INFS 740

# Project Report Vehicle Registration System

Stephen Dias sdias3@gmu.edu (G01387625)
Aditya Milind Limbekar alimbeka@gmu.edu (G01384408)

**Link to source code:** https://github.com/STEPHENDIAS10/INFS740_Project_vehicle-registration-system

## • Introduction

The **Vehicle Registration System** is a comprehensive solution designed to manage information related to vehicles, user details, and reviews. The system provides users with the ability to perform Create, Read, Update, and Delete (CRUD) operations on vehicles, reviews, and user details. This report outlines the key features, components, and functionalities of the system.

## • Technologies used

→ **Frontend**: ReactJS, CSS
→ **Backend**: NodeJS
→ **Database**: MongoDB

## • System Overview

The Vehicle Registration System consists of three main modules:

**a.** **Vehicle Management: -** Allows users to register new vehicles (cars/bikes) by providing necessary details such as make, model, year, and registration information. - Provides the ability to view a list of registered vehicles. - Allows users to update the information of existing vehicles or delete them from the system.

**b.** **Review Management:** - Enables users to submit reviews for registered vehicles, including ratings and comments. - Allows users to view and manage their own reviews. - Provides administrators with the ability to moderate and manage all reviews in the system.

**c.** **User Management:** - Allows users to create accounts by providing personal information. - Provides the ability to view and update user profiles. - Administrators can manage user accounts, including creating, updating, and deleting user profiles.

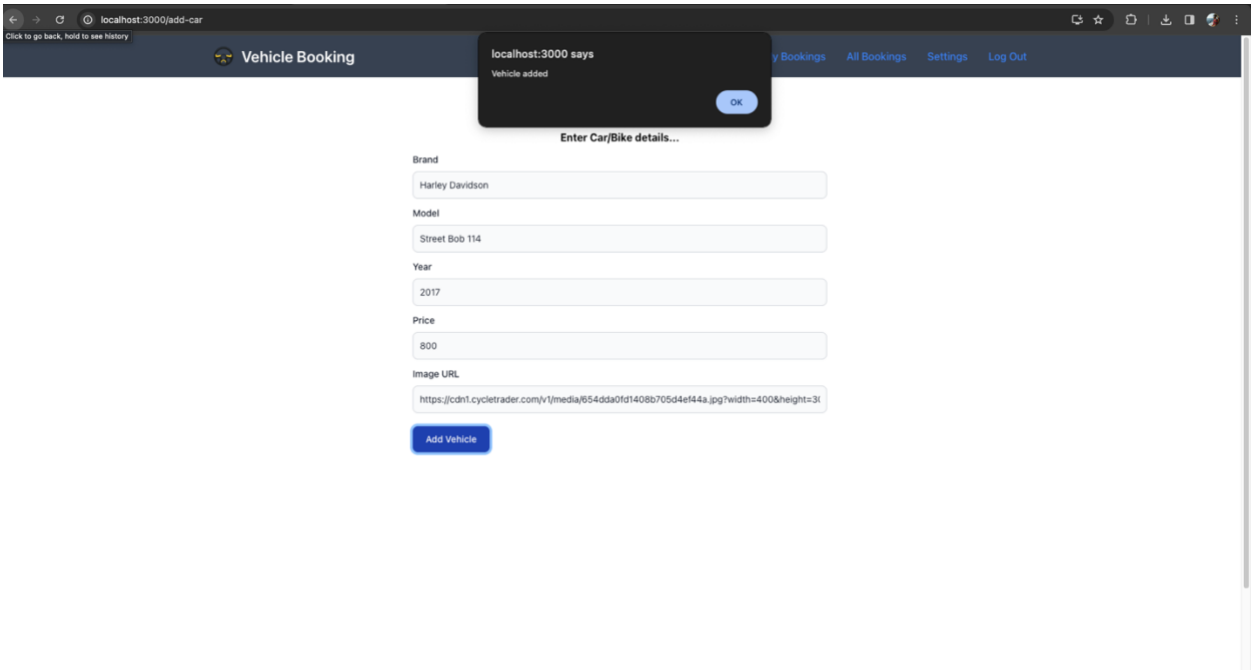
## • CRUD Operations

### a. Vehicle CRUD Operations:

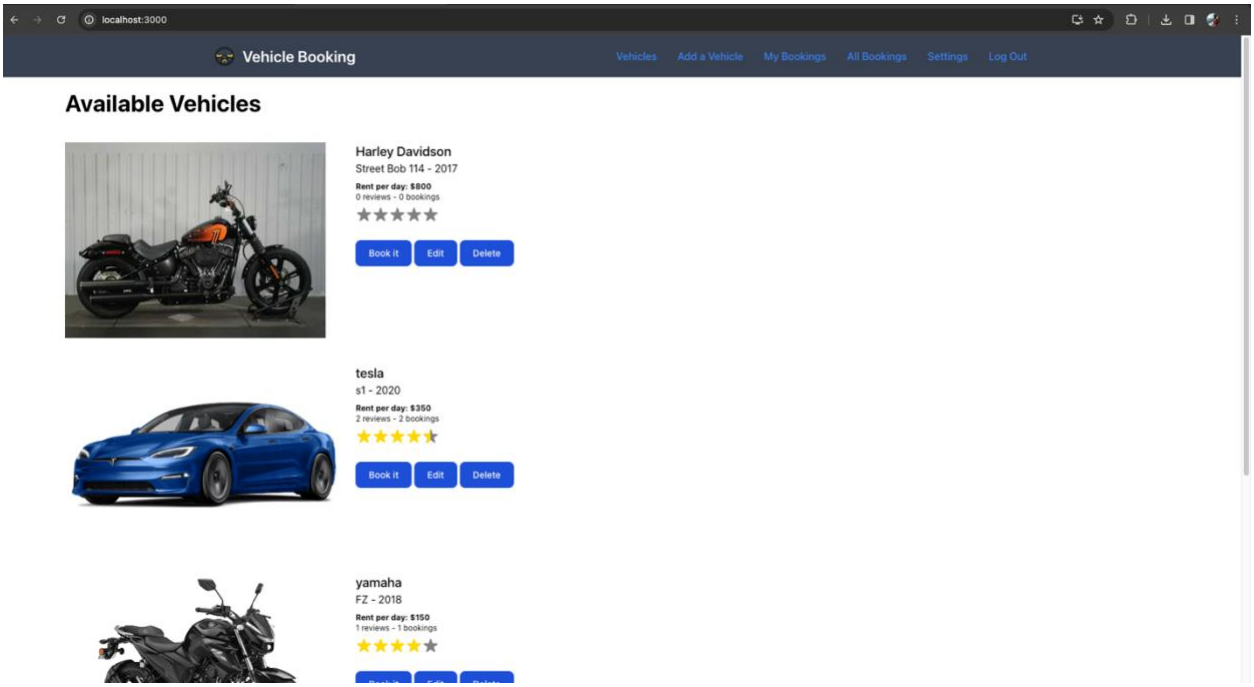
**Create**: Users can add new vehicles to the system by providing relevant information.
**Read**: Users can view a list of all registered vehicles along with their details.

**Update**: Users can modify the information of existing vehicles.
**Delete**: Users can remove vehicles from the system.

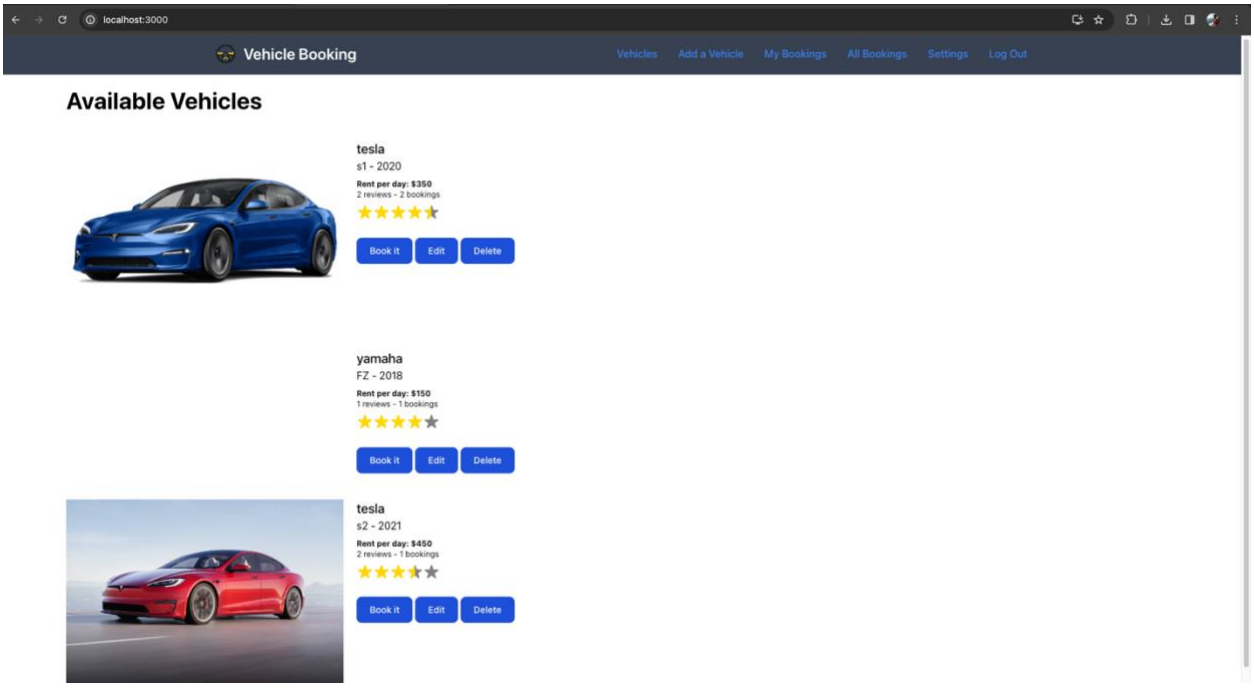
→ **Add a vehicle**

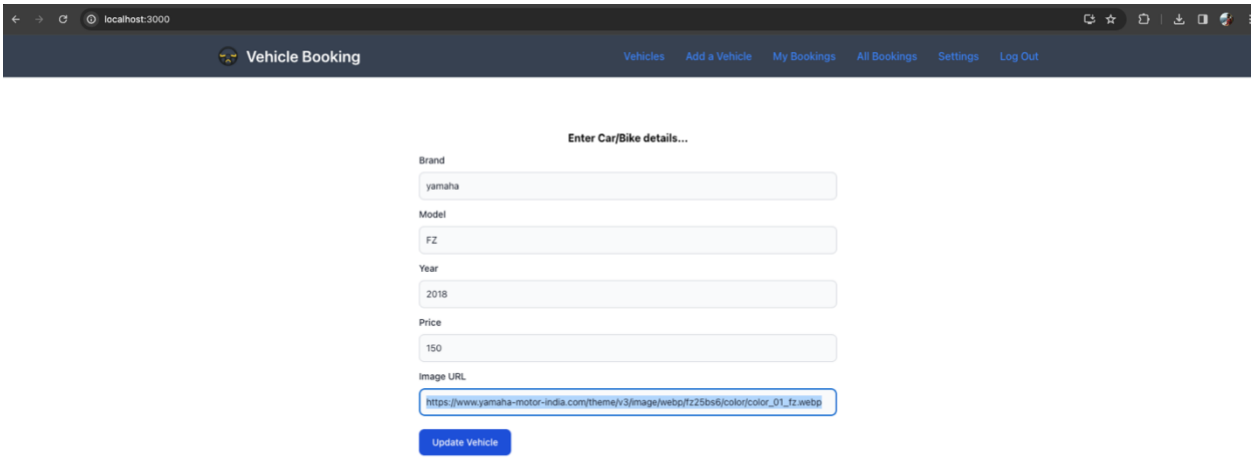


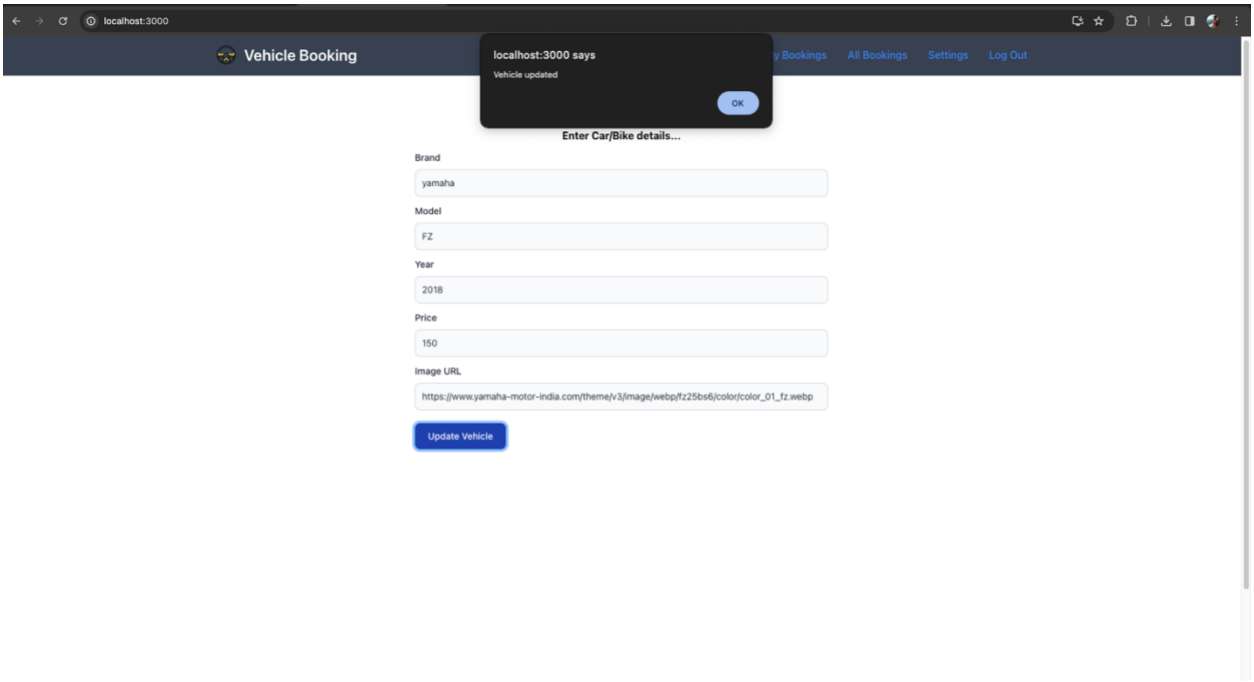

→ **View/Read a vehicle**

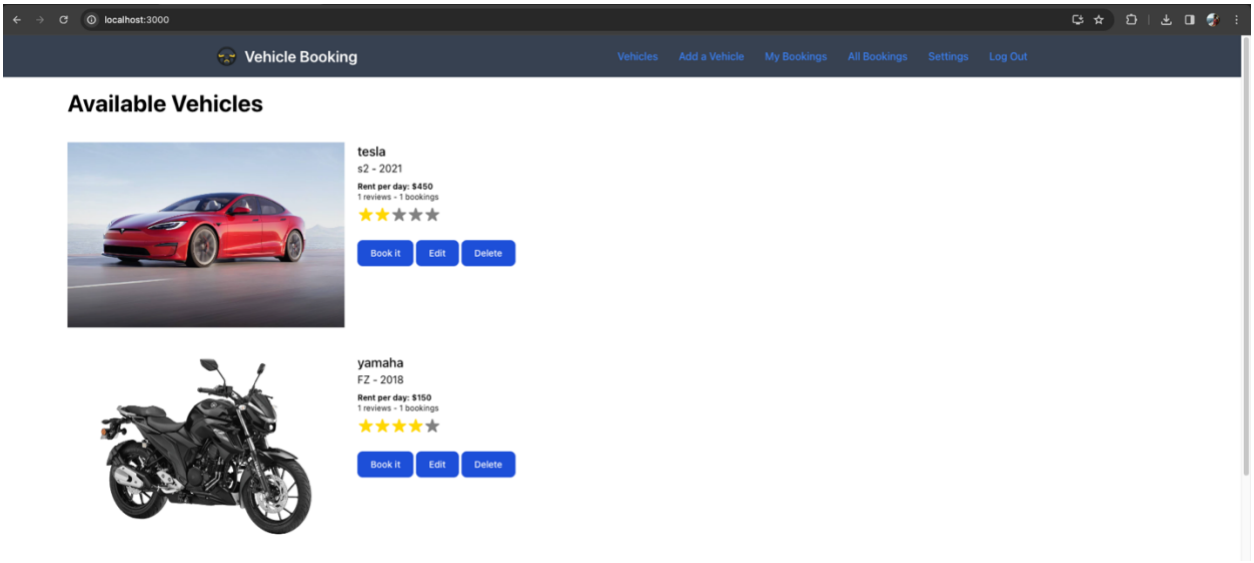



→ **Update a vehicle**

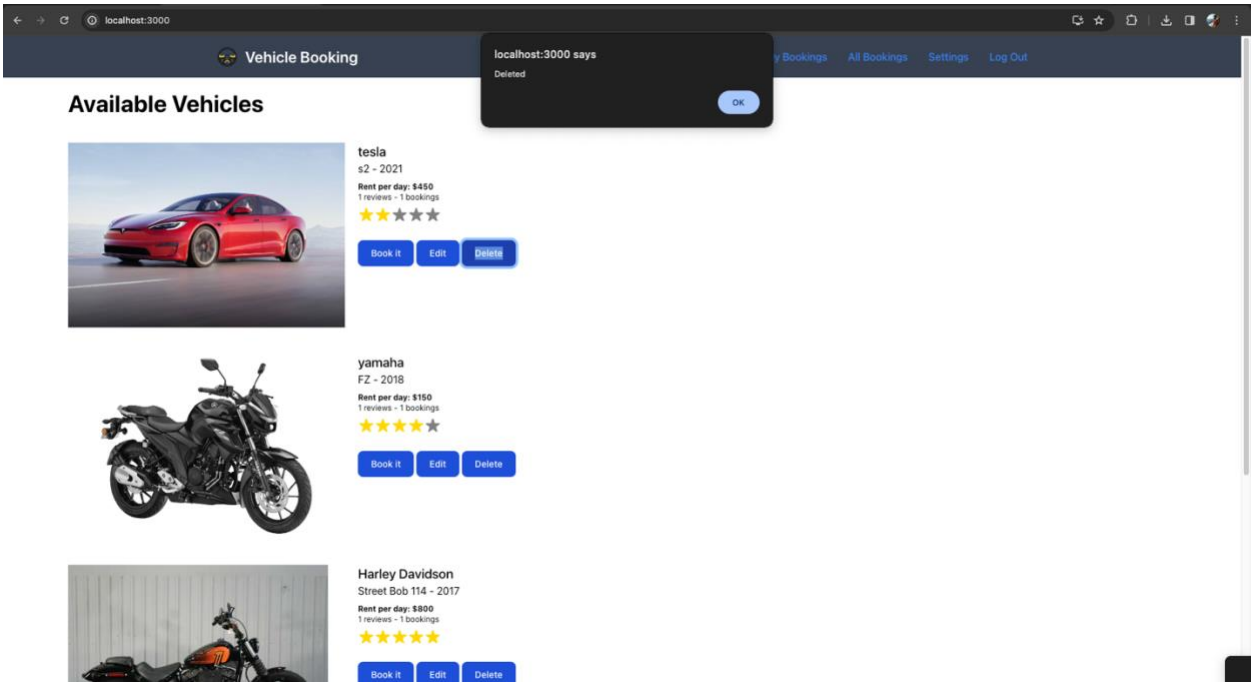


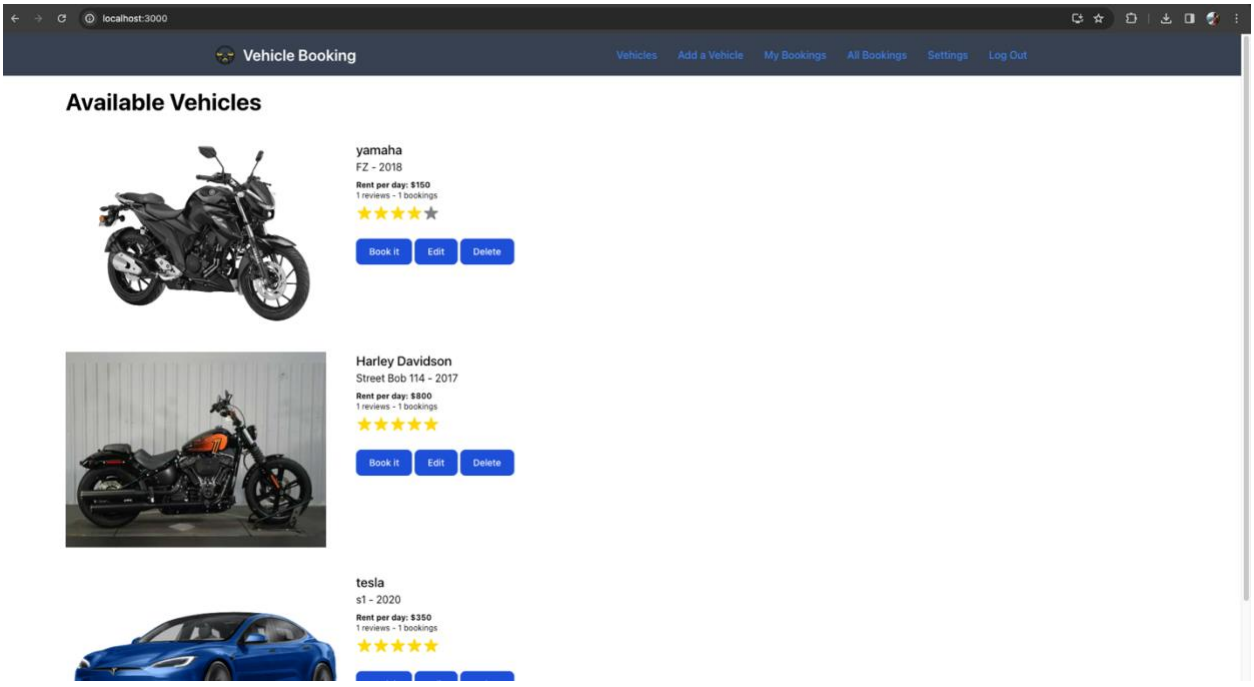

localhost:3000
Vehicle Booking
Vehicles
Add a Vehicle
My Bookings
All Bookings
Settings
Log Out
Enter Car/Bike details...
Brand
yamaha
Model
FZ
Year
2018
Price
150
Image URL
https://www.yamaha-motor-india.com/theme/v3/image/webp/fz25bs6/color/color_01_fz.webp
Update Vehicle


localhost:3000
Vehicle Booking
localhost:3000 says
Vehicle updated
OK
All Bookings
Settings
Log Out
Enter Car/Bike details...
Brand
yamaha
Model
FZ
Year
2018
Price
150
Image URL
https://www.yamaha-motor-india.com/theme/v3/image/webp/fz25bs6/color/color_01_fz.webp
Update Vehicle




→ **Delete a vehicle**



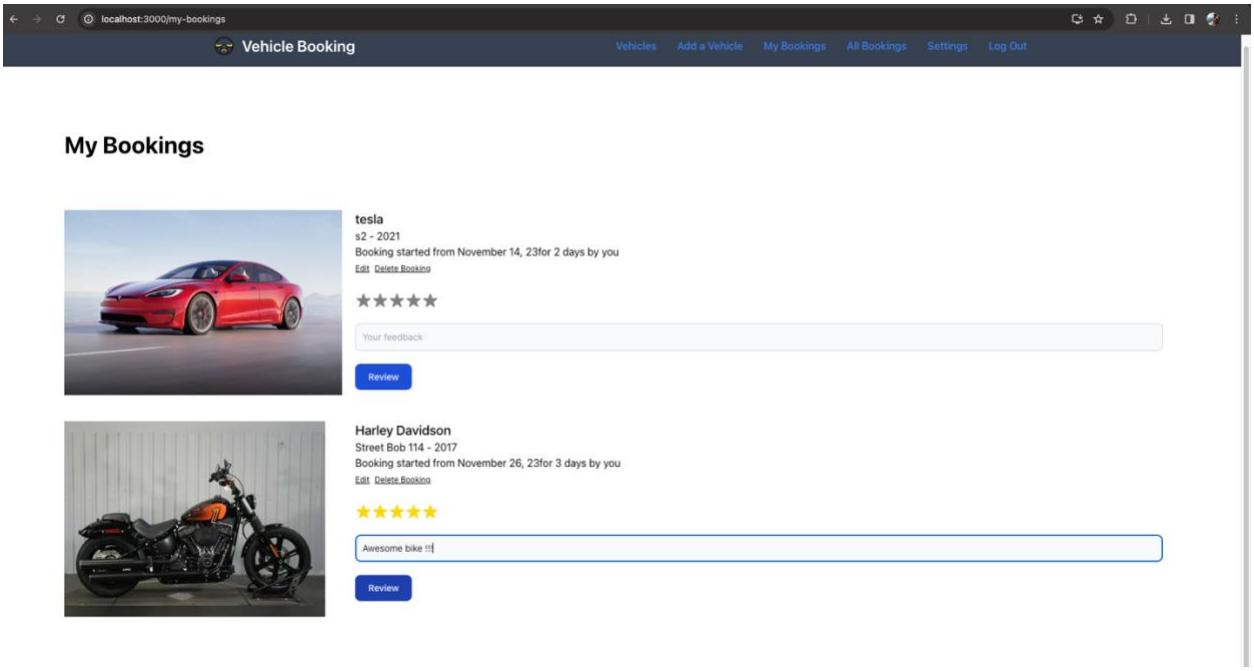



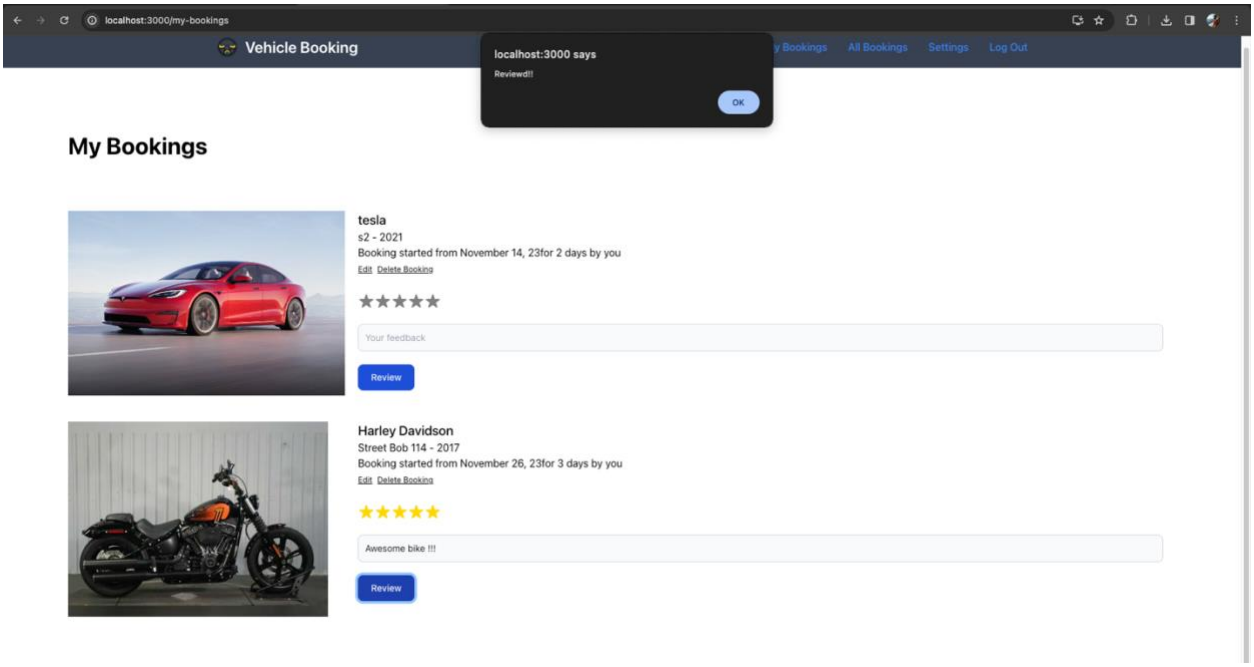

**b. Review CRUD Operations:**

**Create**: Users can submit reviews for specific vehicles.

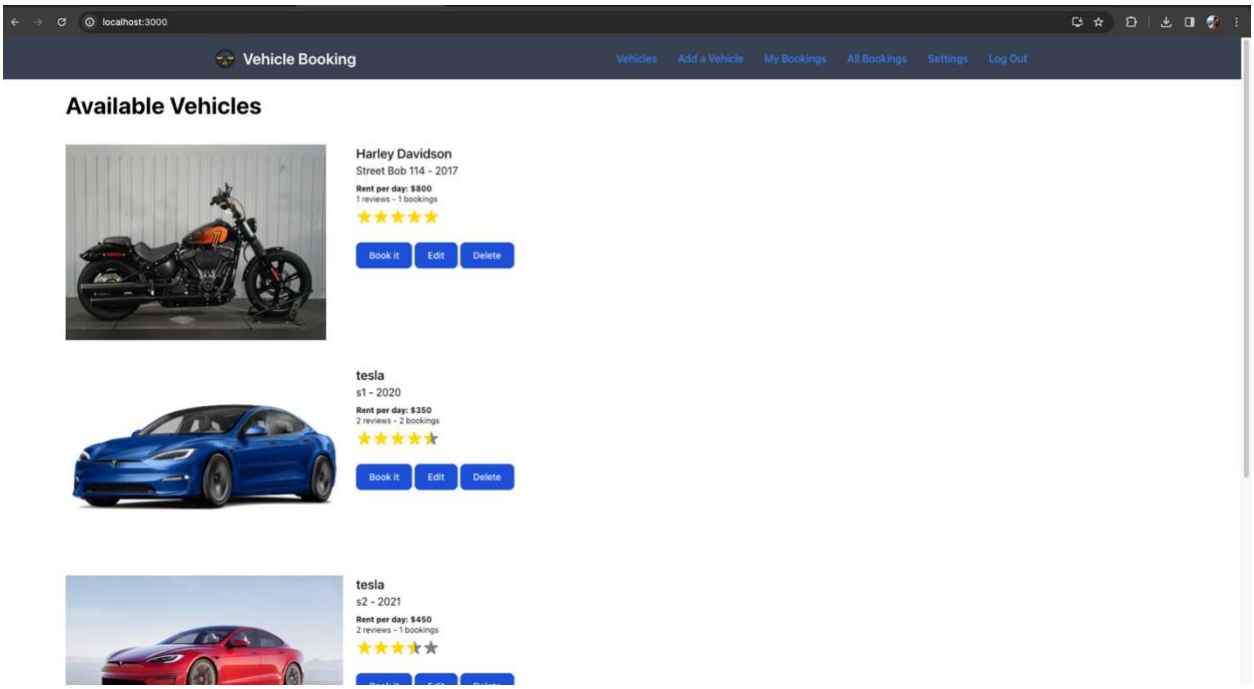
**Read**: Users can view their own reviews and administrators can view all reviews.
**Update**: Users can edit their own reviews.
**Delete**: Users can delete their own reviews, and administrators can delete any review for moderation purposes.

→ **Add a review**

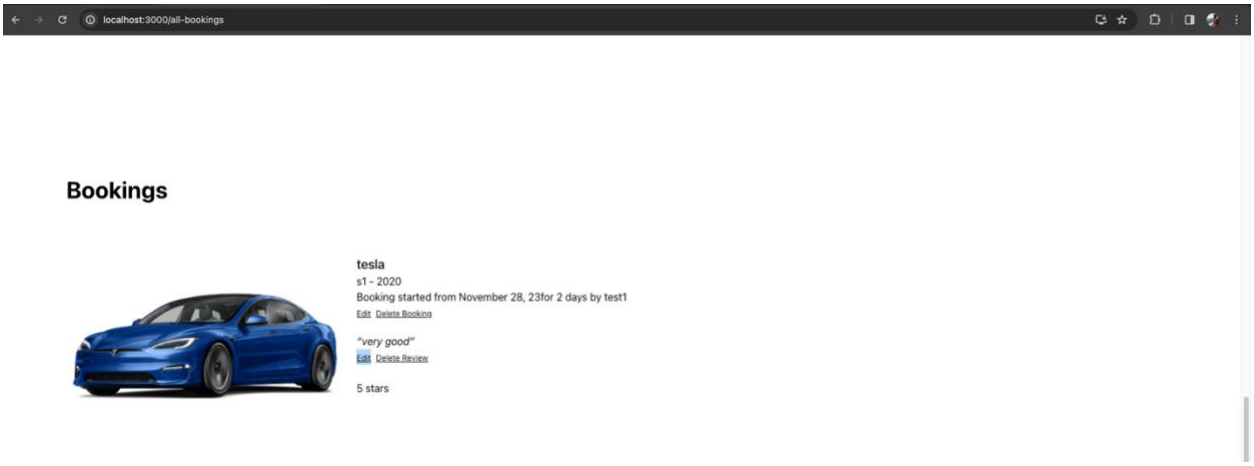


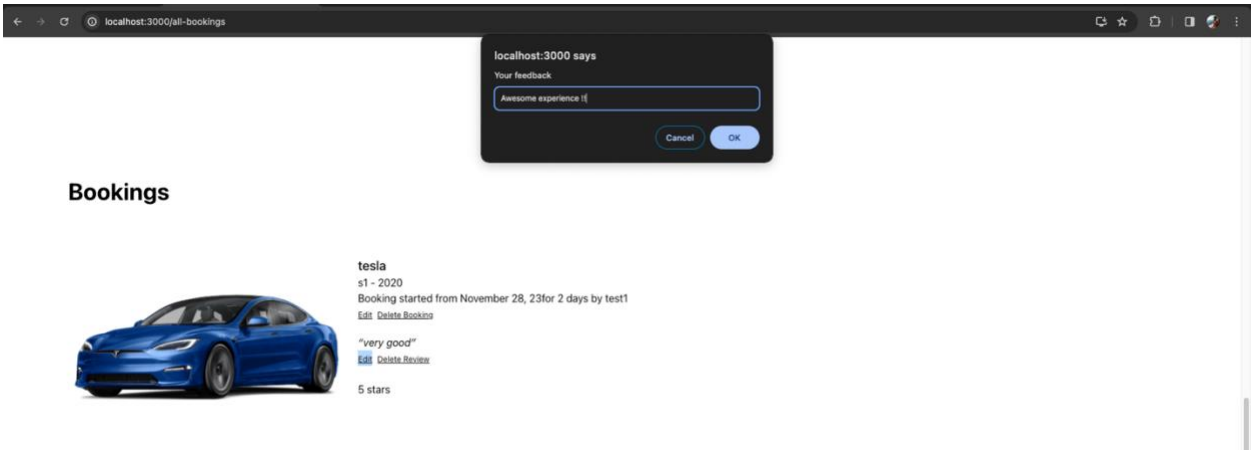



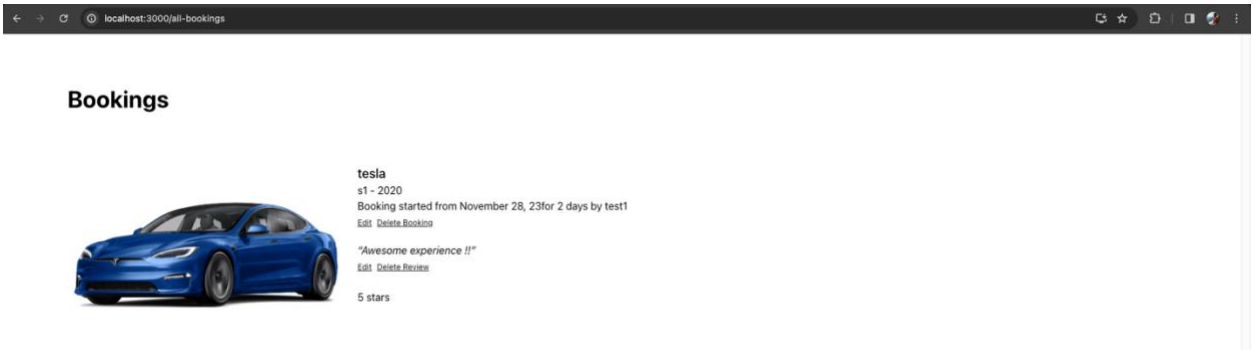
→ **View/Read a review**



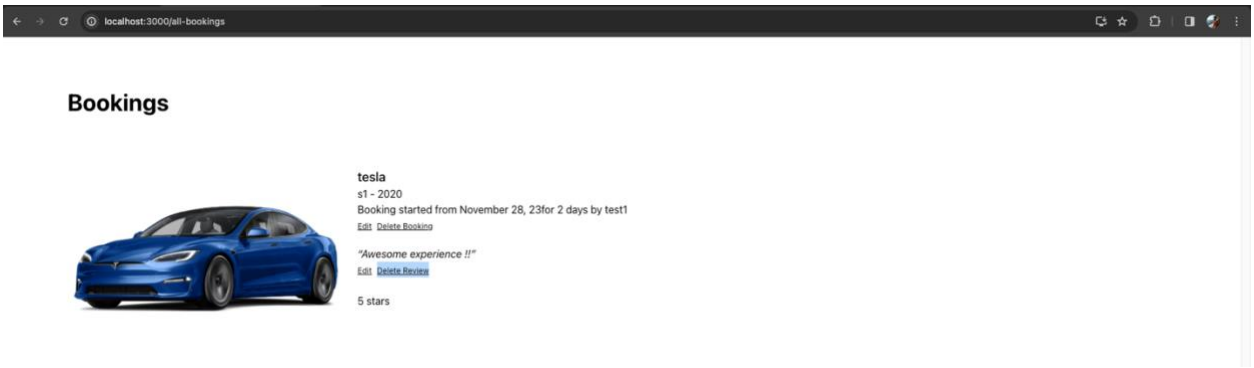

→ **Update a review**

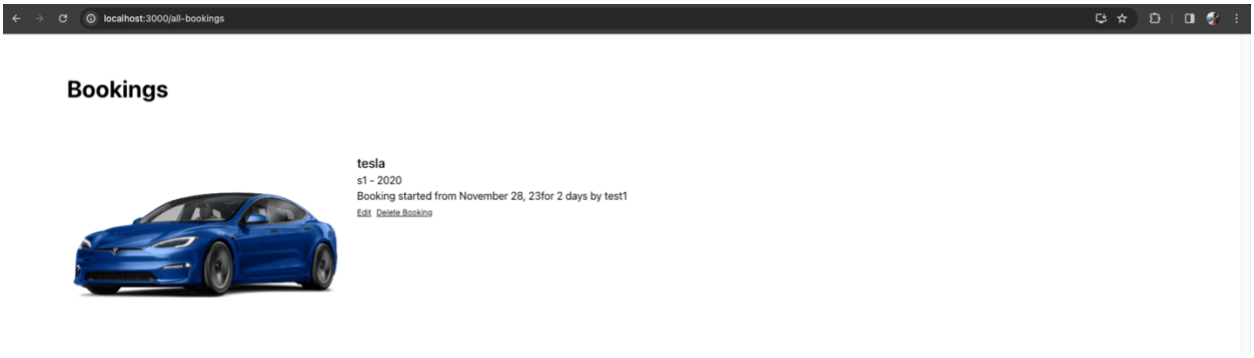





→ **Delete a review**

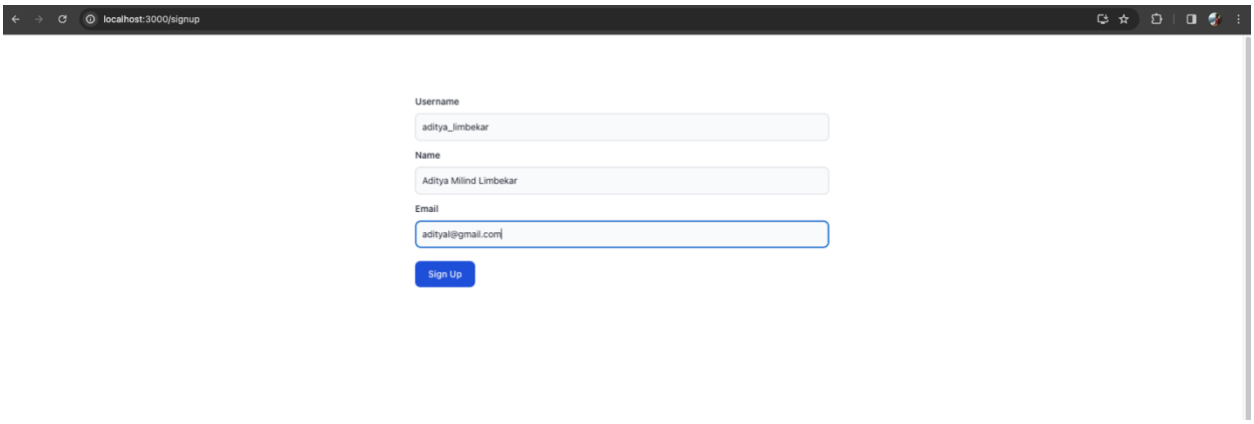


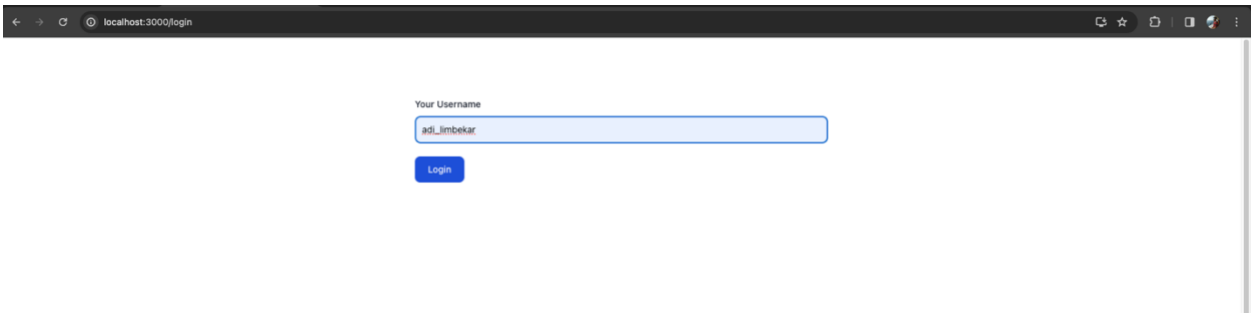



**c. User CRUD Operations:**

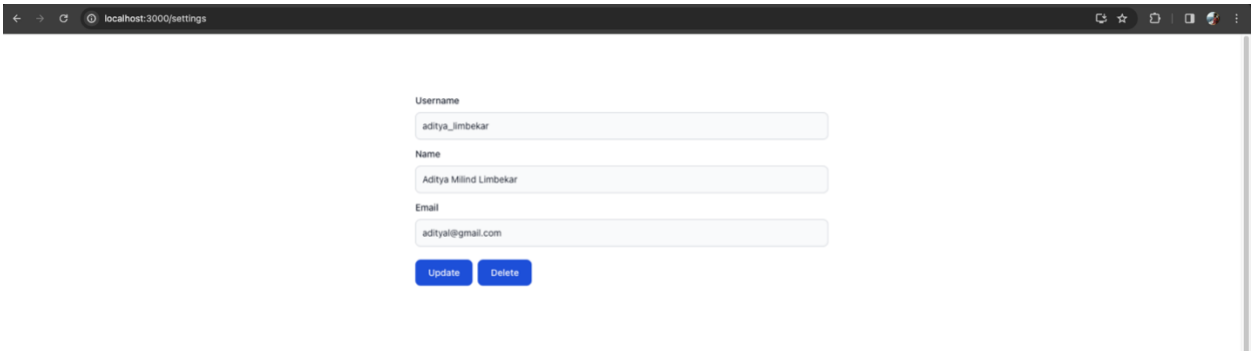
**Create**: Users can create accounts by providing personal information.
**Read**: Users can view their own profiles, and administrators can view all user profiles.
**Update**: Users can update their own profiles, and administrators can update any user profile.
**Delete**: Users can delete their own accounts, and administrators can delete any user account.

→ **Add a user**

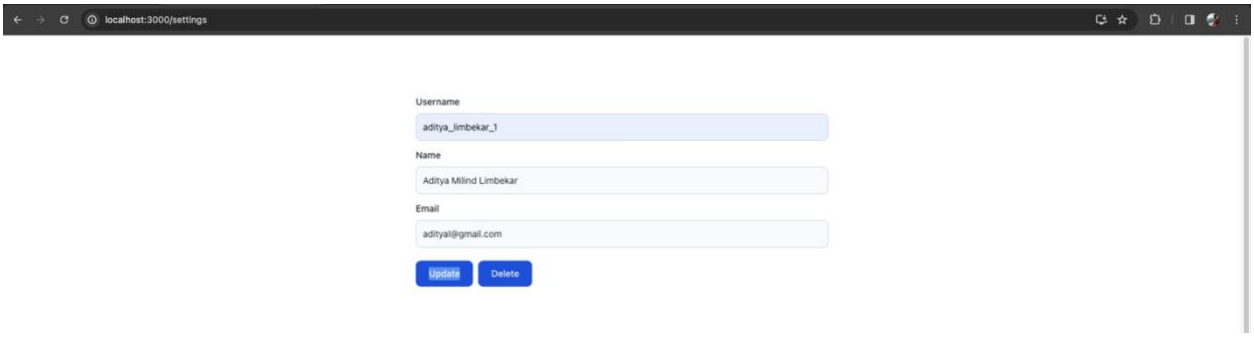


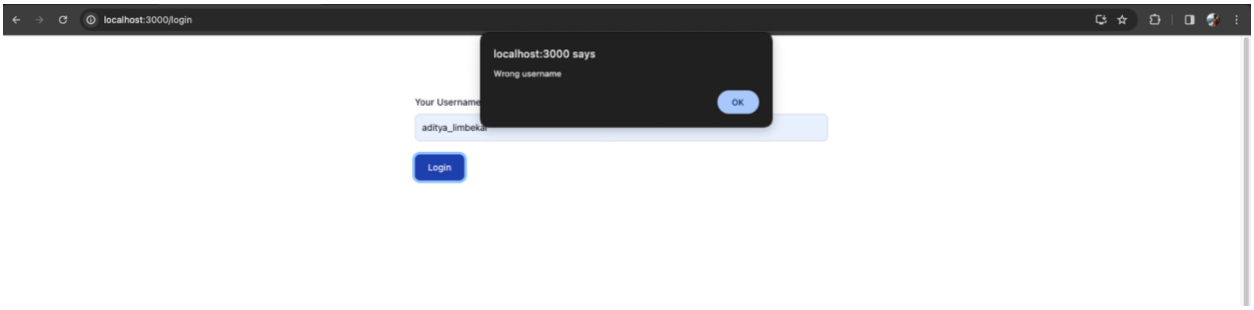



→ **View/Read a user**

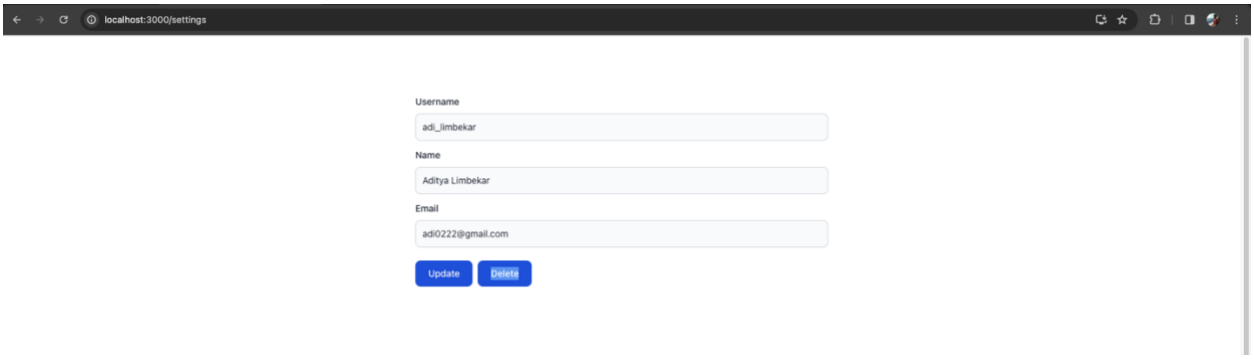


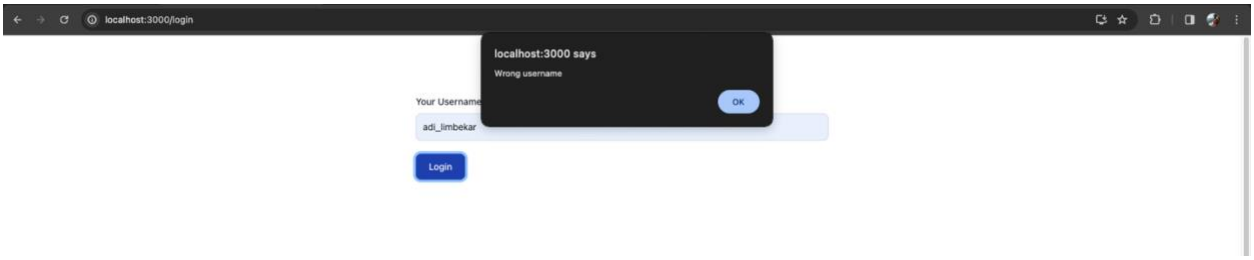

→ **Update a user**



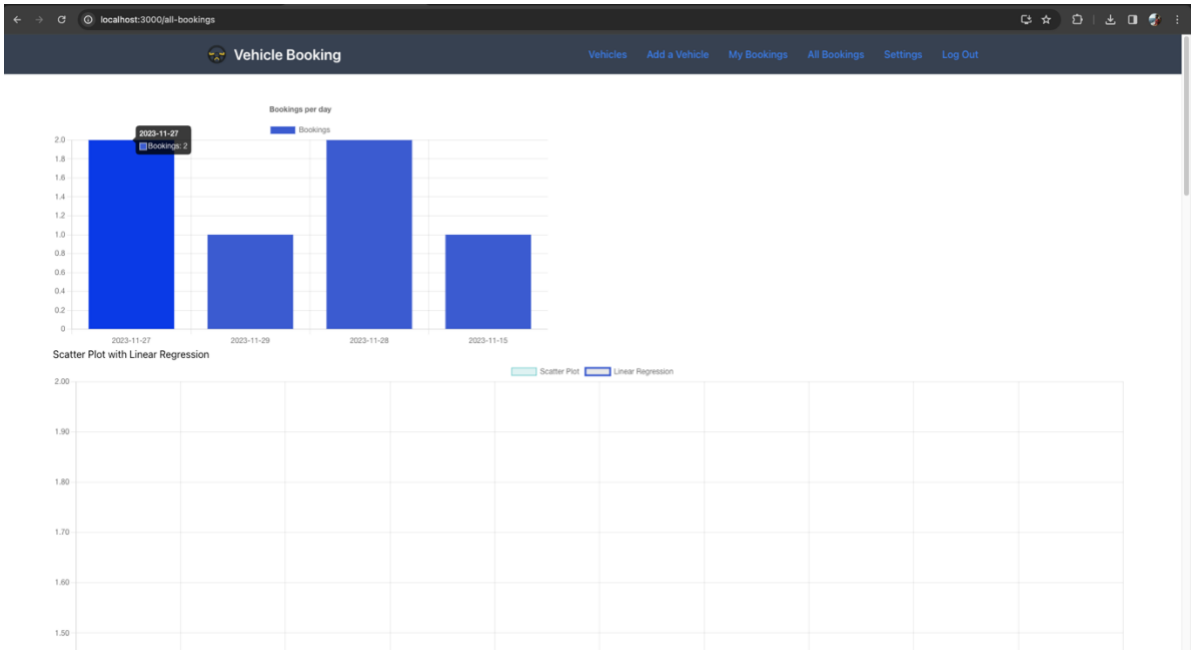



→ **Delete a user**

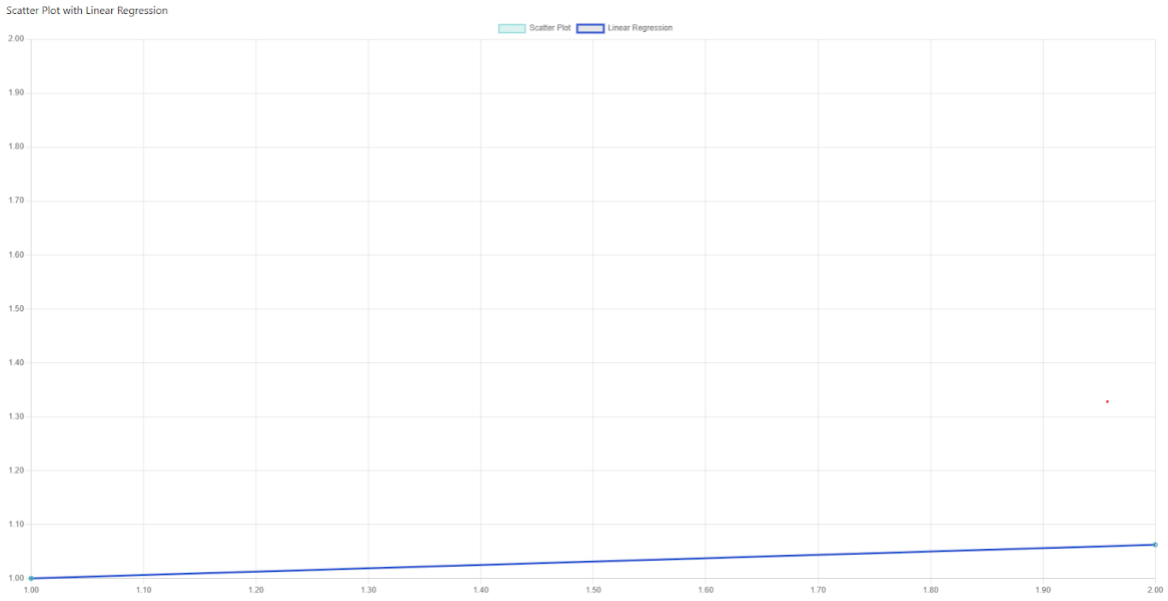


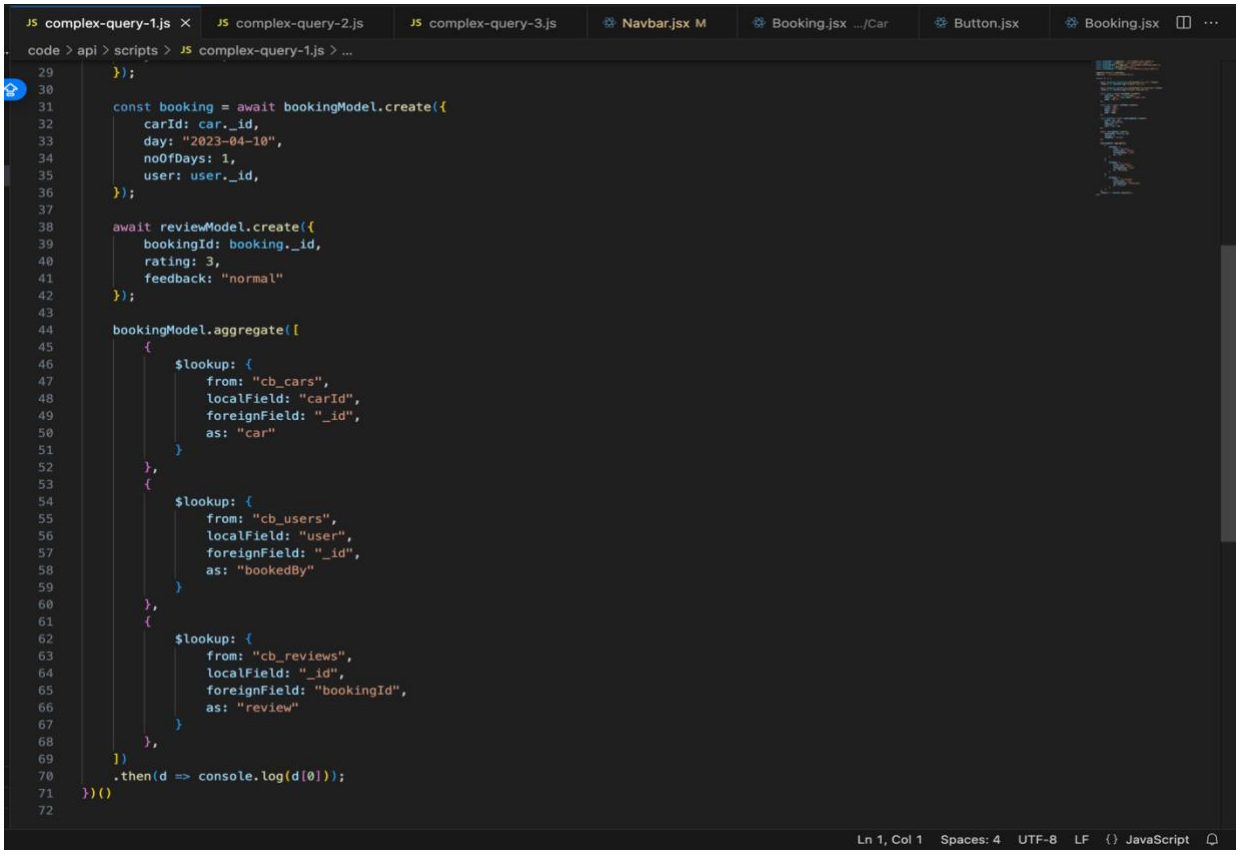

- **Implementation of visualization and linear regression using machine learning**
  - Data visualization has been done for the numbers cars that have been booked for the number of days.

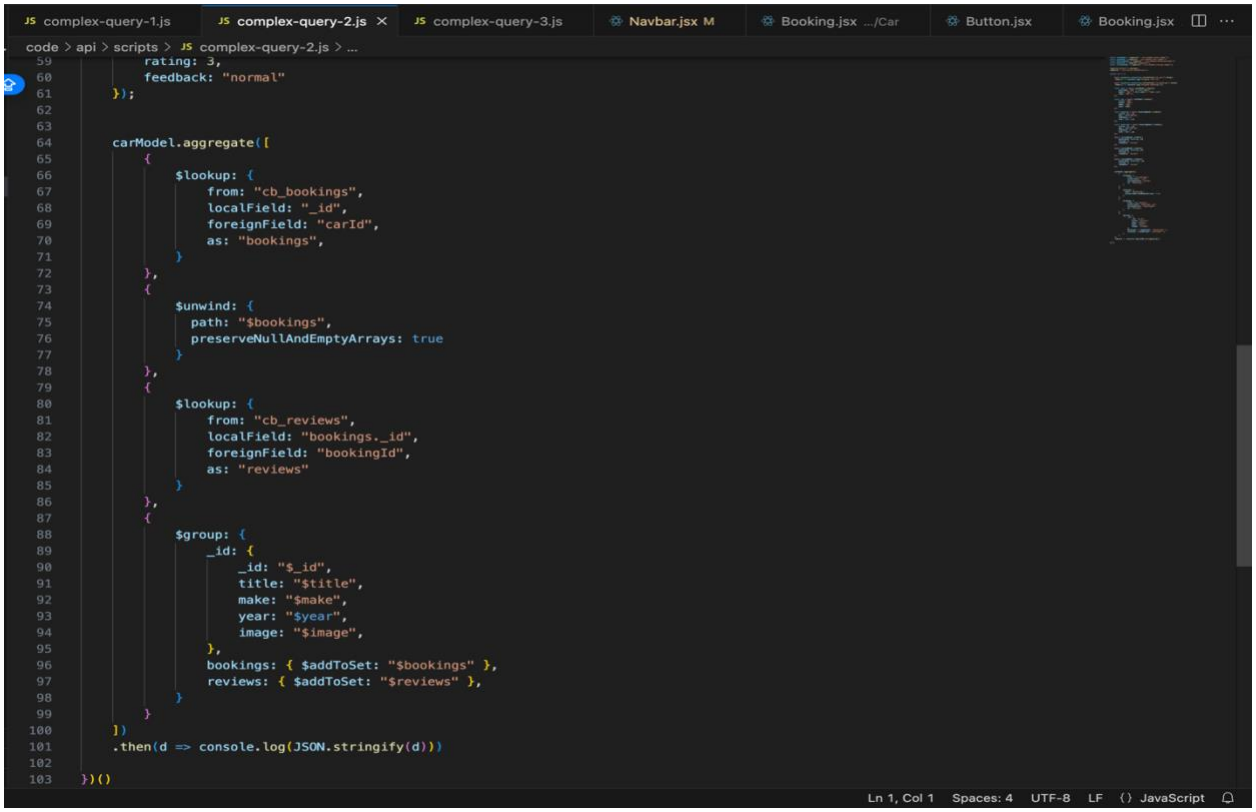



- On the same data the Linear regression has been performed and show how good the system is doing and how they can improve the booking rate.



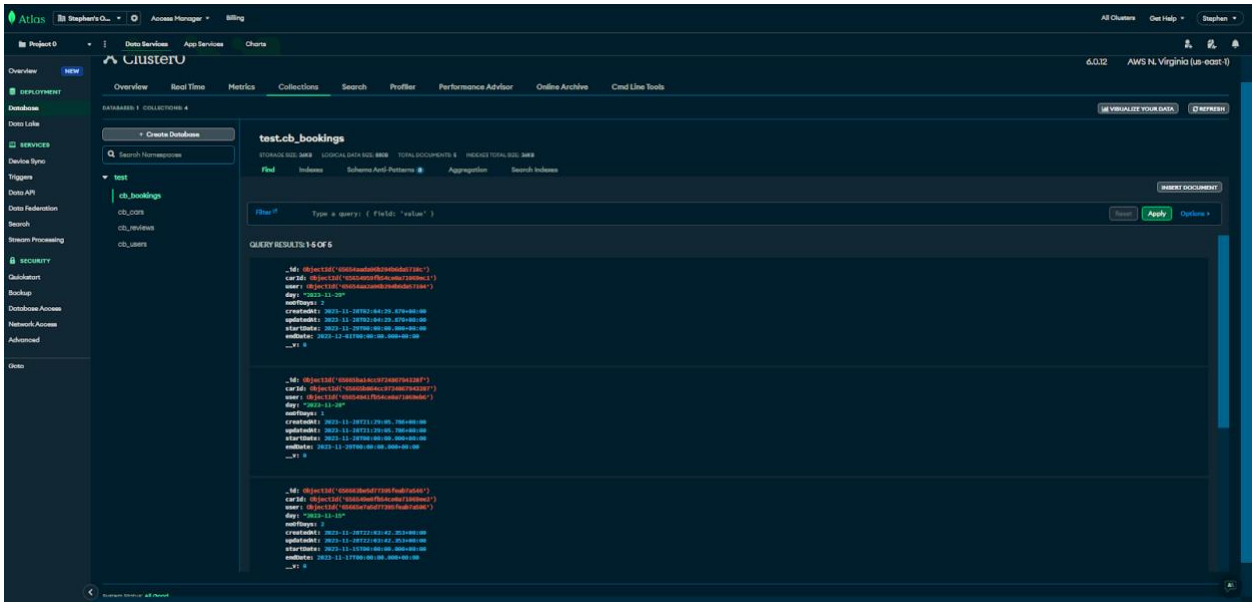

- **Complex queries**
  We have implemented 3 complex queries.

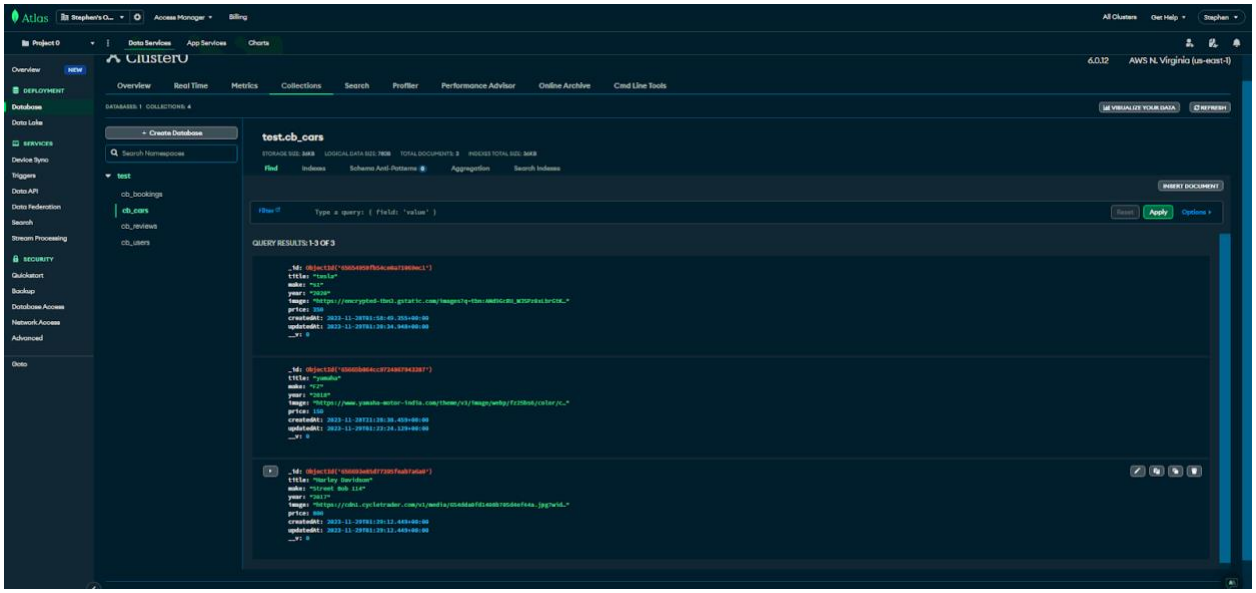

complex-query-1.js
complex-query-2.js
complex-query-3.js
Navbar.jsx M
Booking.jsx .../Car
Button.jsx
Booking.jsx
code > api > scripts > complex-query-1.js > ...
});
const booking = await bookingModel.create({
carId: car._id,
day: "2023-04-10",
noOfDays: 1,
user: user._id,
});
await reviewModel.create({
bookingId: booking._id,
rating: 3,
feedback: "normal"
});
bookingModel.aggregate([
{
$lookup: {
from: "cb_cars",
localField: "carId",
foreignField: "_id",
as: "car"
}
},
{
$lookup: {
from: "cb_users",
localField: "user",
foreignField: "_id",
as: "bookedBy"
}
},
{
$lookup: {
from: "cb_reviews",
localField: "_id",
foreignField: "bookingId",
as: "review"
}
},
])
.then(d => console.log(d[0]));
})()
Ln 1, Col 1
Spaces: 4
UTF-8
LF
{} JavaScript

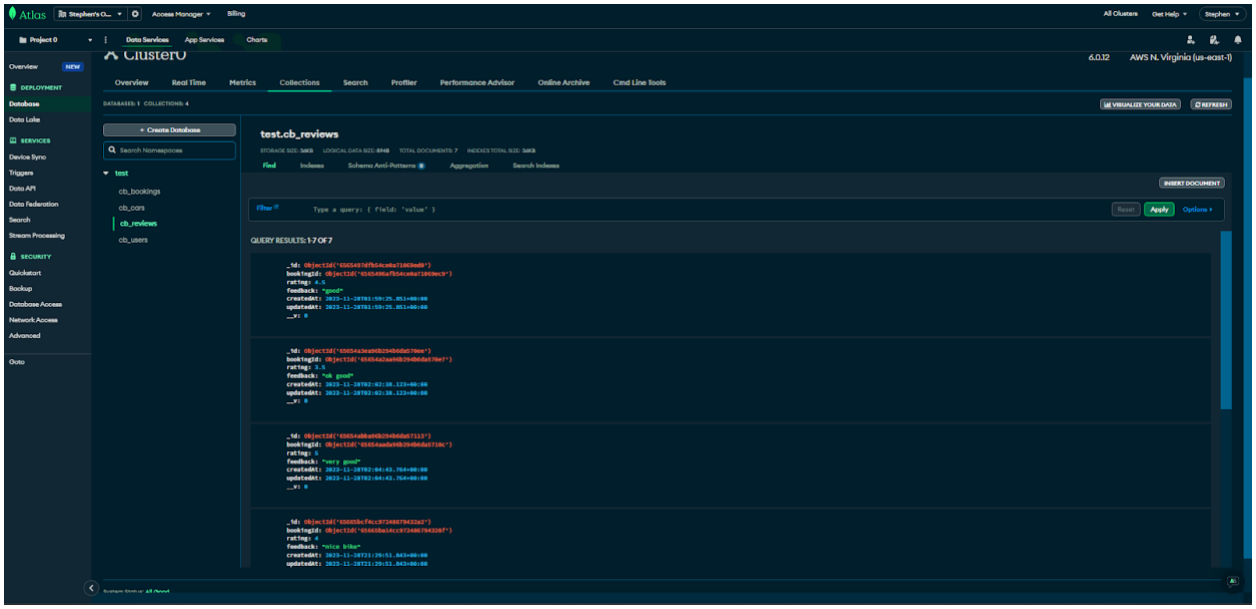

complex-query-1.js
complex-query-2.js
complex-query-3.js
Navbar.jsx M
Booking.jsx .../Car
Button.jsx
Booking.jsx
code > api > scripts > complex-query-2.js > ...
rating: 3,
feedback: "normal"
});
carModel.aggregate([
{
$lookup: {
from: "cb_bookings",
localField: "_id",
foreignField: "carId",
as: "bookings",
}
},
{
$unwind: {
path: "$bookings",
preserveNullAndEmptyArrays: true
}
},
{
$lookup: {
from: "cb_reviews",
localField: "bookings._id",
foreignField: "bookingId",
as: "reviews"
}
},
{
$group: {
_id: {
_id: "$_id",
title: "$title",
make: "$make",
year: "$year",
image: "$image",
},
bookings: { $addToSet: "$bookings" },
reviews: { $addToSet: "$reviews" },
}
}
])
.then(d => console.log(JSON.stringify(d)))
})()
Ln 1, Col 1
Spaces: 4
UTF-8
LF
{} JavaScript

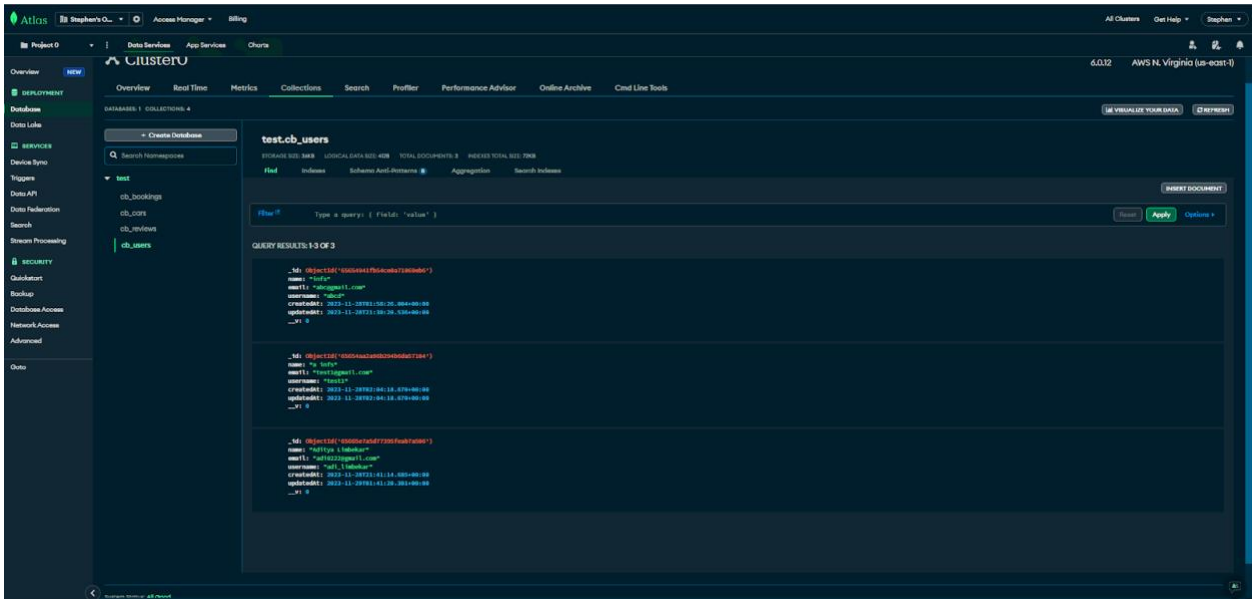
```
JS complex-query-1.js  JS complex-query-2.js  JS complex-query-3.js  Navbar.jsx M  Booking.jsx .../Car  Button.jsx  Booking.jsx

~/Library/CloudStorage/OneDrive-GeorgeMasonUniversity-O365Production/Sem_3/INFS 740/Project/INFS740_Project_vehicle-registration-system/code/api/package.json

    });


    userModel.aggregate([
        {
            $lookup: {
                from: "cb_bookings",
                localField: "_id",
                foreignField: "user",
                as: "bookings"
            }
        },
        {
            $unwind: {
                path: "$bookings",
                preserveNullAndEmptyArrays: true,
            }
        },
        {
            $lookup: {
                from: "cb_cars",
                localField: "bookings.carId",
                foreignField: "_id",
                as: "cars",
            }
        },
        {
            $group: {
                _id: {
                    id: "$id",
                    username: "$username",
                    email: "$email",
                    name: "$name",
                },
                bookings: { $addToSet: "$bookings" },
                cars: { $addToSet: "$cars" },
            }
        }
    ])
    .then(d => console.log(JSON.stringify(d)));
})()

Ln 1, Col 1  Spaces: 4  UTF-8  LF  {} JavaScript
```

- **MongoDB Atlas**




Atlas
Cluster0
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
test.cb_cars
cb_bookings
cb_cars
cb_reviews
cb_users
QUERY RESULTS: 1-3 OF 3


Atlas
Cluster0
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
test.cb_reviews
cb_bookings
cb_cars
cb_reviews
cb_users
QUERY RESULTS: 1-7 OF 7




- **Contributions**
  → **"Vehicles", "Add a Vehicle", "Login", "Signup"** components (UI as well as the respective backend functionality along with queries) – **Aditya Milind Limbekar**

  → **"My Bookings", "All Bookings", "Settings", "Log Out"** components (UI as well as the respective backend functionality along with queries) – **Stephen Dias**