtime, selected for its non-blocking, event-driven architecture, which is well-suited for handling multiple user requests simultaneously in a real-time application like *MyTaxi*.

For database management, MongoDB was used as the central storage solution due to its flexible NoSQL architecture, which accommodated the varying structure of ride data, user accounts, driver profiles, and booking statuses. To facilitate easier interaction with the database, MongoDB Compass served as the graphical interface, enabling developers to visualize collections, perform Create, Read, Update and Delete (CRUD) operations, and monitor data in real-time without the need for command-line input. The backend's data models were constructed using Mongoose, an Object Data Modelling (ODM) library that provided a seamless connection between the application and the database via Compass.

Postman played a key role in testing the backend APIs throughout the development process. It was used to send various HTTP requests to endpoints responsible for tasks such as user registration, login, and ride transactions. By simulating client-side interactions, Postman ensured that the server responded accurately and consistently under different test cases. Additionally, Postman collections were used to document and organize the API endpoints for future reference and collaborative testing.

For analyzing communication at the network level, Wireshark was utilized to inspect data packets exchanged between the client and server. This tool was instrumental in debugging issues related to authentication, session handling, and ride-triggering events by identifying anomalies such as packet loss, delays, or improper responses. It provided a deeper understanding of how data traveled across the network, helping to ensure smooth and reliable operation.

GitHub served as the version control and collaboration platform for the project. It hosted the source code, managed development branches, and tracked changes made over time. Regular commits were made to reflect progress, while feature branches helped isolate new functionalities and bug fixes. GitHub also provided a space for maintaining the API documentation and showcasing the evolution of the project.

In summary, the integration of these development tools allowed for the efficient and organized creation of the MyTaxi system. Each technology contributed to a specific phase of the project from code writing and database handling to testing, debugging, and version tracking ensuring that the final product was functional, stable, and ready for deployment.

4.0 SYSTEM OVERVIEW

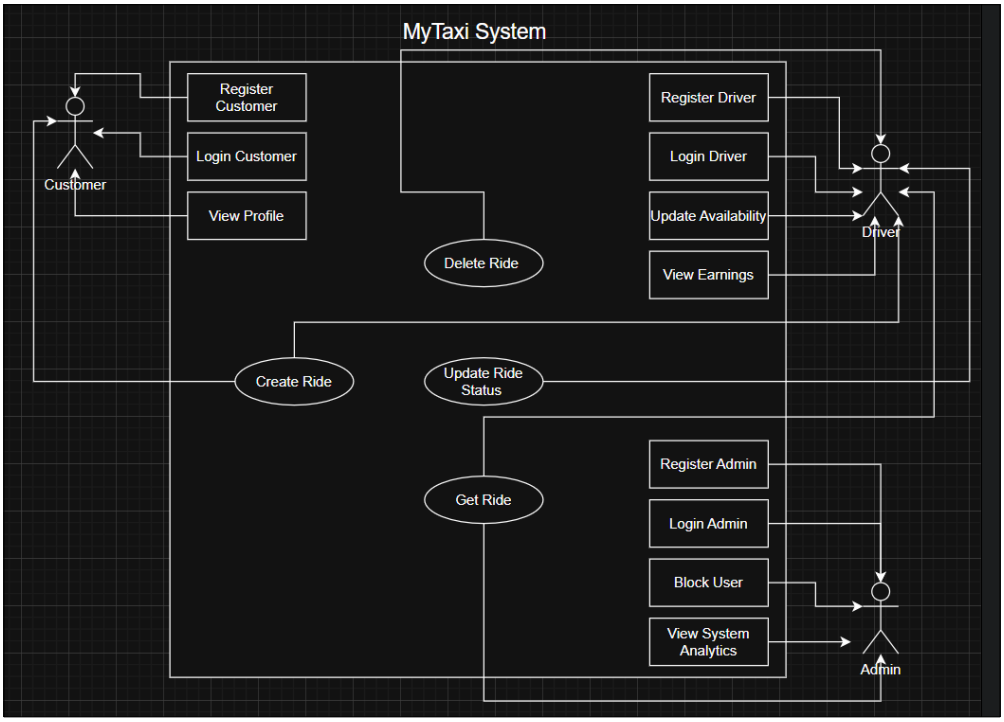

Figure 1 : MyTaxi System Use-Case Diagram

In the MyTaxi System, there are three primary actors: the Passenger, the Driver, and the Admin. The Customer interacts with the system to register, log in, book rides, and track their trips in real time. Meanwhile, the Driver logs into the system to view ride requests, accept bookings, update ride status, and manage their availability. The Admin oversees the entire platform by managing user accounts, and generating operational reports. Each actor has distinct responsibilities, but they're all interconnected to ensure the smooth functioning of the e-hailing service.

5.0 DATABASE MODEL

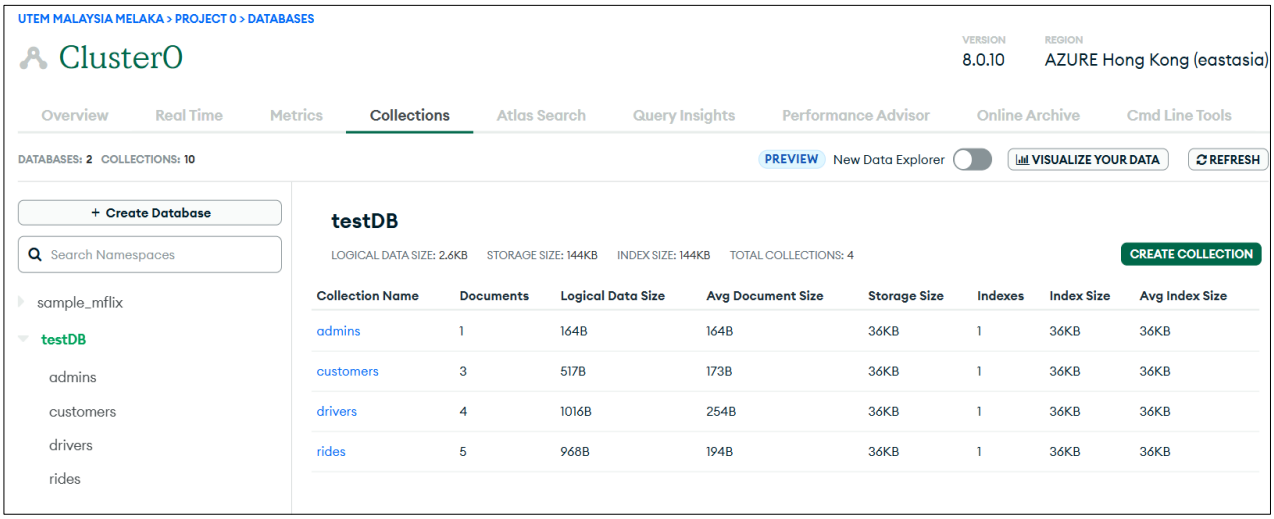

Figure 2 : MyTaxi System Database Collections in MongoDB Atlas

The MongoDB Atlas database used in this MyTaxi backend system consists of multiple collections that organize and manage the application's core data. The key collections include customers, drivers, admins, and rides. The customers collection stores information about regular users, including their name, email, hashed password, and role. The drivers collection holds additional fields specific to drivers, such as vehicle type, license plate number, current availability status, and total earnings. Administrative users are stored in the admins collection, which contains basic authentication information. The rides collection is central to the platform, recording ride transactions with details such as pickup and destination locations, ride status, fare, and references to both the driver and customer involved, stored by their unique IDs. These collections are hosted on MongoDB Atlas, a cloud-based NoSQL database service that provides high availability, scalability, and secure access to the backend system. This structure allows for efficient role-based access and ensures that data operations are performed securely and logically across the application.

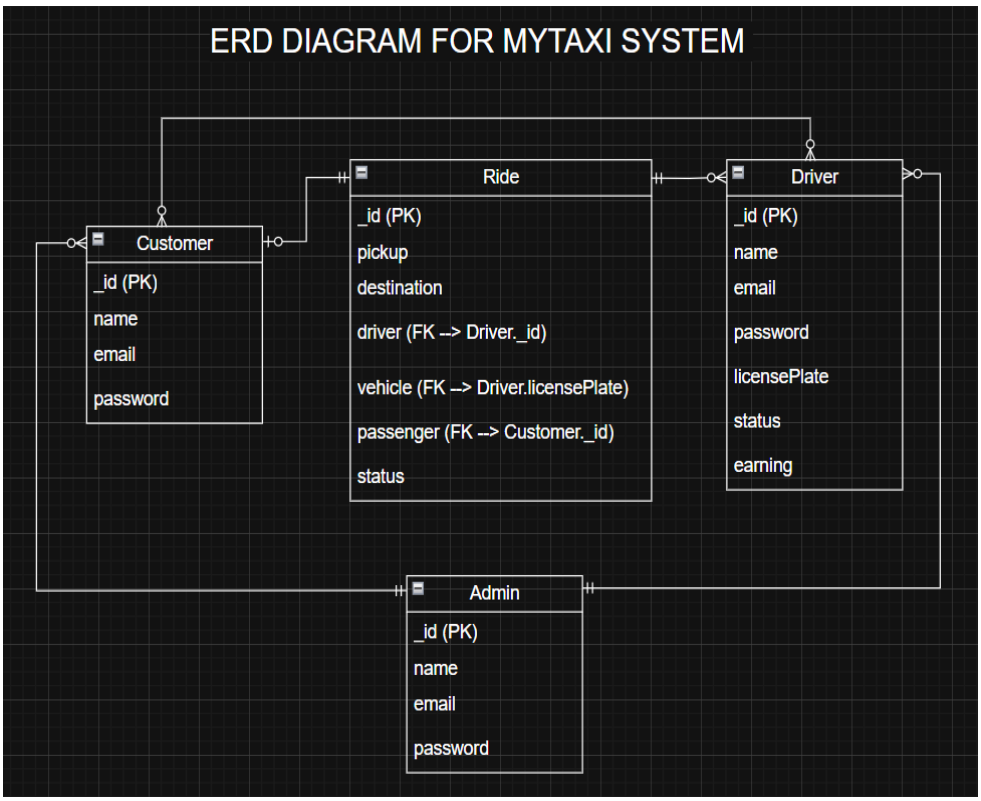

Figure 3 : MyTaxi System ERD Diagram

The Entity Relationship Diagram (ERD) for the MyTaxi system defines four main entities which is Customer, Driver, Ride, and Admin, and maps their relationships using Crow's Foot Notation. The relationship between Customer and Ride is optional one to mandatory one, meaning a customer may have zero or more ride records, but each ride must be associated with one customer. The Driver entity has an optional many to mandatory one relationship with Ride, indicating that while a driver can be linked to many rides, each ride must have exactly one assigned driver. Additionally, the system defines an optional many-to-optional many relationship between Customer and Driver, representing the real-world possibility where multiple customers can interact with multiple drivers over time and vice versa this is logically managed through the shared involvement in the Ride entity. Lastly, the Admin entity has a mandatory one to optional many relationship with both Customer and Driver, reflecting that each customer or driver must be supervised by at least one admin, while an admin can manage many users. This structure allows for flexibility in ride management, user supervision, and system control, all hosted efficiently within MongoDB Atlas.

6.0 BACKEND DEVELOPMENT

Method	Endpoint	Function
ADMIN		
POST	/admins/register	Register Admin
GET	/admins/analytics	Display Customers, Drivers, and Rides
DRIVER		
POST	/drivers/register	Register Driver
GET	/drivers/_id/earnings	Displays Driver's earnings
PATCH	/drivers/_id/status	Update Driver's Availability
CUSTOMER		
POST	/customers/register	Register Customer
GET	/customers/_id	Display Customer details
AUTH (All)		
POST	/auth/login	Login Authorities
RIDE		
POST (Customer)	/rides	Create Ride
GET (All)	/rides	Display Ride's details
PATCH (Driver)	/rides/_id	Update Ride's status

Table 1 : MyTaxi System's backend

The MyTaxi backend system is a secure, role-based RESTful API architecture that manages the operations of a ride-hailing service, supporting three distinct user roles: Admins, Drivers, and Customers. All users authenticate via a centralized login endpoint (POST /auth/login), which validates credentials and issues a JWT (JSON Web Token) for session management and authorization. To enhance security, user passwords are securely hashed using bcrypt before storage, ensuring that raw passwords are never stored in the database and protecting against common vulnerabilities such as brute force or dictionary attacks. Admins can register using (POST /admins/register) and granted privileges to monitor system-wide metrics via (GET /admins/analytics), which provides aggregated data on the number of registered users and total rides. Drivers are registered through (POST /drivers/register), can view their earnings (GET /drivers/_id/earnings), and manage their availability status using (PATCH /drivers/_id/status). Customers can register (POST /customers/register), view their profile (GET /customers/_id), and initiate ride bookings (POST /rides) based on predefined pickup and destination locations as well as its fare on the frontend. Ride data is accessible to all roles through (GET /rides), while drivers can update the ride's progress using (PATCH /rides/_id). The system's architecture enforces role-based access control, integrates real-time ride and earnings tracking, and prioritizes data privacy and operational security across all endpoints making it a scalable and reliable backend for modern e-hailing applications.

7.0 SYSTEM DEMONSTRATION

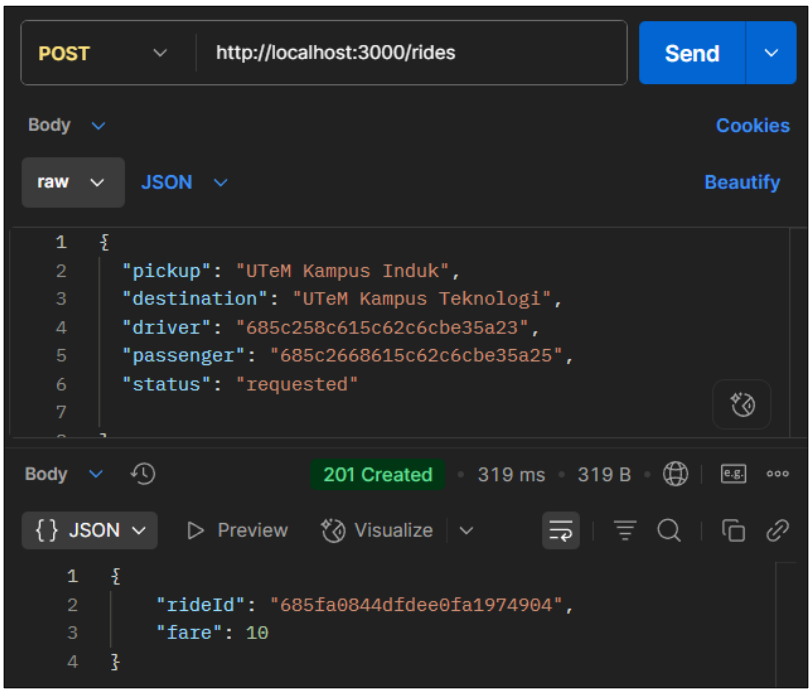

Figure 4 : MyTaxi System : Customer create ride

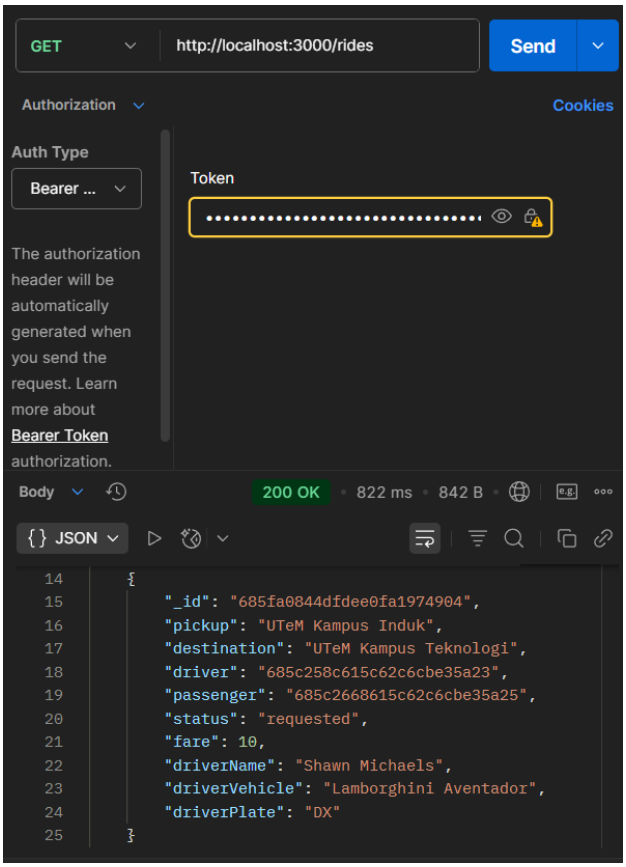

Figure 5 : MyTaxi System : Customer get ride details

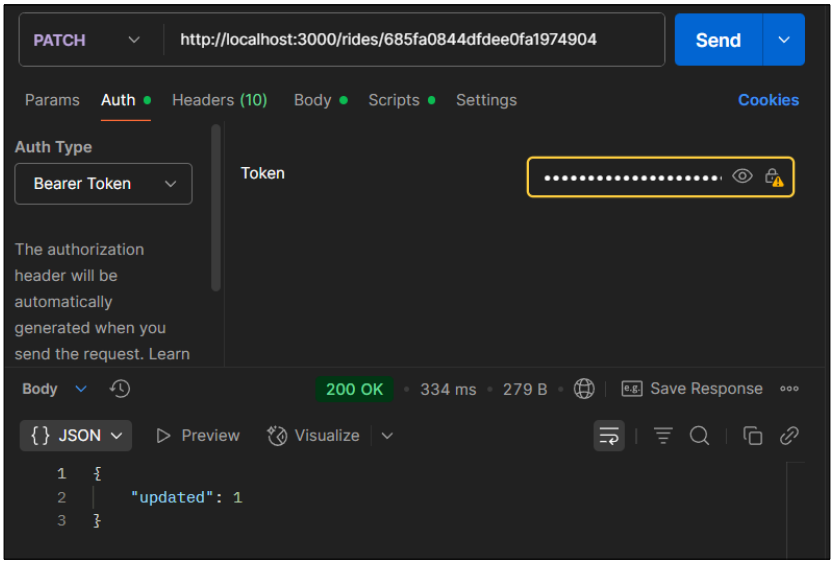

Figure 6 : MyTaxi System : Driver update ride status indicates the ride accepted

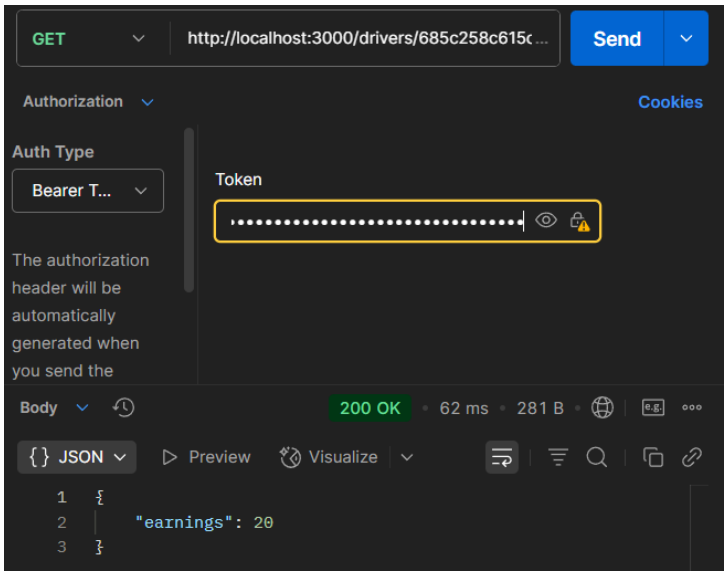

Figure 7 : MyTaxi System : Driver earning's increases based on the previous ride's fare

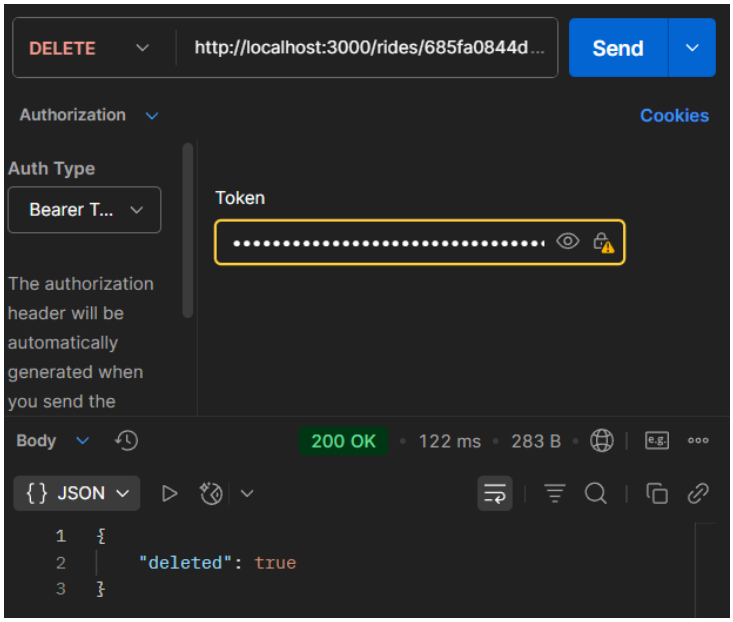

Figure 8 : MyTaxi System : Admin delete rides history

8.0 USER INTERFACE (UI) INTEGRATION

8.1 SYSTEM'S DASHBOARD

Figure 9 : MyTaxi System : Main Dashboard

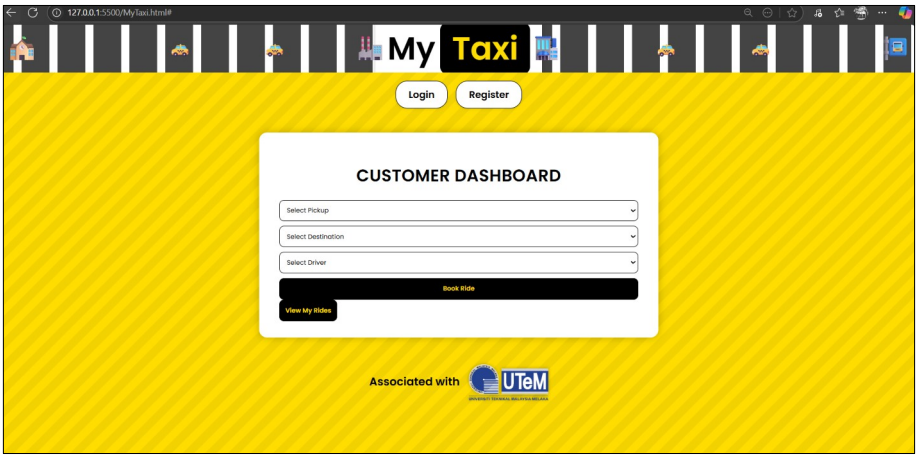

Figure 10 : MyTaxi System : Customer Dashboard

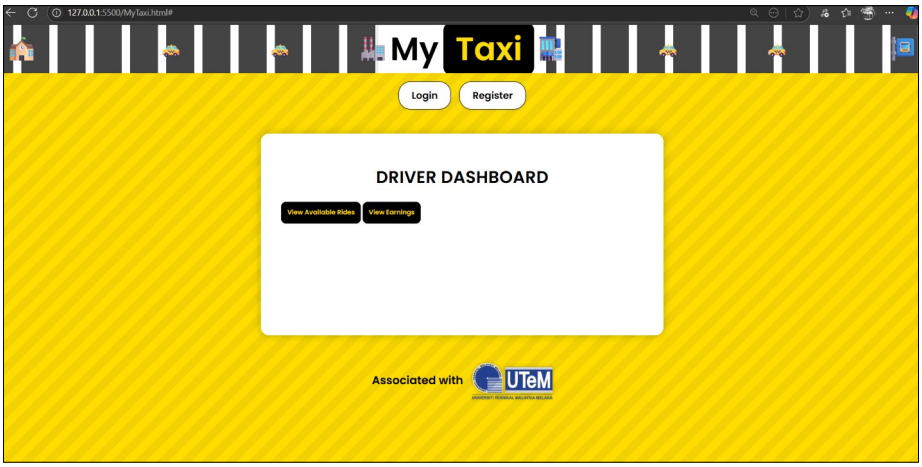

Figure 11 : MyTaxi System : Driver dashboard

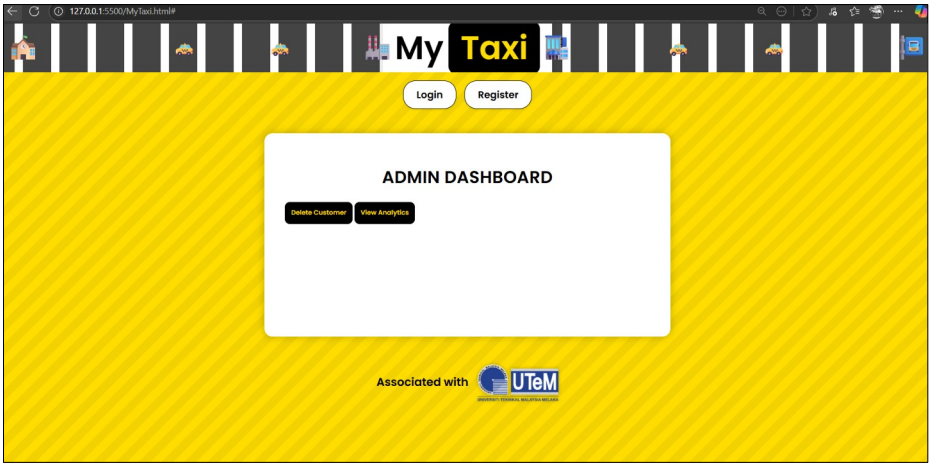

Figure 12 : MyTaxi System : Register/Driver dashboard

8.2 CUSTOMER'S BACKEND

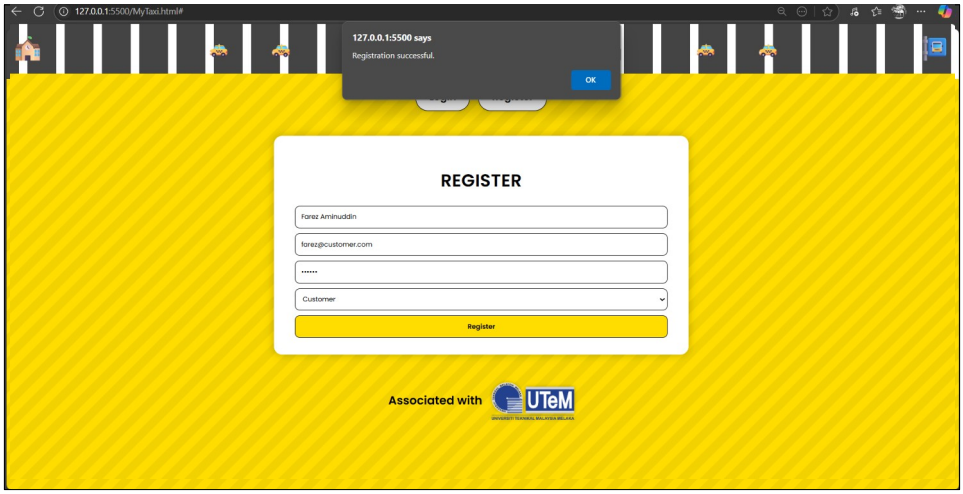

Figure 13 : MyTaxi System : POST customers/register

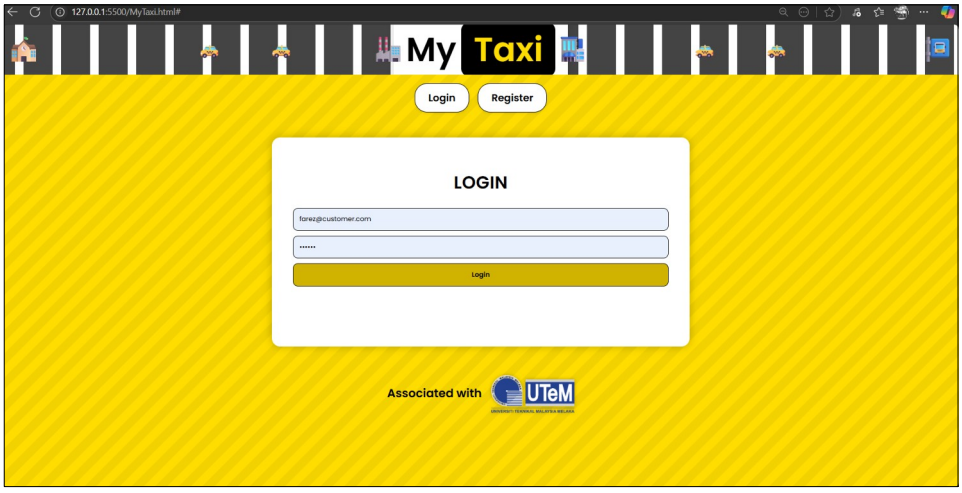

Figure 14 : MyTaxi System : POST auth/login

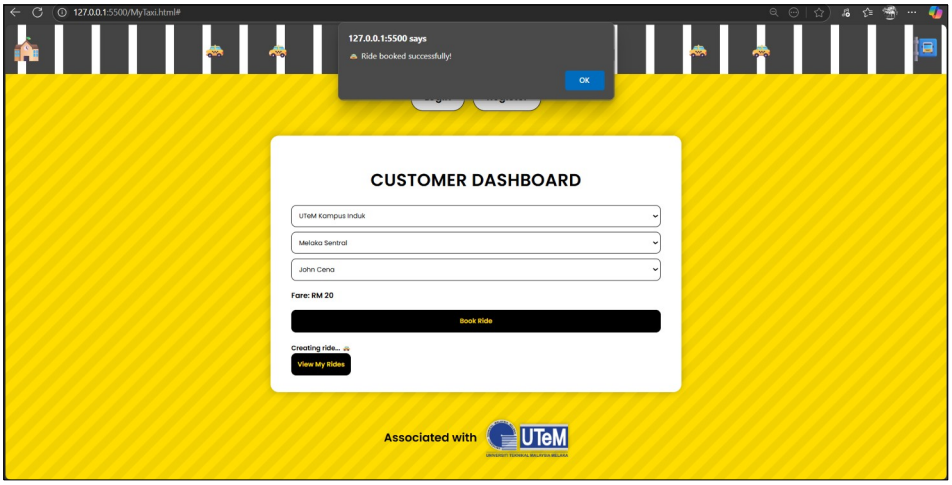

Figure 15 : MyTaxi System : POST rides (customer authorization token)

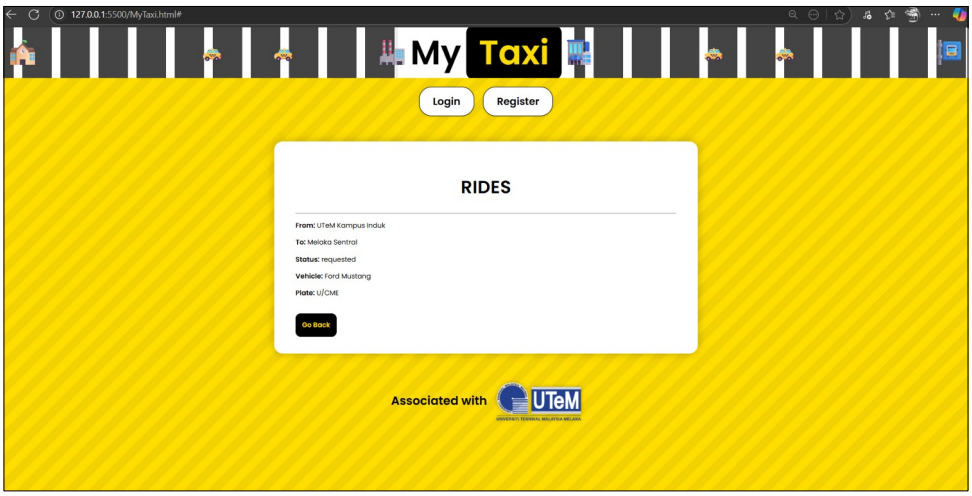

Figure 16 : MyTaxi System : GET rides (customer authorization token)

8.3 DRIVER'S BACKEND

Figure 17 : MyTaxi System : POST drivers/register

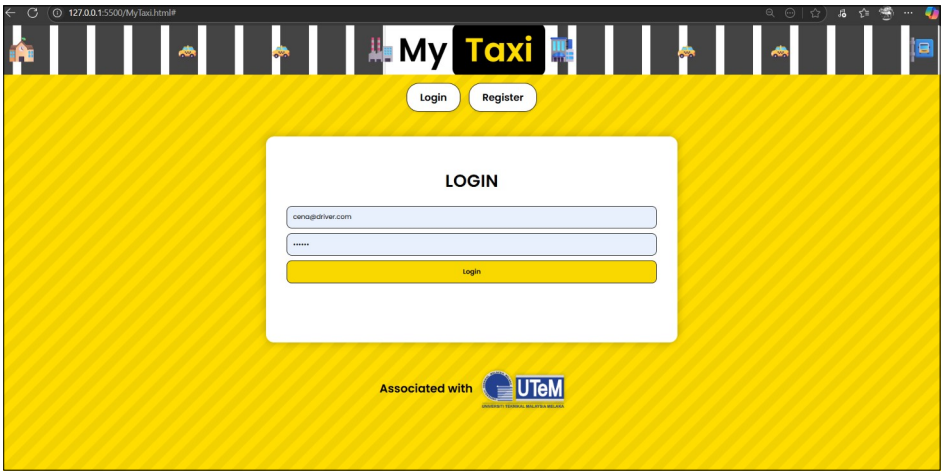

Figure 18 : MyTaxi System : POST auth/login

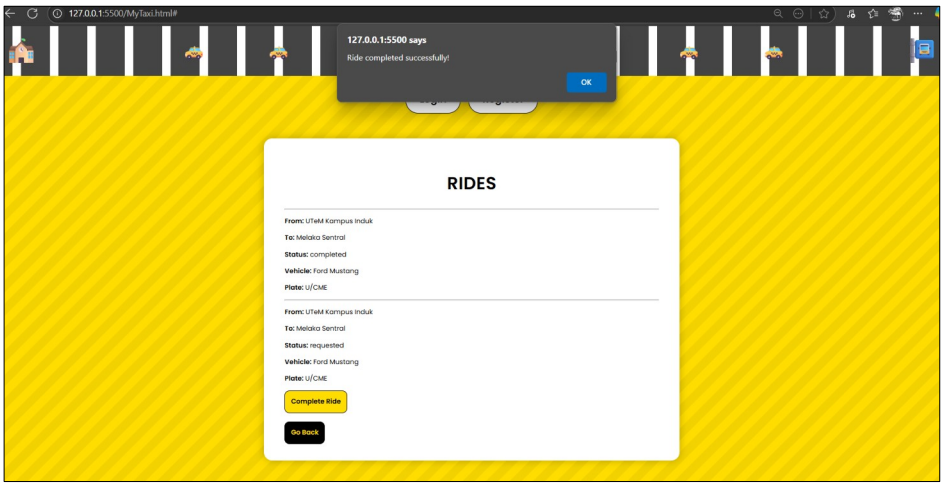

Figure 19 : MyTaxi System : PATCH rides/ride_id & PATCH drivers/driver_id/ status

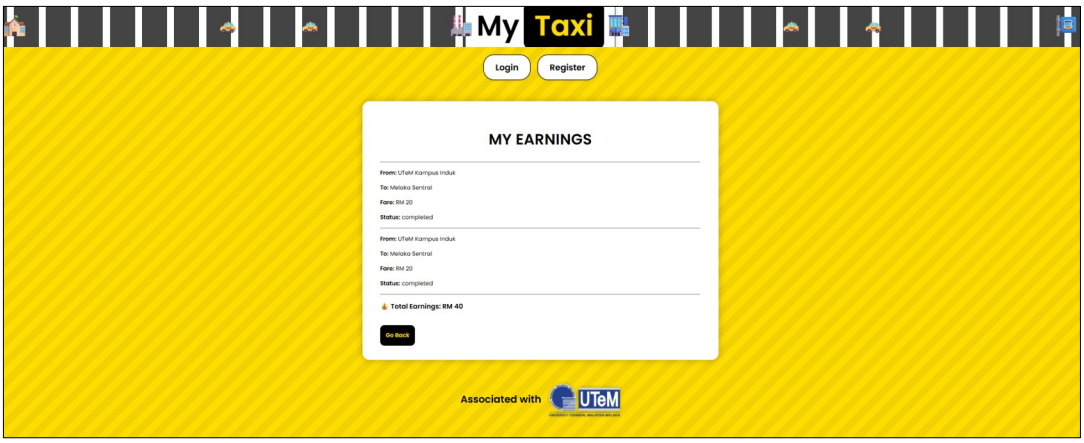

Figure 20 : MyTaxi System : GET/driver_id/earnings

8.4 ADMIN'S BACKEND

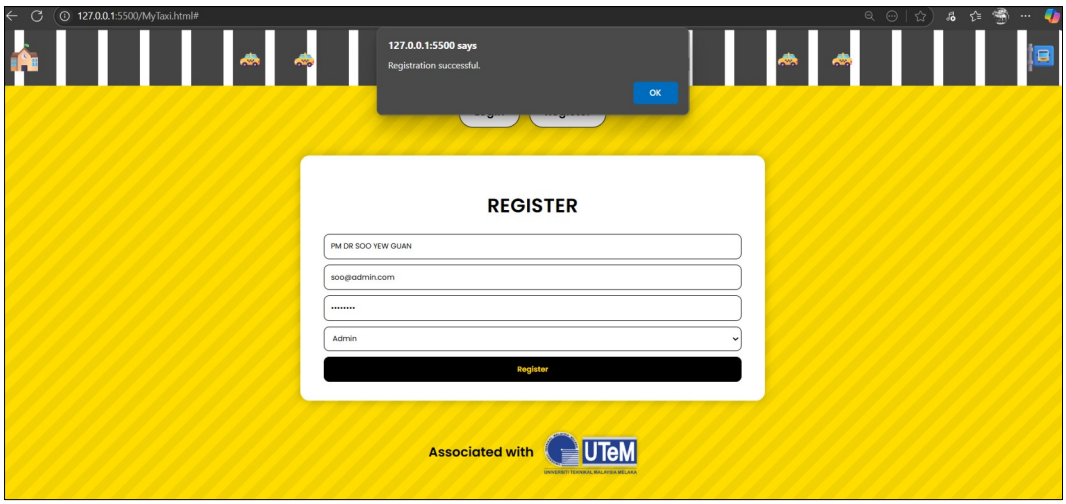

Figure 21 : MyTaxi System : POST admins/register

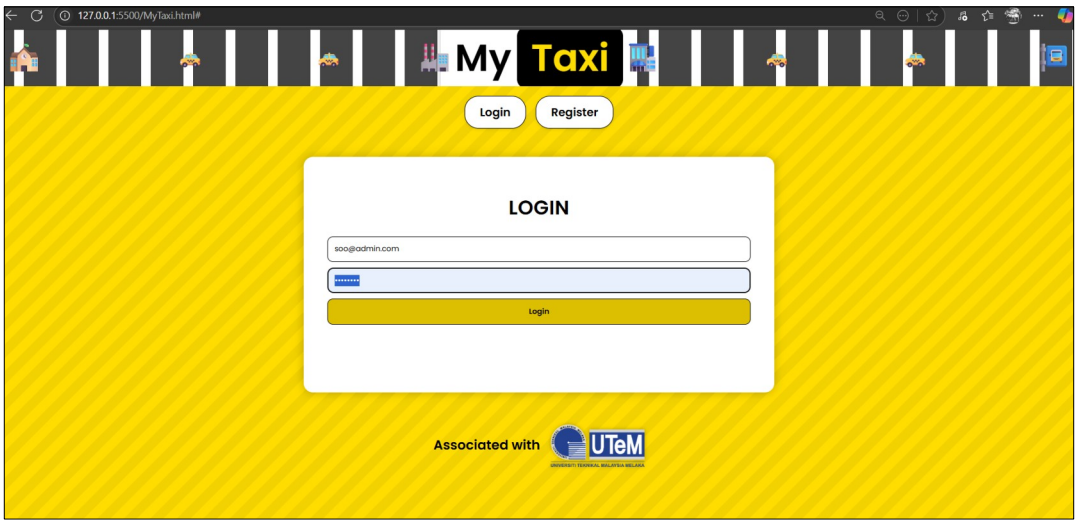

Figure 22 : MyTaxi System : POST auth/login

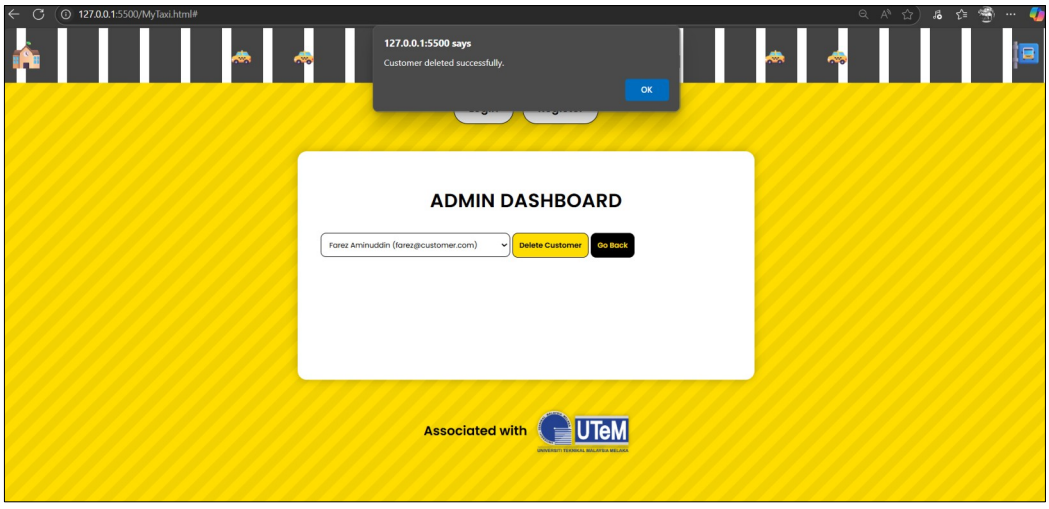

Figure 23 : MyTaxi System : POST admin/customers/customer_id

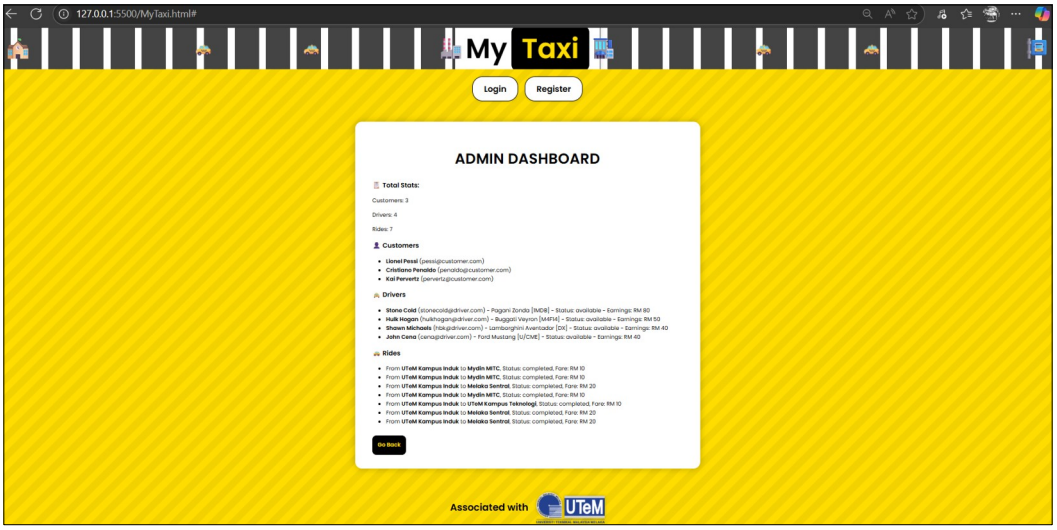

Figure 24 : MyTaxi System : POST admin/analytics

The MyTaxi system provides a complete web-based interface that integrates seamlessly with the backend API to deliver a role-based experience for customers, drivers, and administrators. Upon visiting the main page, users are greeted with a dynamic and visually engaging landing interface, complete with animated cabs and landmark emojis representing key pickup and destination points. The homepage presents two main options: Login and Register. During registration, users can choose their role customer, driver, or admin. If "driver" is selected, additional fields for vehicle type and license plate appear, ensuring proper driver profile creation. Once logged in, users are redirected to their respective dashboards based on the decoded JWT token, which identifies their role and user ID.

For customers, the dashboard provides a form to book rides by selecting predefined pickup and destination points from dropdown menus. The fare is automatically calculated and displayed using a built-in fare chart. Customers can also choose from available drivers dynamically fetched from the backend and view their ride history via the "View My Rides" button. Drivers, on the other hand, can view all rides (especially pending ones) and mark rides as "completed," which triggers fare accumulation to their earnings. They also have a dedicated section to review total earnings, including fare breakdowns.

The admin dashboard offers control features such as deleting customer accounts and viewing detailed analytics. Admins can retrieve real-time data on customers, drivers, and rides, which are displayed in an organized list format with total counts and user details. This is achieved through secure API calls authenticated via JWT tokens stored in the browser’s localStorage. Each section of the UI is hidden or shown dynamically through JavaScript functions using simple show/hide logic and role-based rendering. This client-side logic ensures a smooth and interactive user experience, tightly integrated with the Node.js backend and MongoDB Atlas database for complete system functionality.

9.0 CLOUD DEPLOYMENT

9.1 MONGODB COMPASS

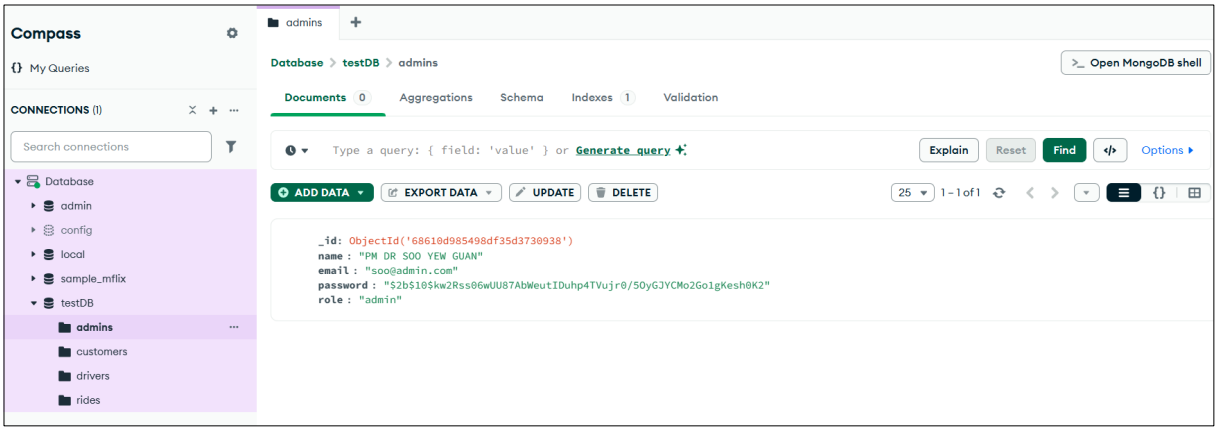

Figure 25 : MyTaxi System : Admin Database

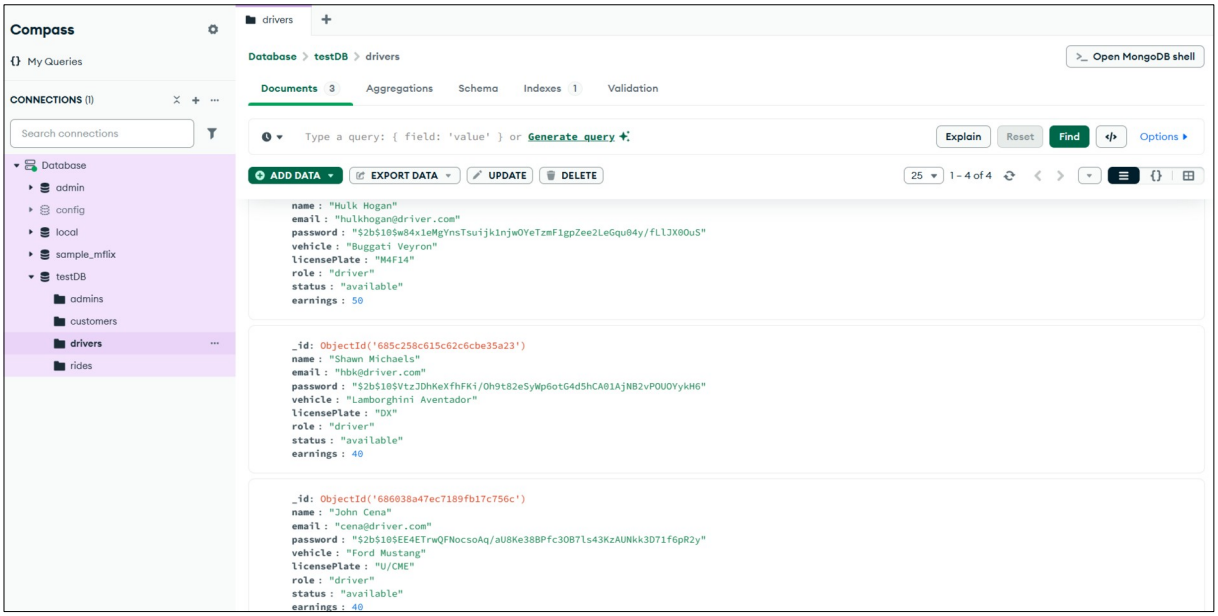

Figure 26 : MyTaxi System : Driver Database

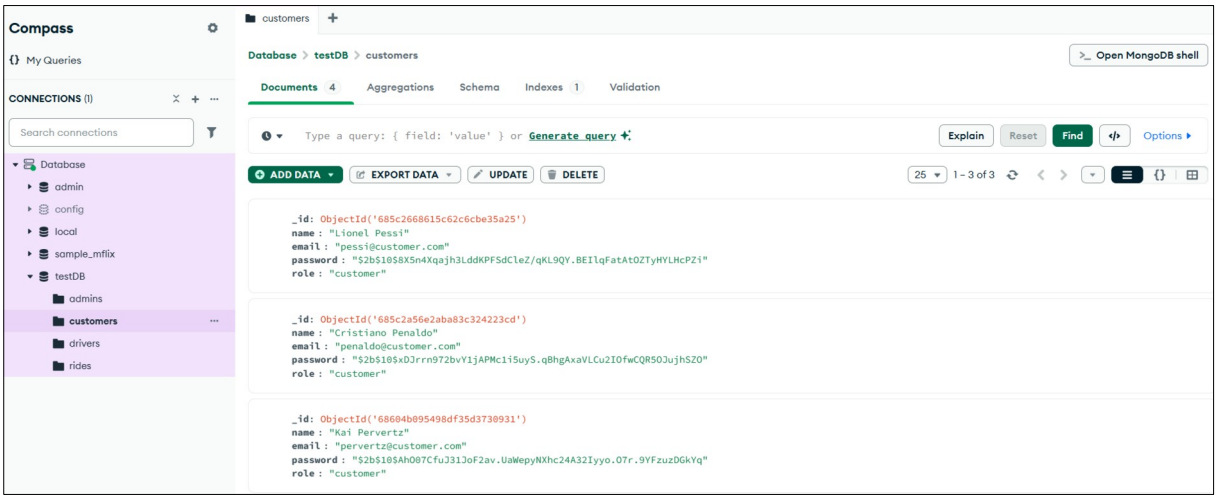

Figure 27 : MyTaxi System : Customer Database

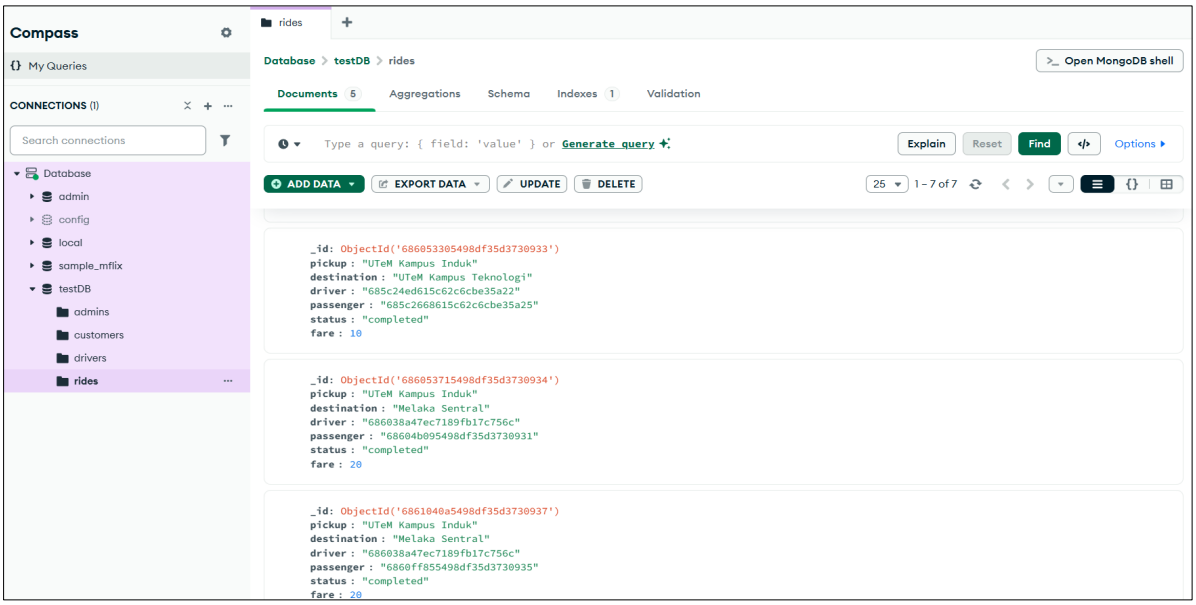

Figure 28: MyTaxi System : Ride Database

The MyTaxi system utilizes MongoDB Atlas, a fully managed cloud database platform, to host and manage all its backend data. By leveraging MongoDB Atlas, the application ensures high availability, scalability, and security without the need to maintain on-premises infrastructure. The deployment begins with the creation of a cloud cluster on the MongoDB Atlas platform. A new database (e.g., testDB) is provisioned within this cluster, and collections such as customers, drivers, admins, and rides are structured to reflect the system's core entities. The backend Node.js application connects to this database securely using a connection URI, which includes the cluster address, username, and password all managed through environment variables to prevent sensitive data exposure.

Access controls are enforced by defining network IP whitelists and user-based authentication in the Atlas dashboard. Role-based access is maintained at the application level, while MongoDB handles data persistence and operational uptime. With features such as automatic backups, monitoring tools, and performance metrics, MongoDB Atlas provides a reliable foundation for handling real-time operations, such as ride bookings, user logins, and data analytics. This cloud deployment ensures that the database is accessible to the backend server regardless of deployment location, enabling seamless development, testing, and scaling for future growth.

9.2 GITHUB (https://github.com/FarezAmin/BERR2243_FAREZAMIN.git)

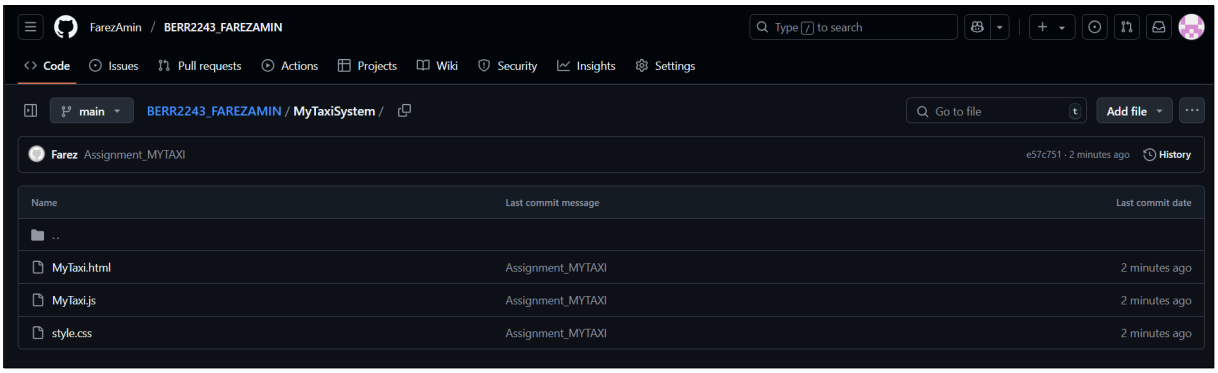

Figure 29: MyTaxi System : Push .js, html, css to GitHub Repository

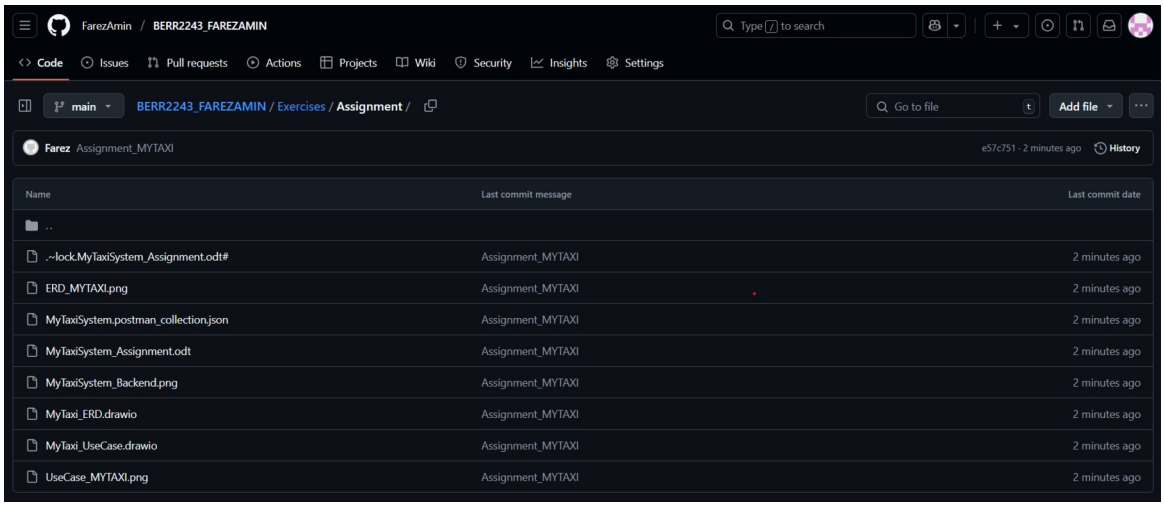

Figure 30: MyTaxi System : Push MyTaxi System files from device to repository

Version control and project collaboration for the MyTaxi system are managed using GitHub, a web-based Git repository platform. All source code including backend APIs, route handlers, database integration logic, and frontend HTML/CSS/JS files are organized and tracked in a central GitHub repository. Developers push regular commits to the repository, enabling transparent tracking of project progress, code changes, and team contributions over time. GitHub branches can be used to isolate feature development or bug fixes, which are then merged into the main branch after review.

The repository's README file provides essential documentation, including project setup instructions, API endpoint references, environment configuration requirements, and system usage guidelines. This makes it easy for collaborators and external users to understand and deploy the application. Additionally, GitHub Issues and Pull Requests are useful tools for managing bugs, code reviews, and updates during development. By hosting the project on GitHub, the team benefits from continuous backup, easy access from any location. GitHub thus plays a vital role in ensuring code quality, collaboration, and deployment readiness throughout the software development lifecycle.

10.0 CONCLUSION

The development of the MyTaxi system represents a significant step toward equipping students with hands-on experience in modern software engineering practices, particularly in the areas of cloud computing, backend development, and database architecture. By focusing on the integration of MongoDB Atlas for cloud-hosted data management and Node.js for server-side API development, this project demonstrates a full-cycle implementation of a scalable, secure, and role-based platform that aligns with real-world industry standards.

For students, the importance of implementing a database and cloud system extends far beyond academic fulfillment. It cultivates a deep understanding of data modeling, asynchronous communication, RESTful API architecture, and user authentication mechanisms using secure technologies like JWT and bcrypt. Through the MyTaxi system, students also gain insight into multi-role application logic, supporting differentiated access for customers, drivers, and administrators a feature common in many enterprise and SaaS solutions.

Leveraging cloud technologies such as MongoDB Atlas exposes students to critical concepts in cloud-native application deployment, including database scaling, access control, uptime reliability, and disaster recovery all of which are essential for building robust, production-grade systems. The integration with version control platforms like GitHub further reinforces modern development workflows such as collaboration, issue tracking, version management, and remote deployment readiness.

In a broader sense, this project contributes to future innovation by laying a technical foundation that students can extend or adapt for other domains such as logistics, delivery services, or institutional transport solutions. It also encourages problem-solving in a digital economy where on-demand services are in high demand. As digital transformation becomes a priority across all sectors, students with practical experience in designing, deploying, and managing cloud-based backend systems will be well-prepared to contribute meaningfully to industry, research, and entrepreneurial ventures.

Ultimately, the MyTaxi project not only delivers a working application but also acts as a learning platform that strengthens the students' competencies in modern backend development, cloud database management, and full-stack system integration all of which are vital for thriving in the fast-evolving field of information technology.

11.0 REFERRENCES

1. MongoDB Inc. (n.d.). *MongoDB Atlas: Cloud database as a service*. https://www.mongodb.com/cloud/atlas

2. Node.js Foundation. (n.d.). *About Node.js*. https://nodejs.org/en/about/

3. Auth0. (n.d.). *Introduction to JSON Web Tokens*. https://auth0.com/intro-to-jwt/

4. bcrypt.js Contributors. (n.d.). *bcrypt.js - A bcrypt library for Node.js*. https://github.com/dcodeIO/bcrypt.js

5. Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Doctoral dissertation, University of California, Irvine]. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

6. GitHub Docs. (n.d.). *About repositories*. https://docs.github.com/en/repositories

7. Mozilla Developer Network. (n.d.). *Cross-Origin Resource Sharing (CORS)*. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS