ST10108446 Sibusiso Dlamini

APDS API Testing document

POE Final

Frontend and Backend

# Coverpage

[Coverpage 1](#_Toc759094073)

[MongoDB Database Setup 3](#_Toc1924770179)

[Key and Certificate Generations 4](#_Toc443607051)

[Get from Database 4](#_Toc1491317293)

[Creating new posts in DATABASE 5](#_Toc1722223974)

[DELETE from Database 7](#_Toc1835584684)

[Users 8](#_Toc852101044)

[Frontend 10](#_Toc1642387043)

[API: Get all posts 11](#_Toc707790751)

[API: Register a new users 11](#_Toc920948637)

[API: Logon user A 12](#_Toc405349940)

[API: Create a new post User A 12](#_Toc1964860020)

[API logon User B 13](#_Toc132401031)

[API: Create a new post User B 13](#_Toc124812830)

[User B delete user A’s post 14](#_Toc1655073137)

[User A delete user B’s post 15](#_Toc533299088)

[Security packages 16](#_Toc1741827598)

[Implementing express-brute, helmet, Morgan 17](#_Toc1028233360)

[Custom Error Messages 17](#_Toc158625100)

[Security input field validation, password obscuring and sanitations 18](#_Toc1161584153)

[Service Auth 18](#_Toc922695743)

# MongoDB Database Setup

This section will be showing illustrations of the created database and tables

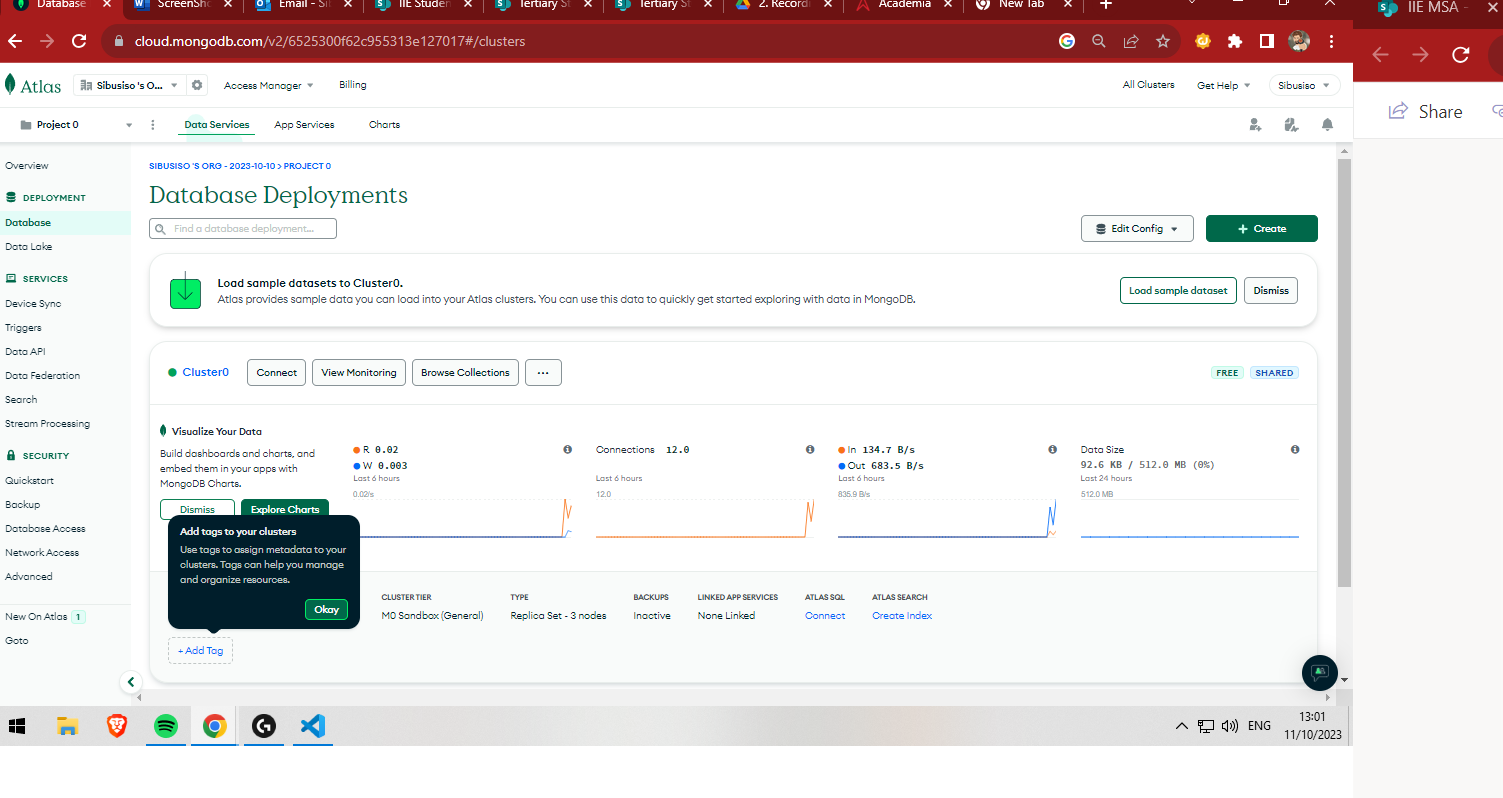


Figure1 : Mongo DB Setup

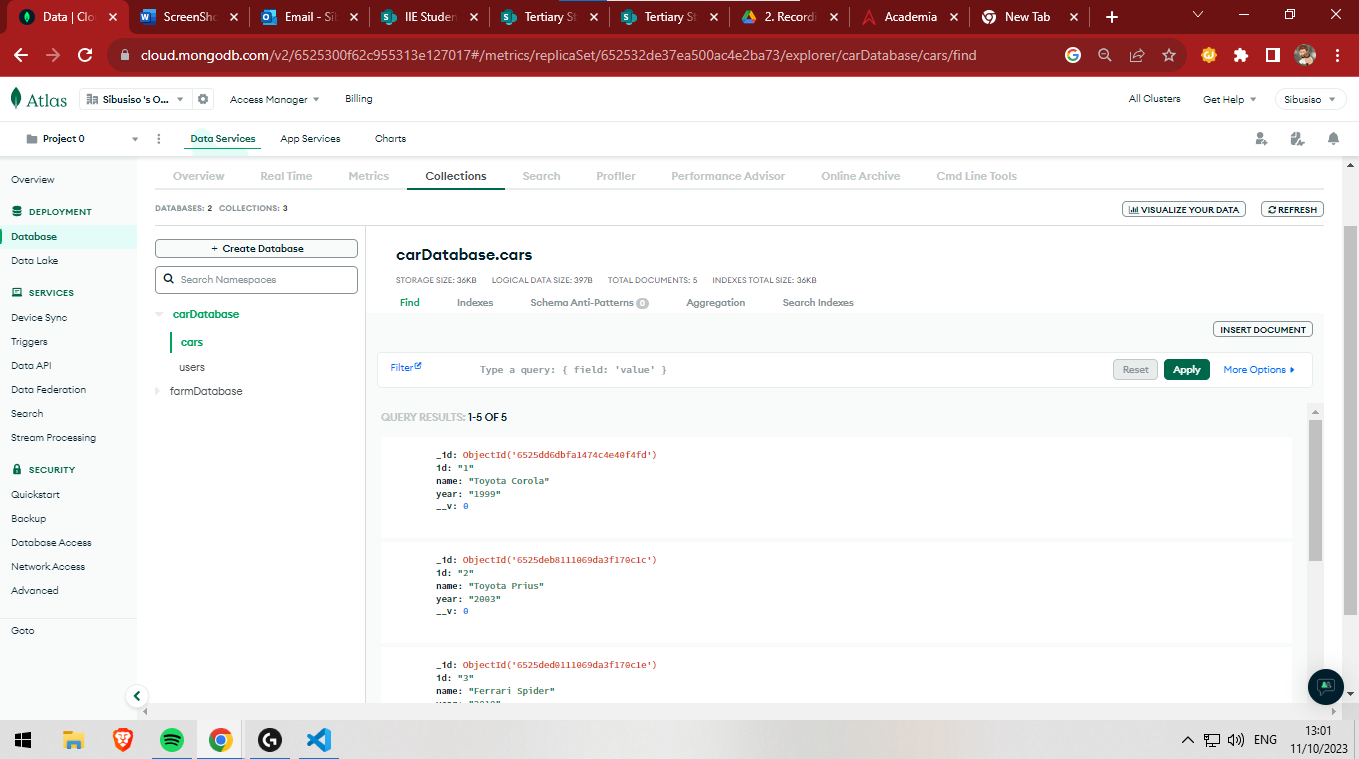


Figure 2: Database created

# Key and Certificate Generations

This section will be showing illustrations of how the private key and certificate was created

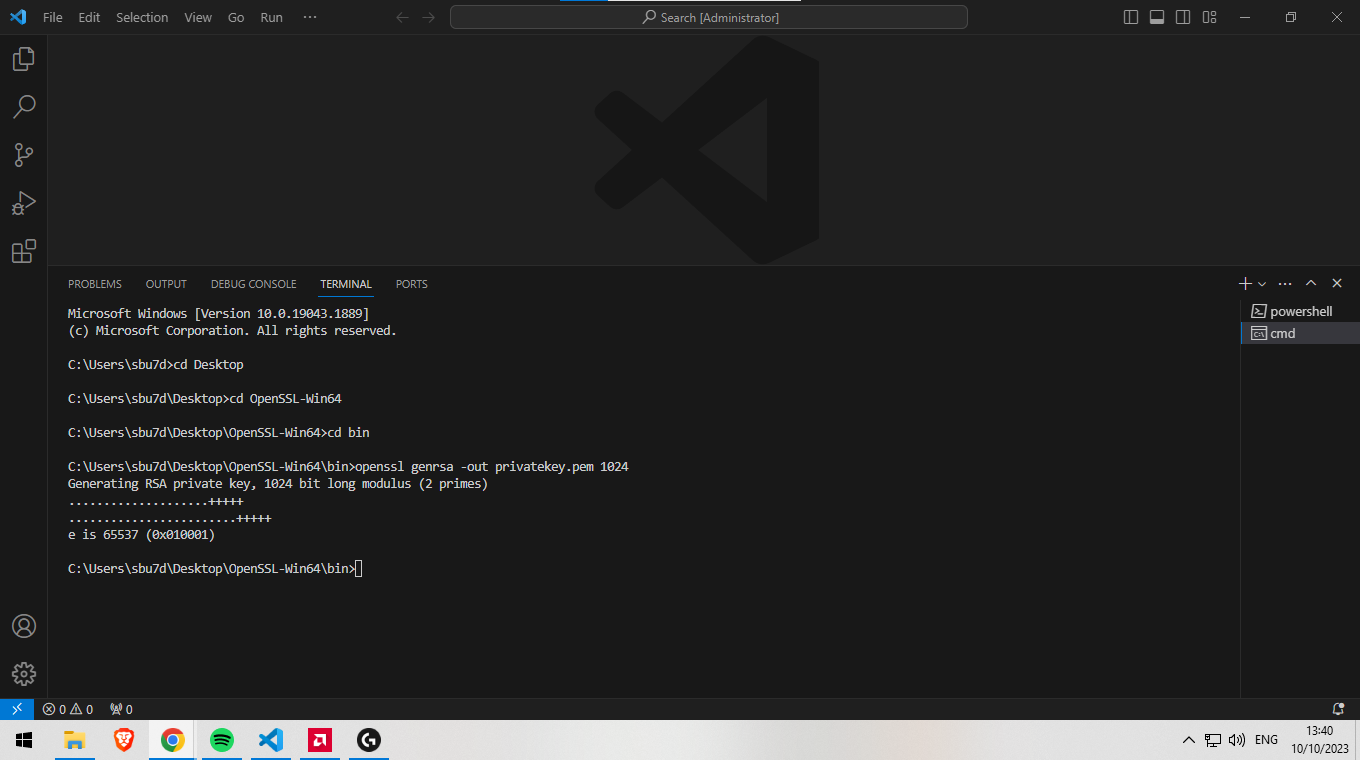


Figure 3: Private Key Generations

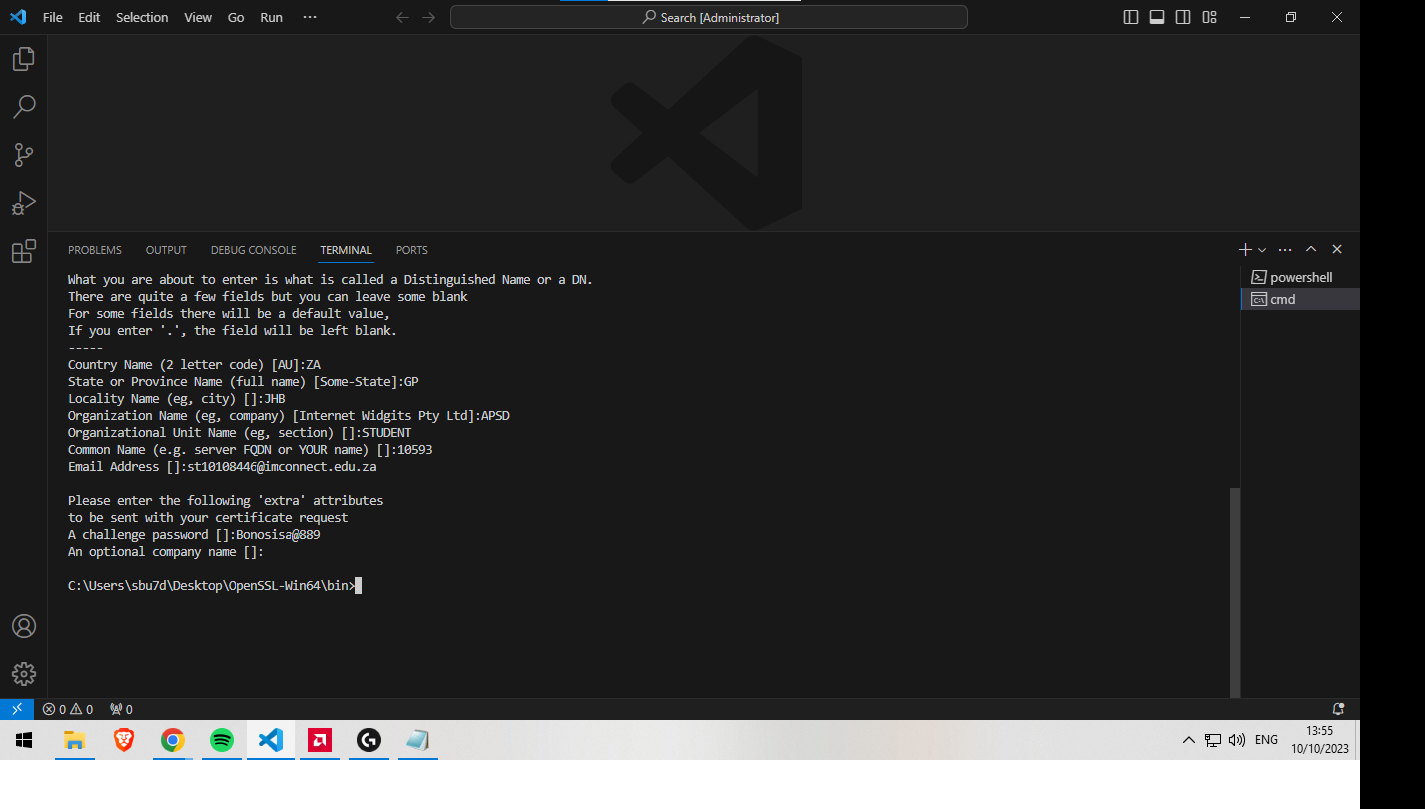


Figure 4: Certificate sign in request

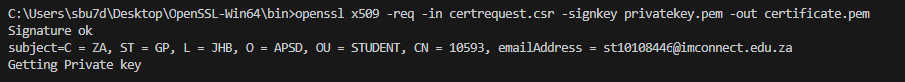


Figure 5: Signing Certificate using private key

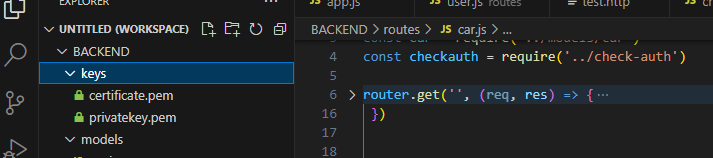


Figure 6: Key and Certificate in keys folder

# Get from Database

This section will be showing illustrations of the get method (which gets list of cars in the database)

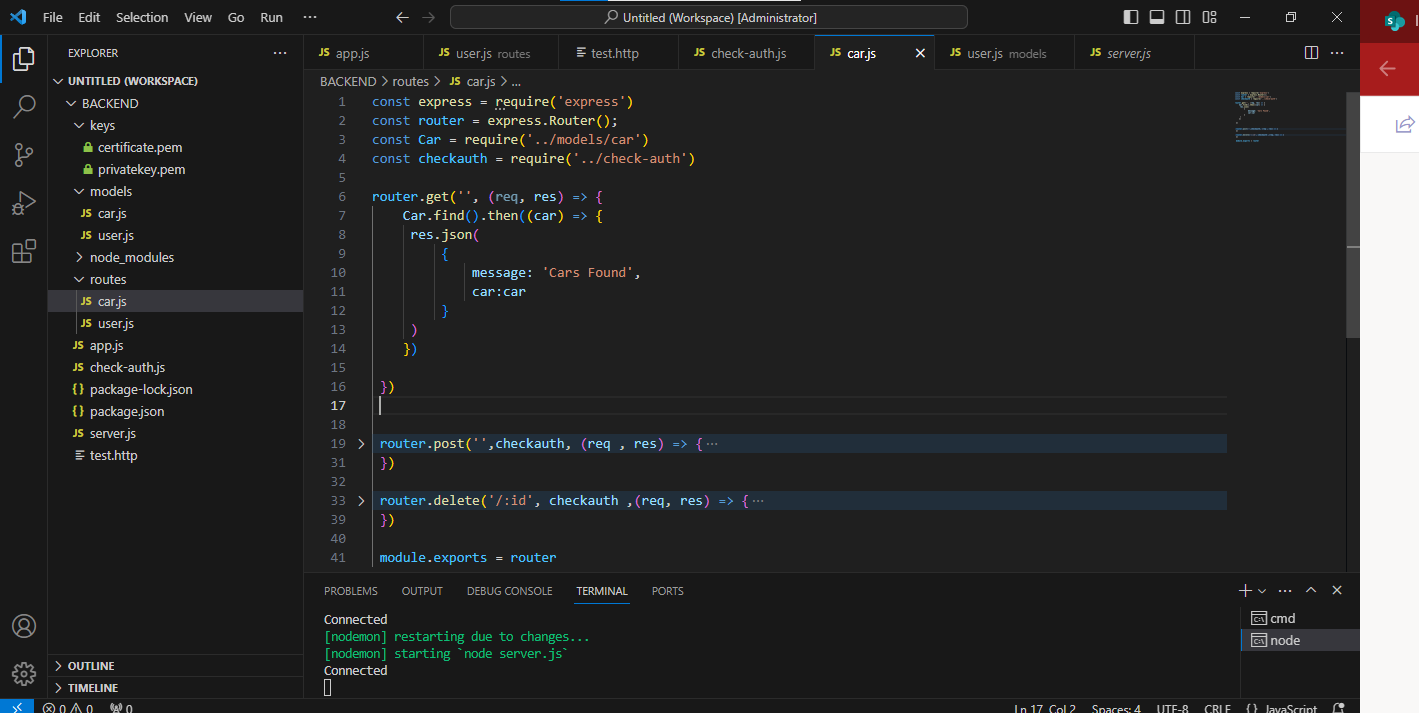


Figure 7: Code for getting Cars in database

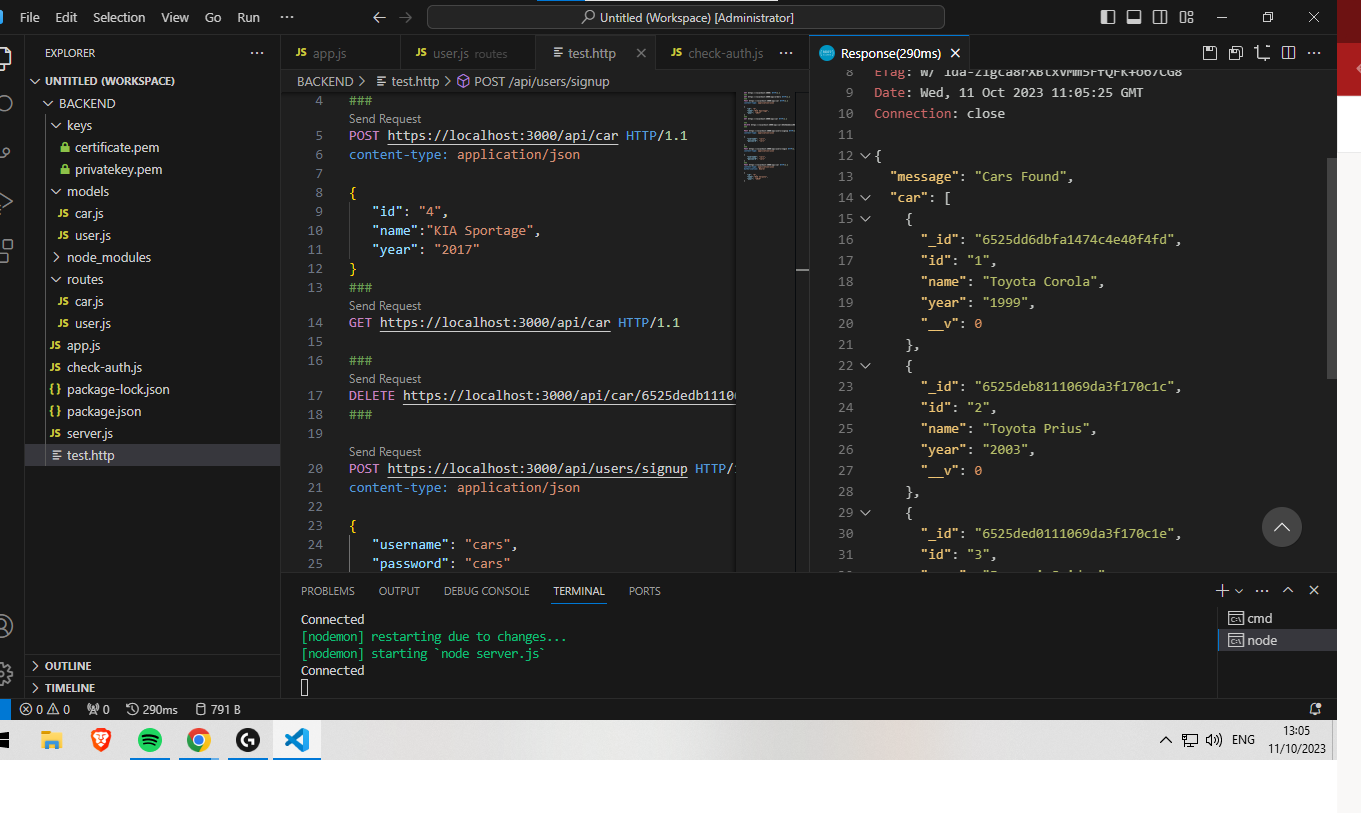


Figure 8: Showing test code for getting cars from database

# Creating new posts in DATABASE

This section will be showing illustrations of the post method (which will post new cars into the database)

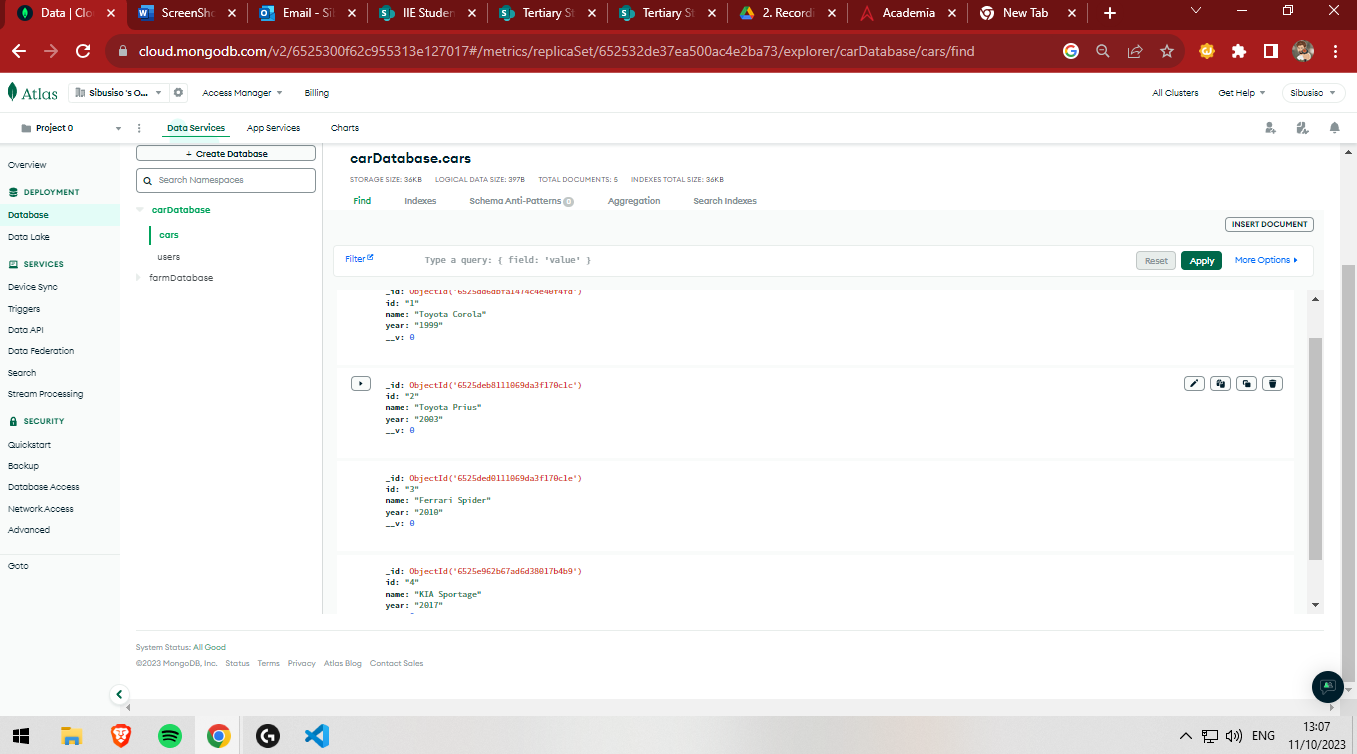


Figure 9: Database Pre-post

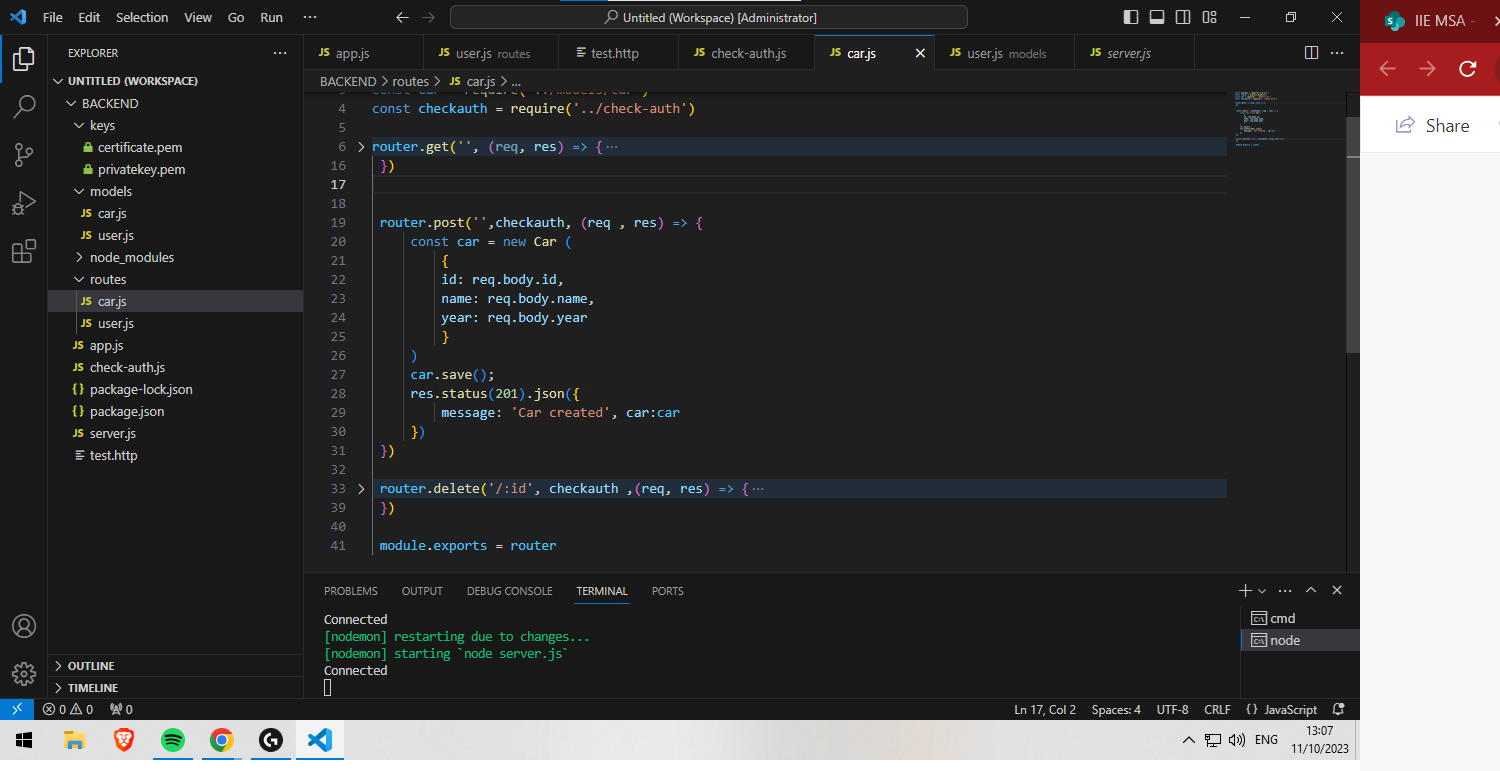


Figure 10: Route to Code for posting

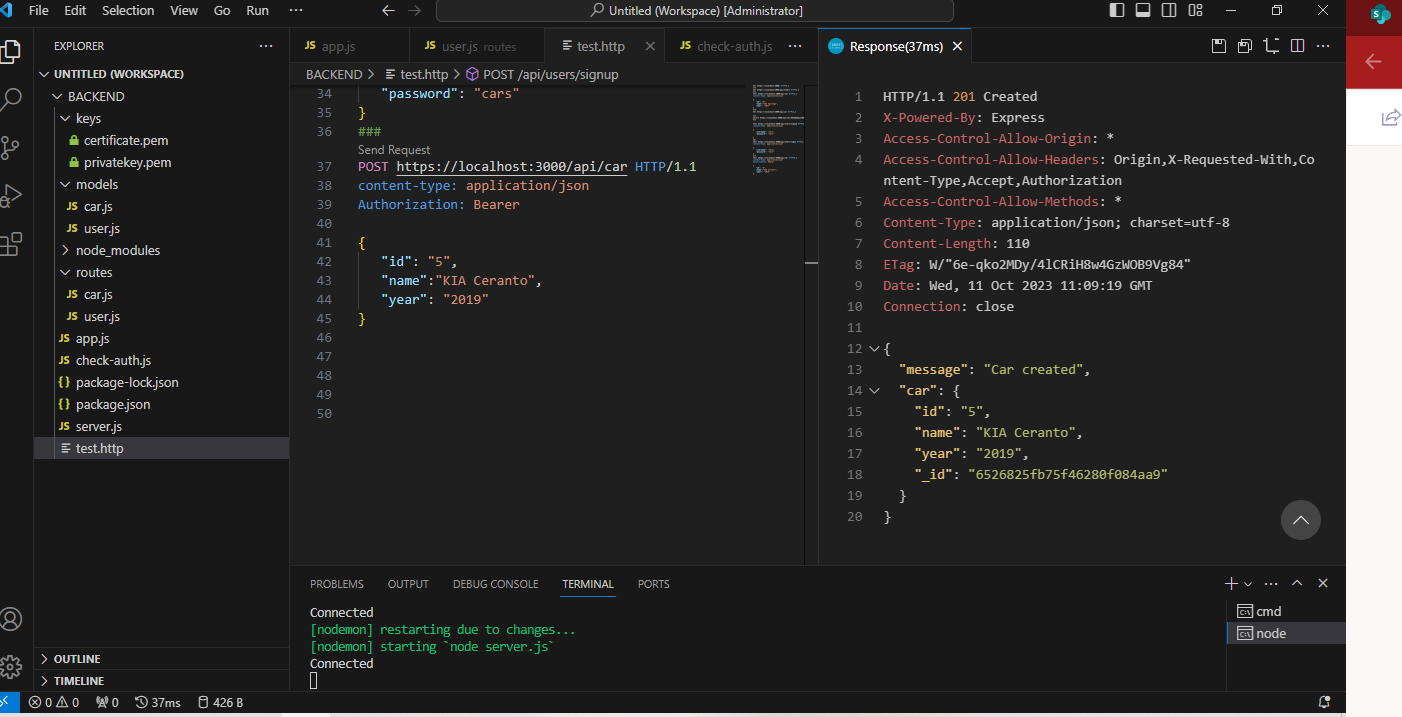


Figure 11: Posting to Database

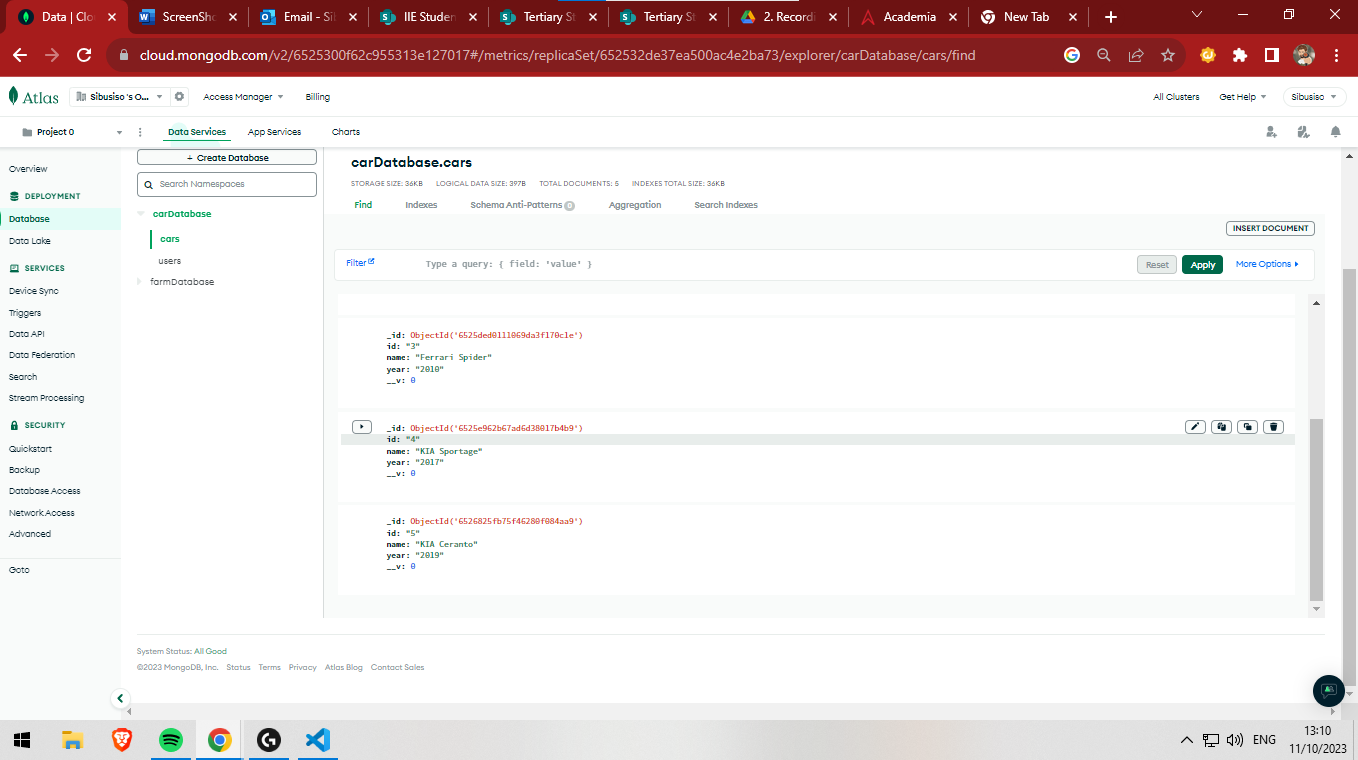


Figure 12: Database after post (before delete function)

# DELETE from Database

This section will be showing illustrations of the delete function (this allows users to delete cars from database)

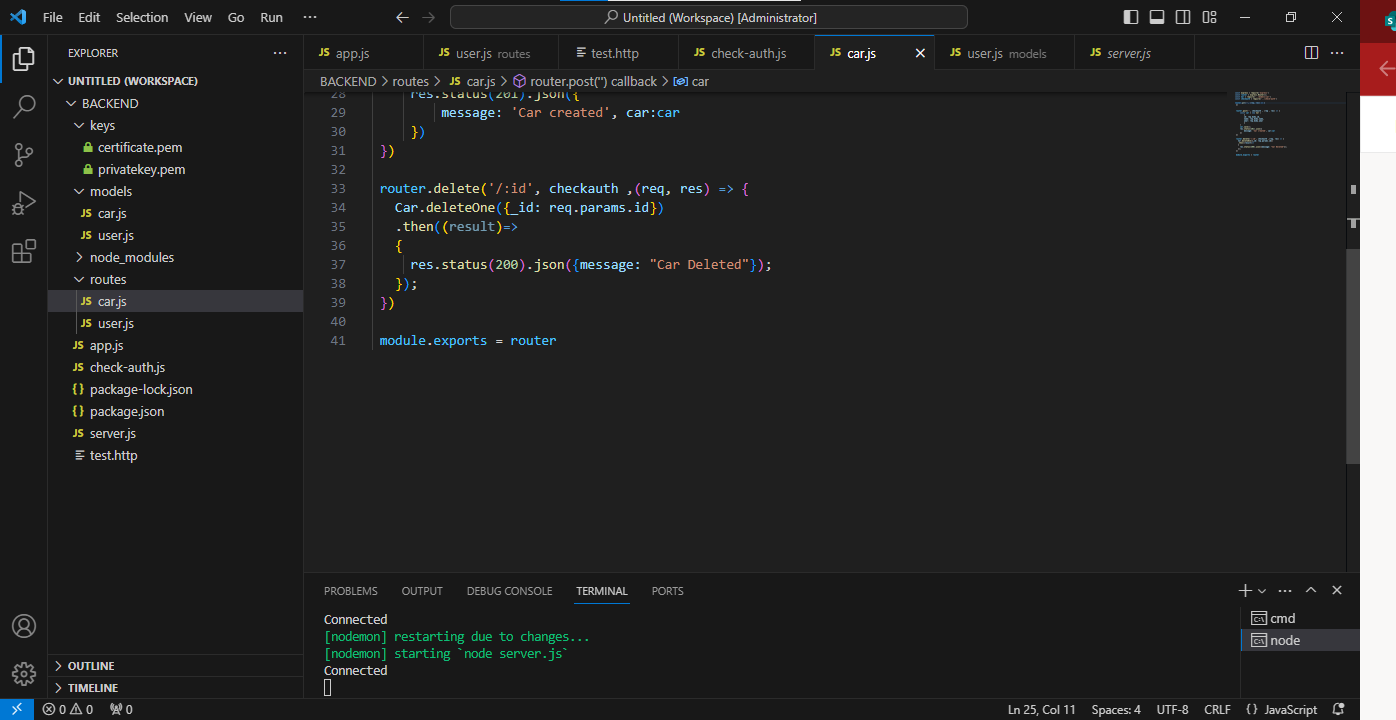


Figure 13: Route to Delete code function

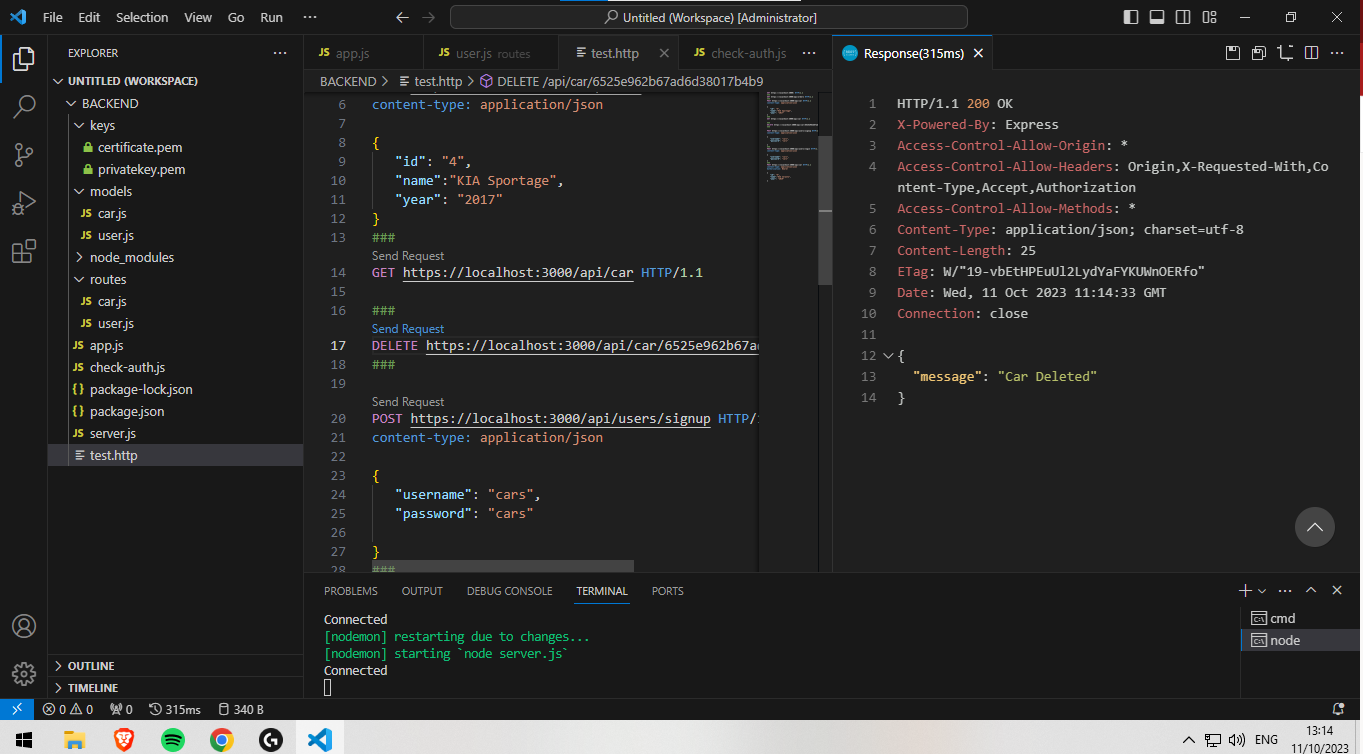


Figure 14: Running delete code for item ‘6525e962b67ad6d38017b4b9’ Kia Sportage, ID 4

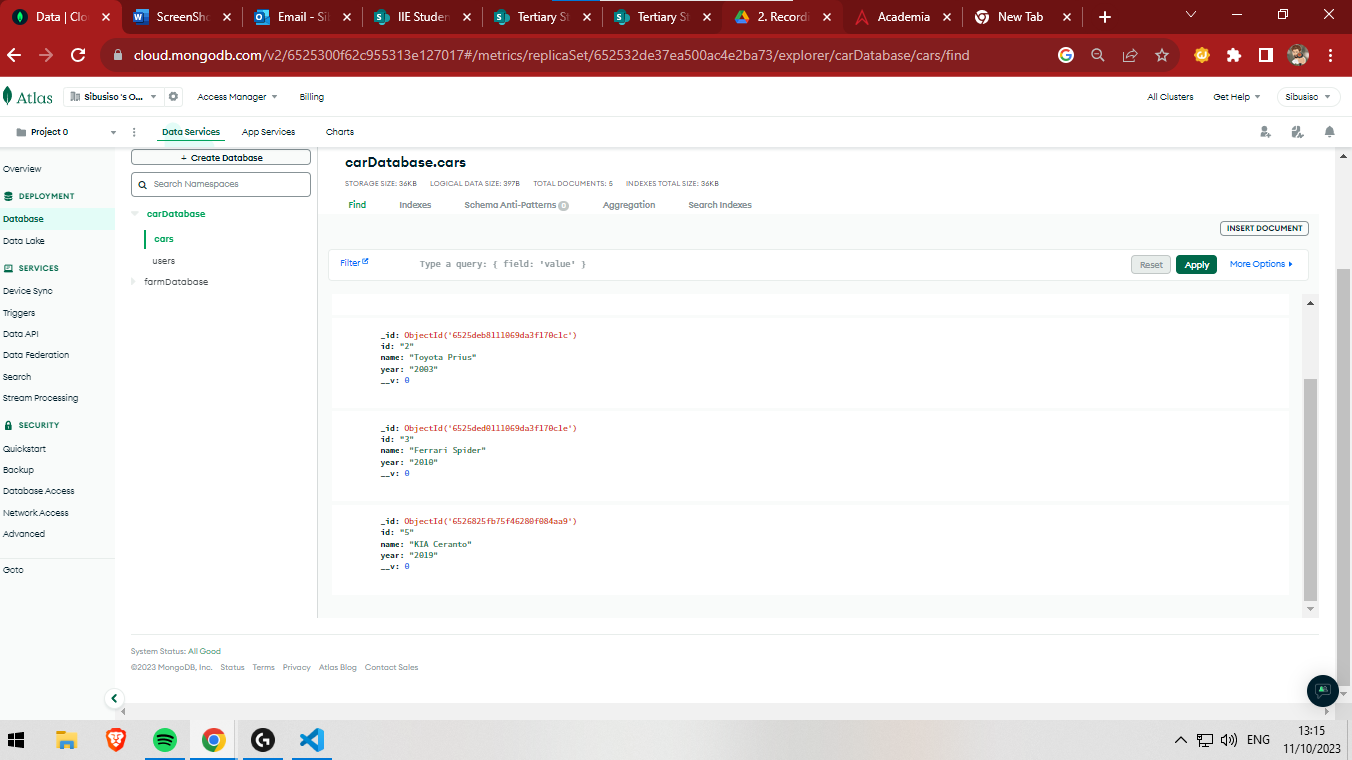


Figure 15: Database After Delete code

# Users

This section will be showing illustrations of how users will signup (added to database) and login

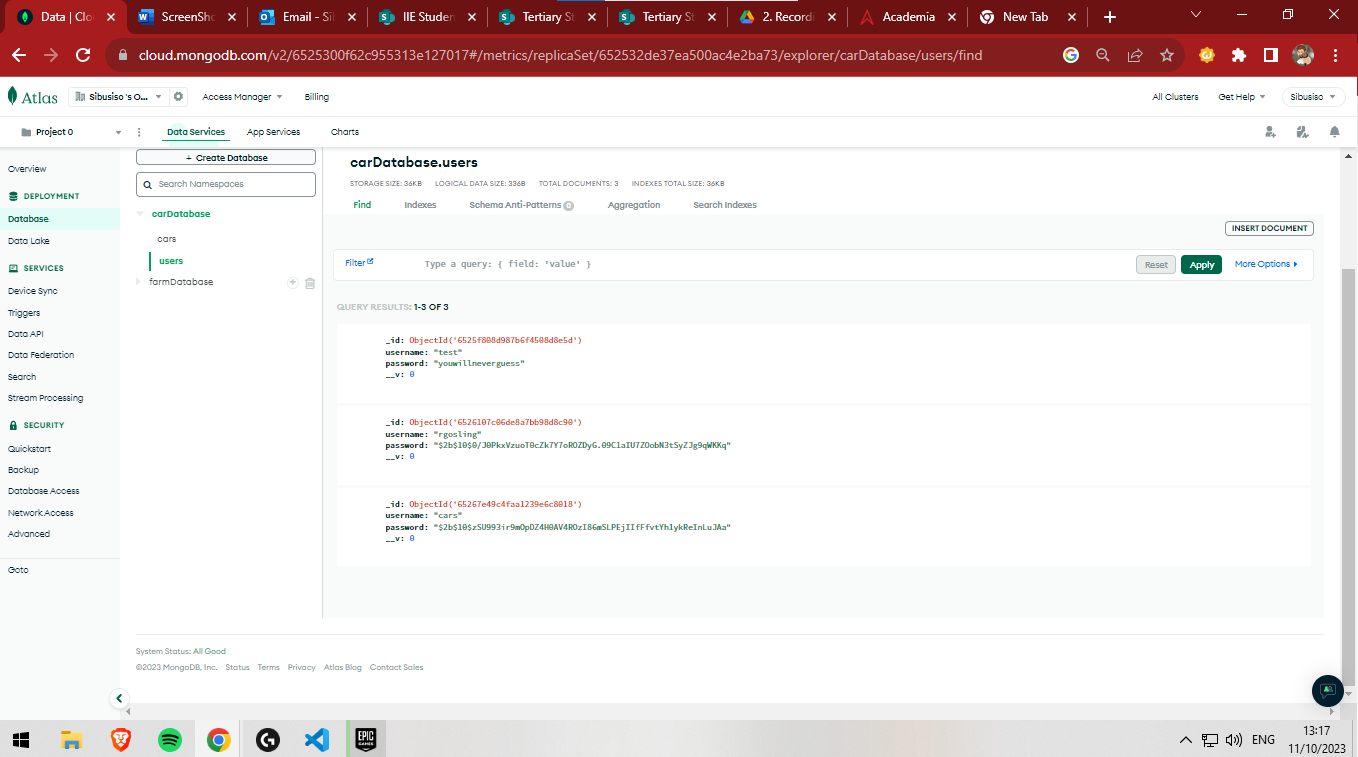


Figure 16: Users table in Database

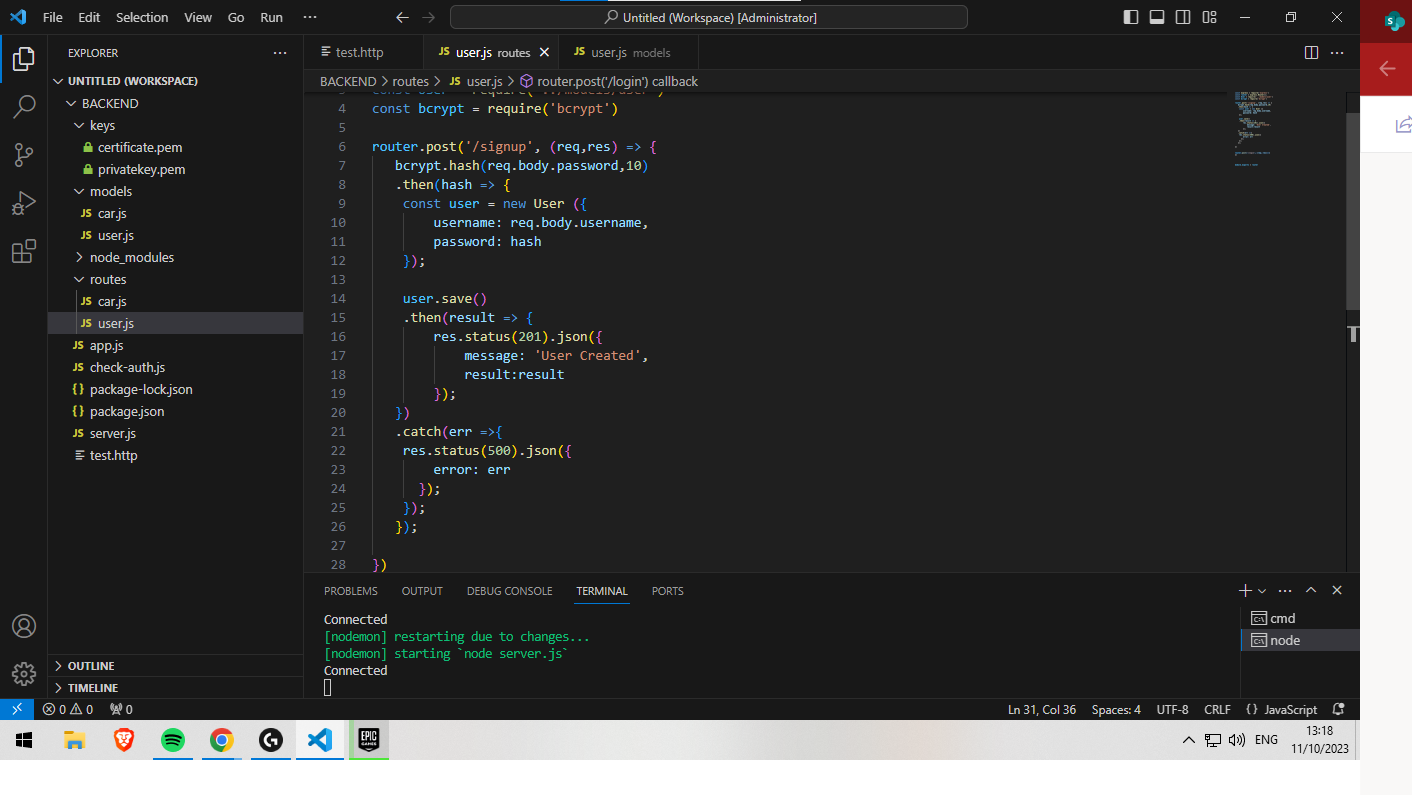


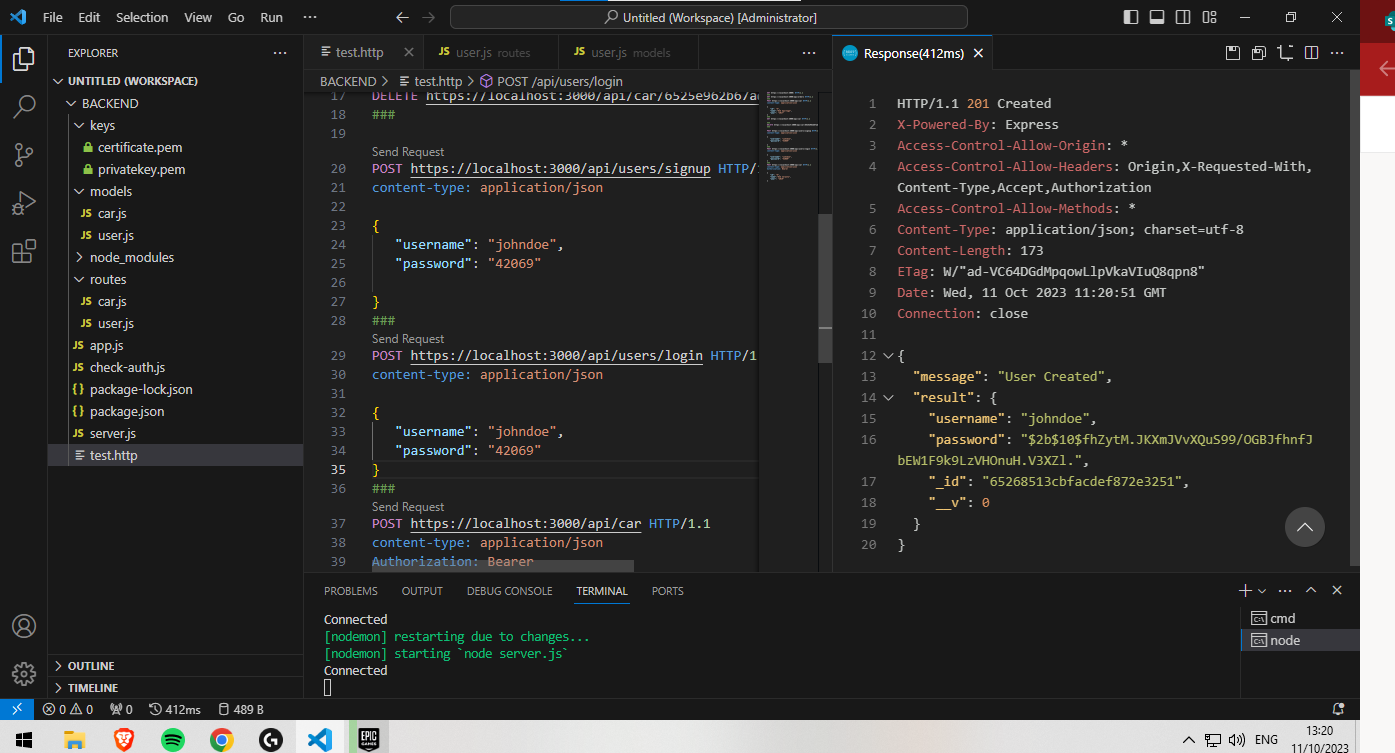
Figure 17: Route to code to sign up a new user

Figure 18: Code signing up a new user ‘johndoe’

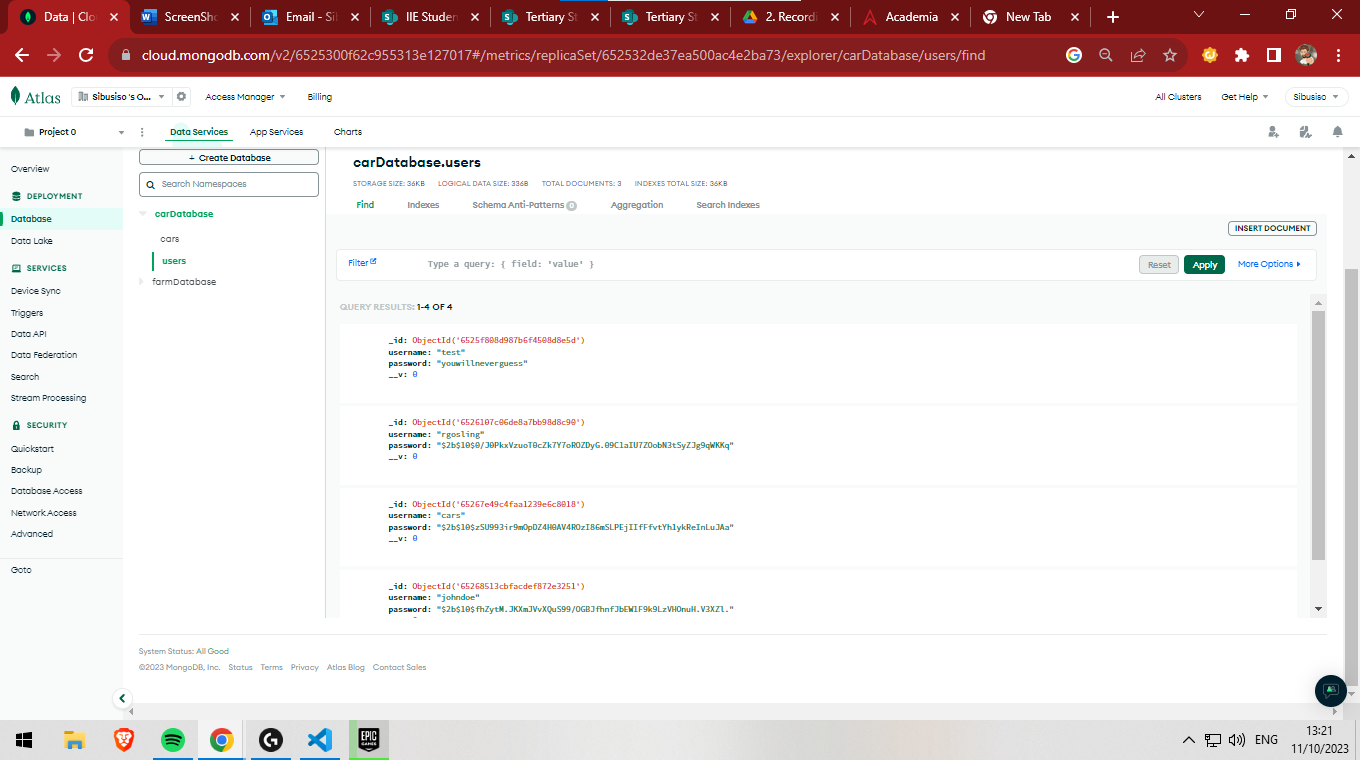


Figure 19: Database after new user is added

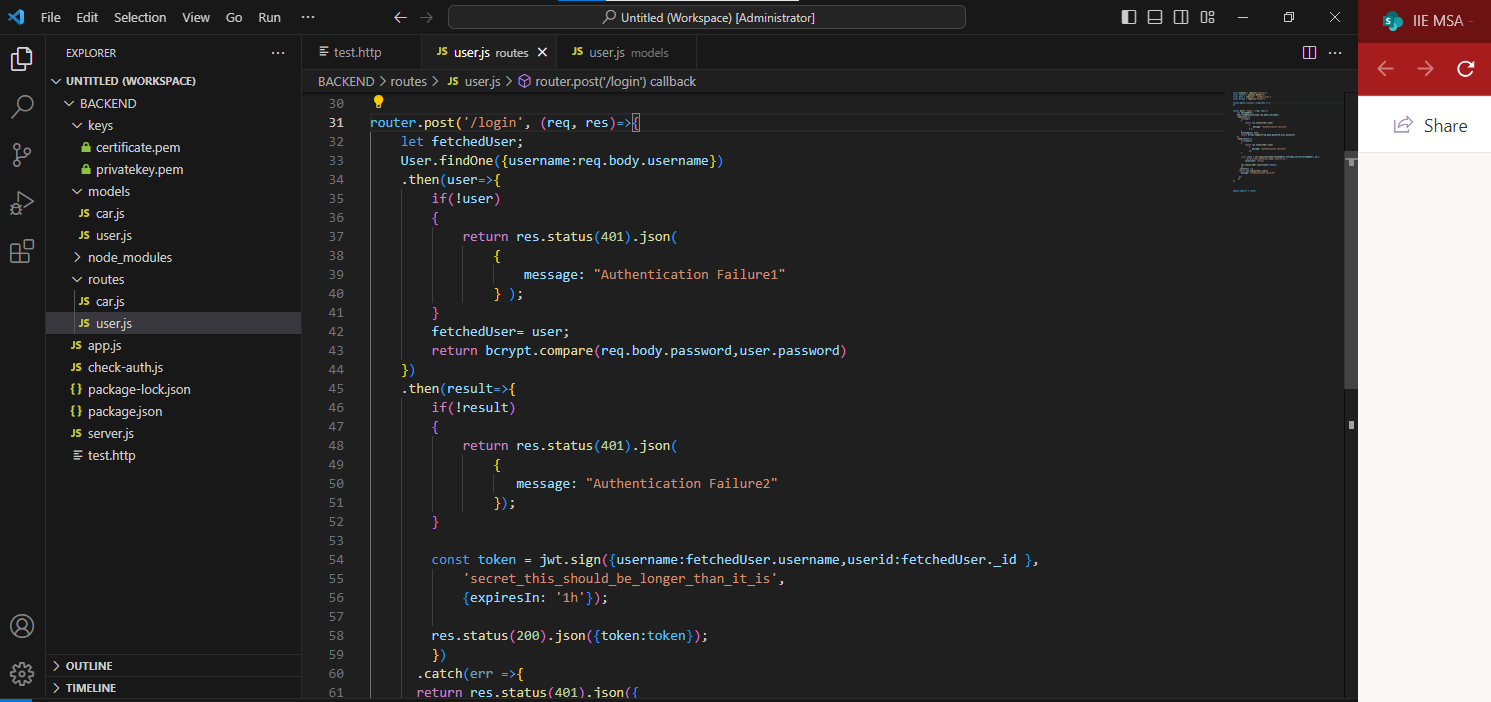


Figure 20: Code for login into application

# Frontend

## API: Get all posts

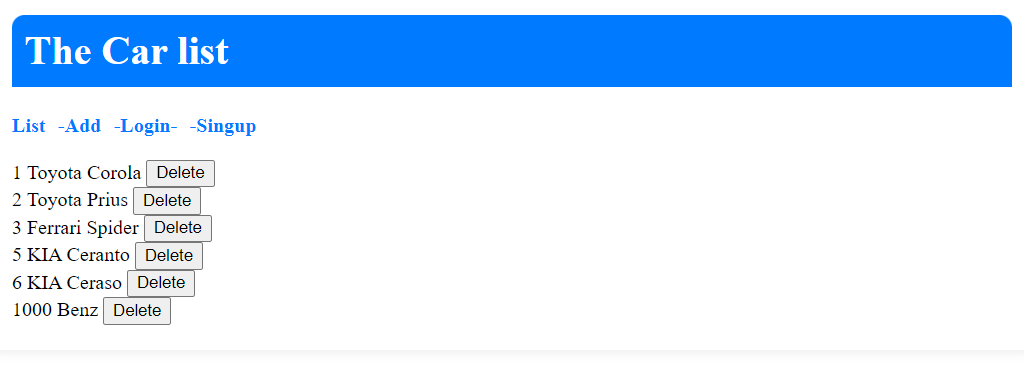


Figure 21: Shows list of cars from databaste

## API: Register a new users

Invalid fields

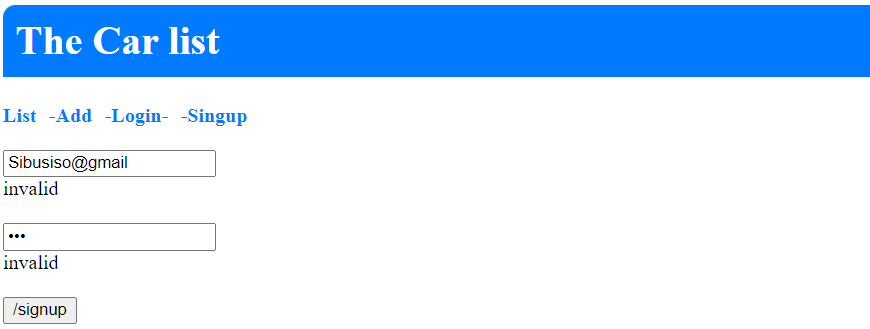


Figure 22: Shows error checking for invalid inputs

Valid Fields

User A



Figure 23: Shows the creation of new user A

User B

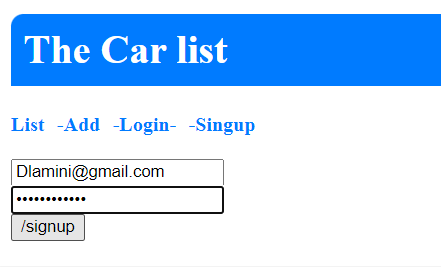


Figure 24: Shows the creation of new user B

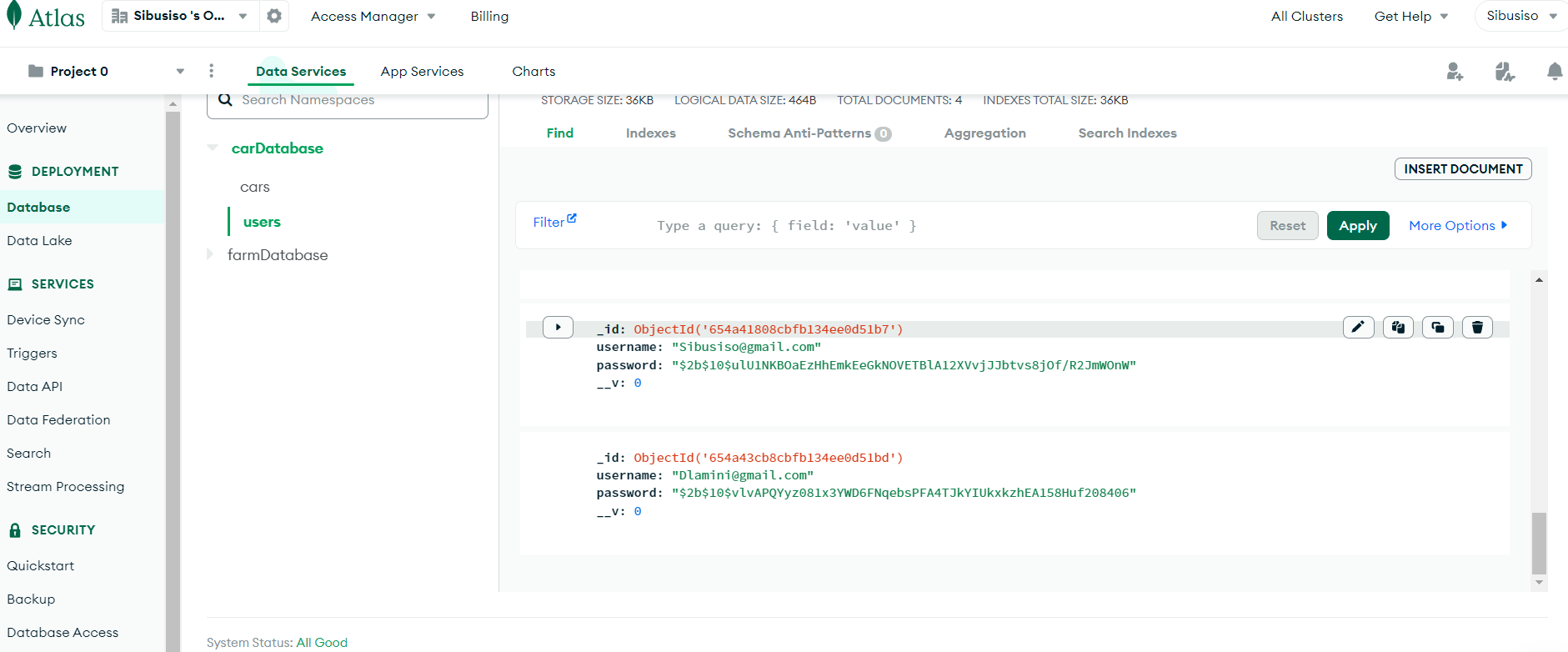


Figure 25: shows the created users in MongoDB

## API: Logon user A

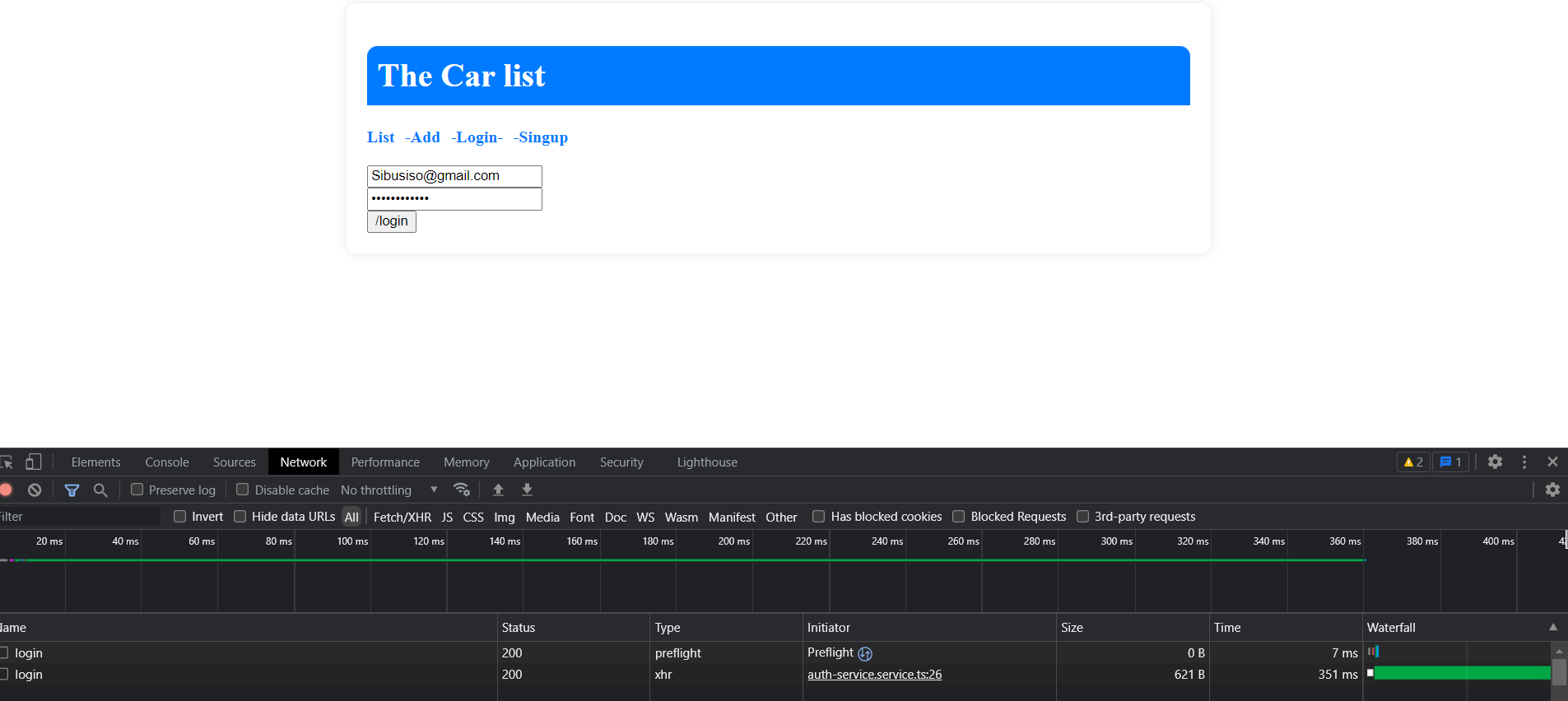


Figure 26: Shows user A logging in

No Error persist so user has been logged on

## API: Create a new post User A

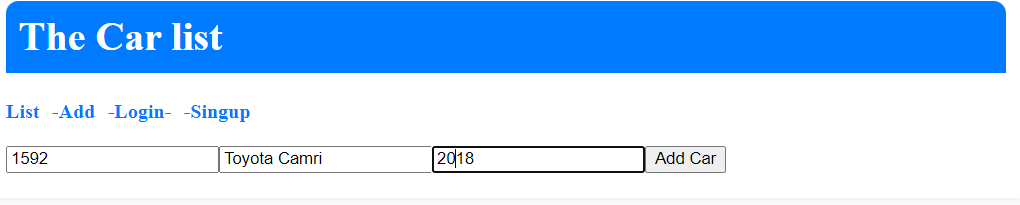


Figure 27: shows user creating a new post



Figure 28: Confirmation of pose

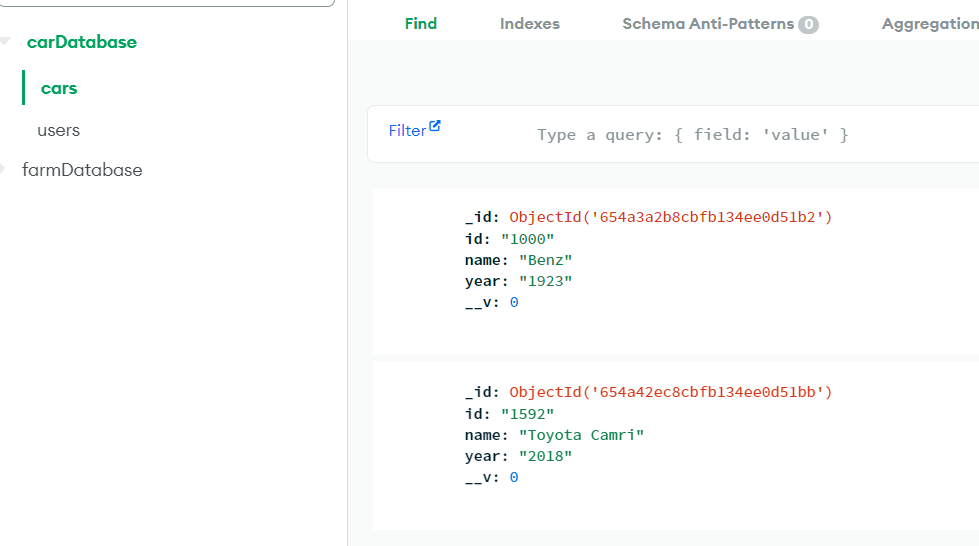


Figure 29: Shows the created post in MongoDB

## API logon User B

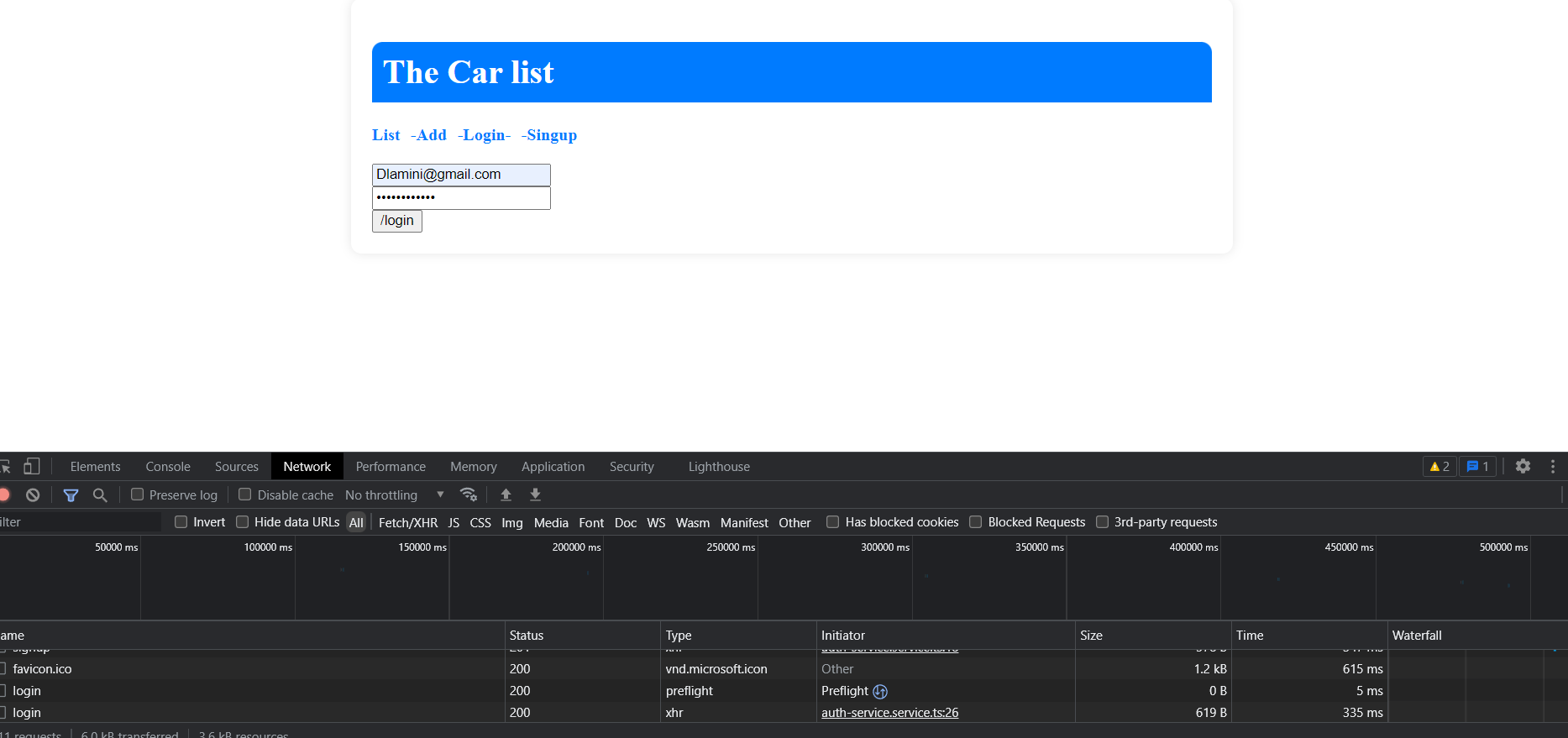


Figure 30: Shows user B logging in

No Error persist so user has been logged on

## API: Create a new post User B

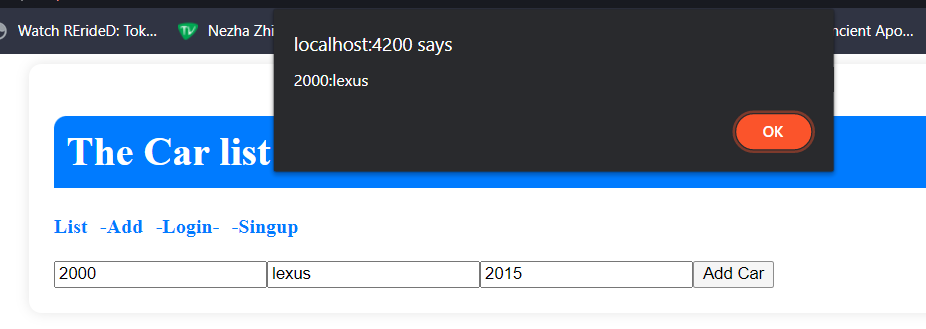


Figure 31: Shows user B creating a new post

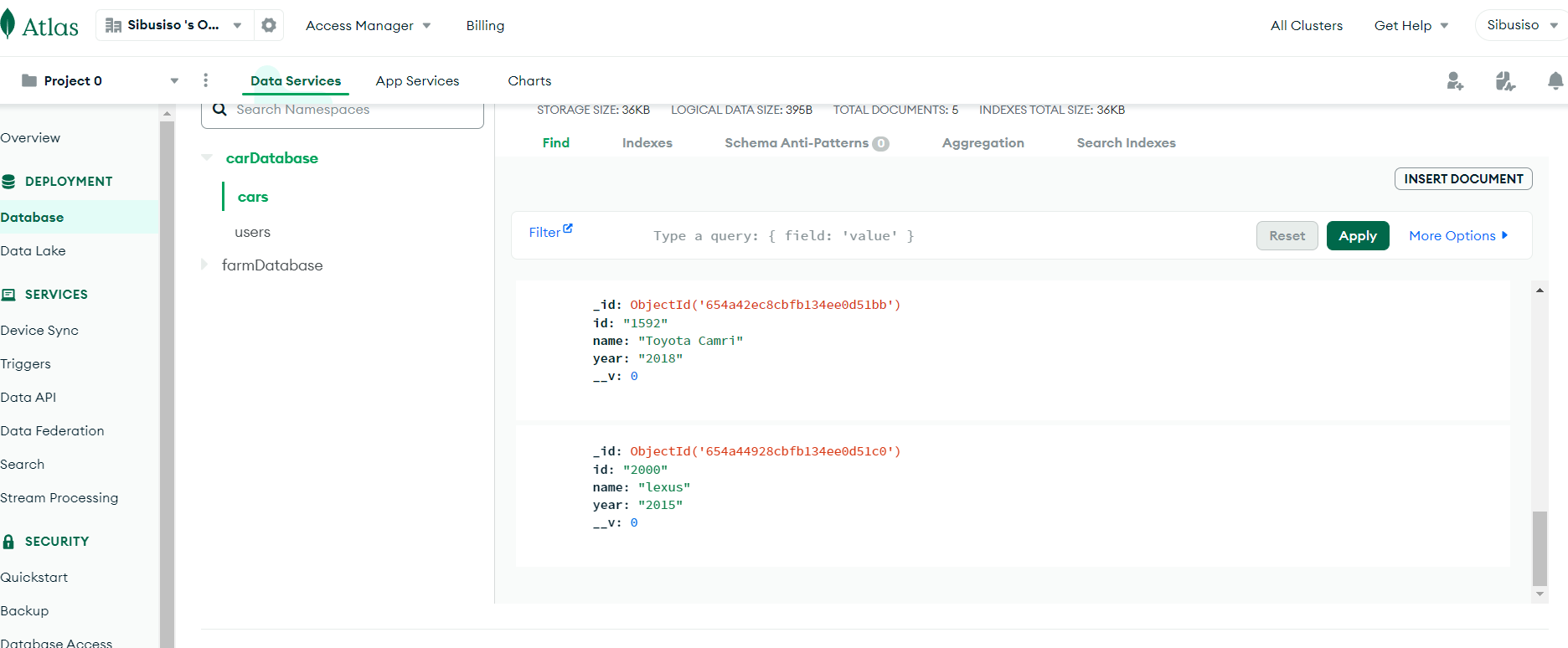


Figure 32: Shows user B’s post in MongoDB

## User B delete user A’s post

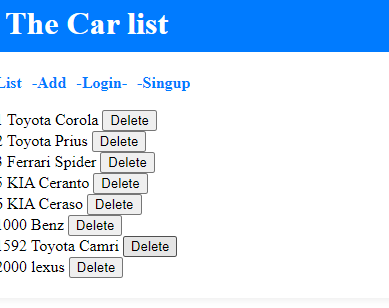
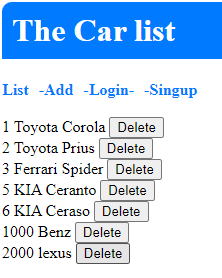


Figure 33:Shows the car list before deletion has taken place

Figure 34: Shows Toyata Camri post has been Deleted

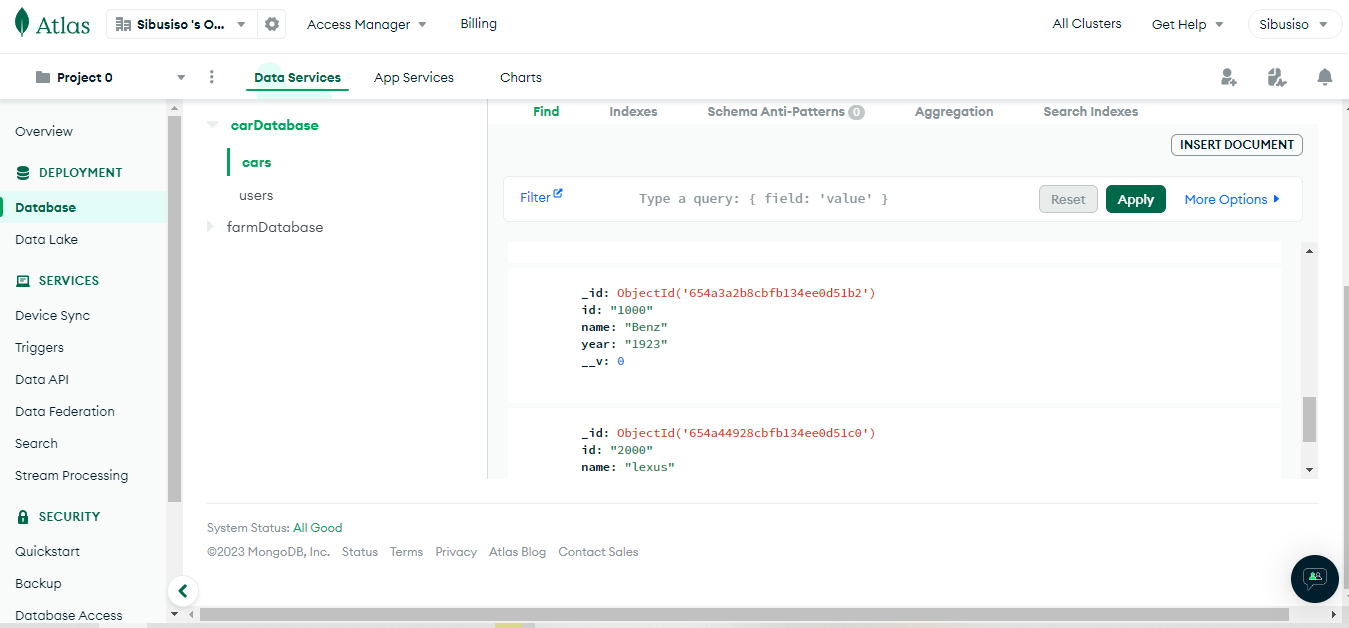


Figure 35: Shows Toyata Camri post has been Deleted from MongoDB

## User A delete user B’s post

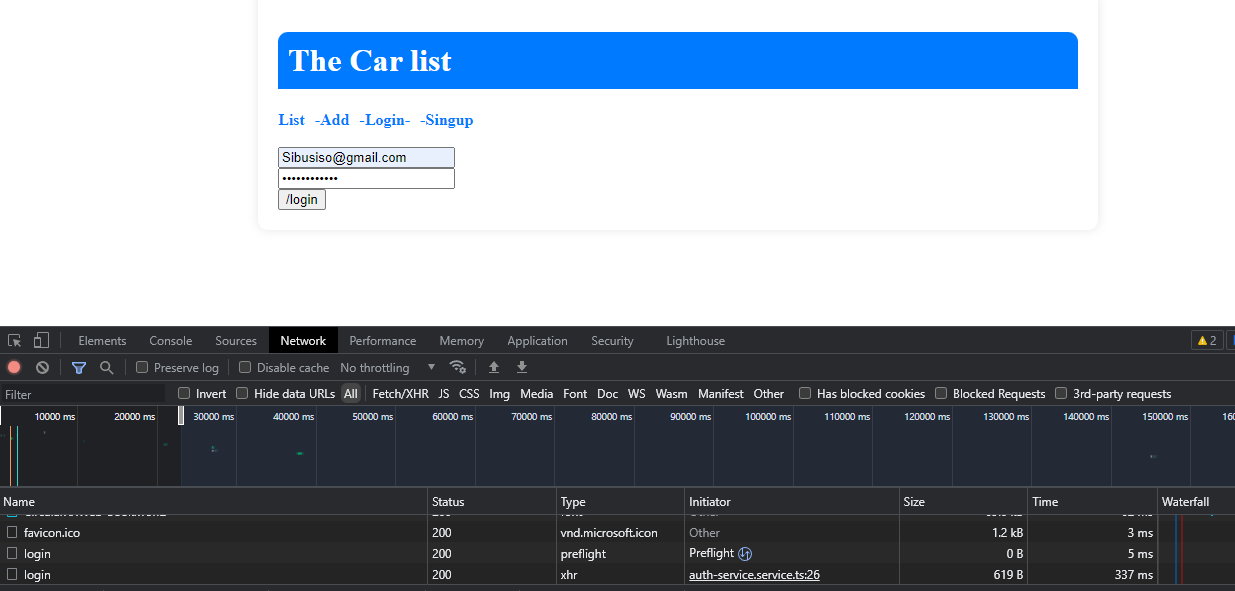
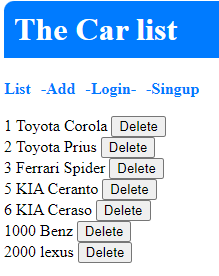
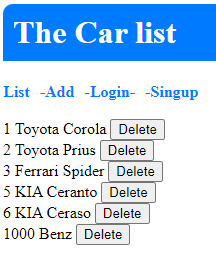


Figure 36: Shows user A logging in again

Figure 36: shows car list before deletion

Figure 37:shows car Lexus has been deleted from the list

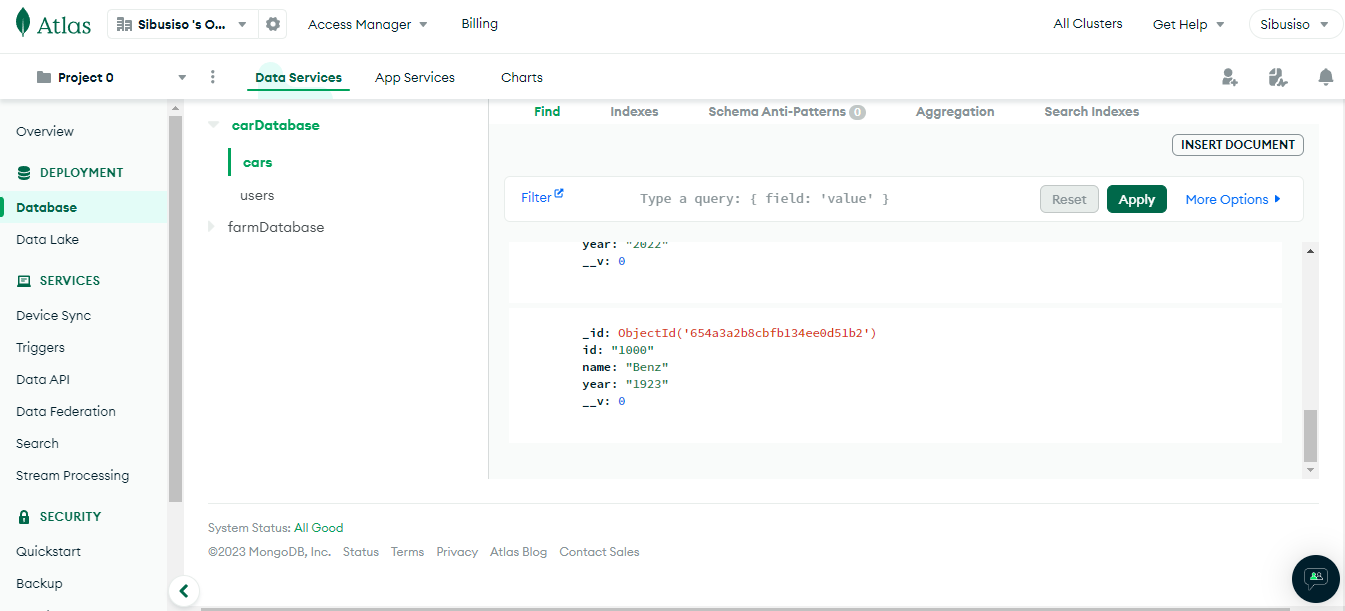


Figure 38: shows car Lexus has been deleted from MongoDB

# Security packages

## Implementing express-brute, helmet, Morgan

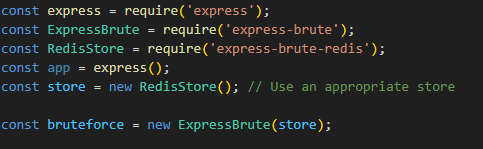


Figure 39 : Shows the implementation of express-brute in the backend workspace

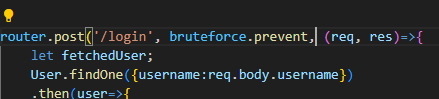


Figure 40: same as figure 39

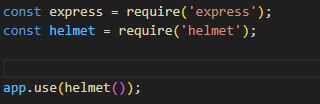


Figure 41: Shows the implementation of helmet in the backend workspace

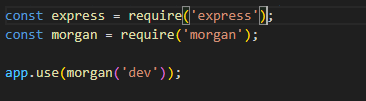


Figure 42: Shows the implementation of Morgan in the backend workspace

# Custom Error Messages

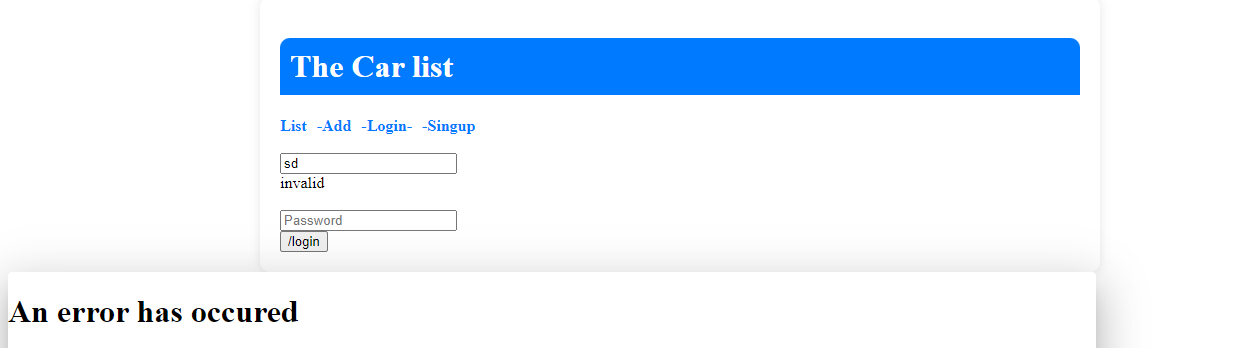


Figure 43 : shows the custom message for when an error is caught

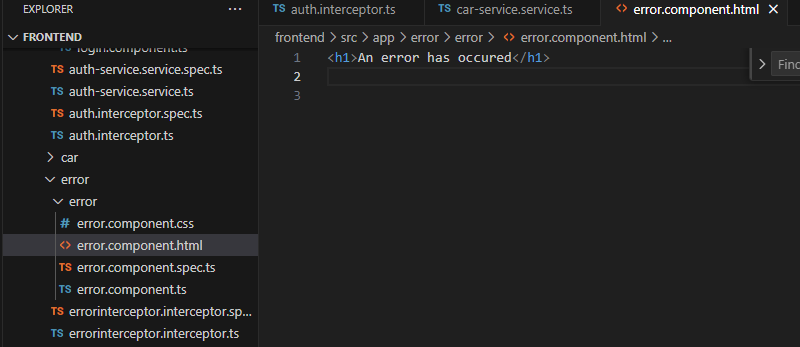


Figure 44: shows the code for custom error messages

# Security input field validation, password obscuring and sanitations

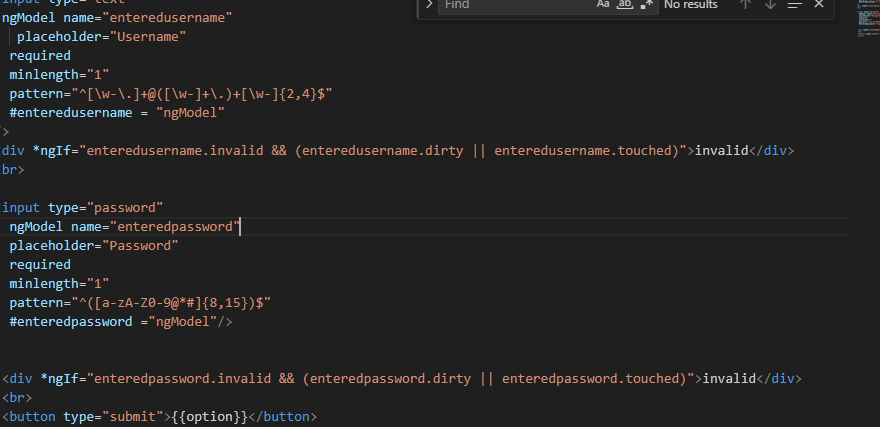


Figure 45: shows the code for input field validation(for username and password), password obscuring(hides password ‘\*\*\*\*\*’) and sanitations (enteredpassword.dirty).

# Service Auth

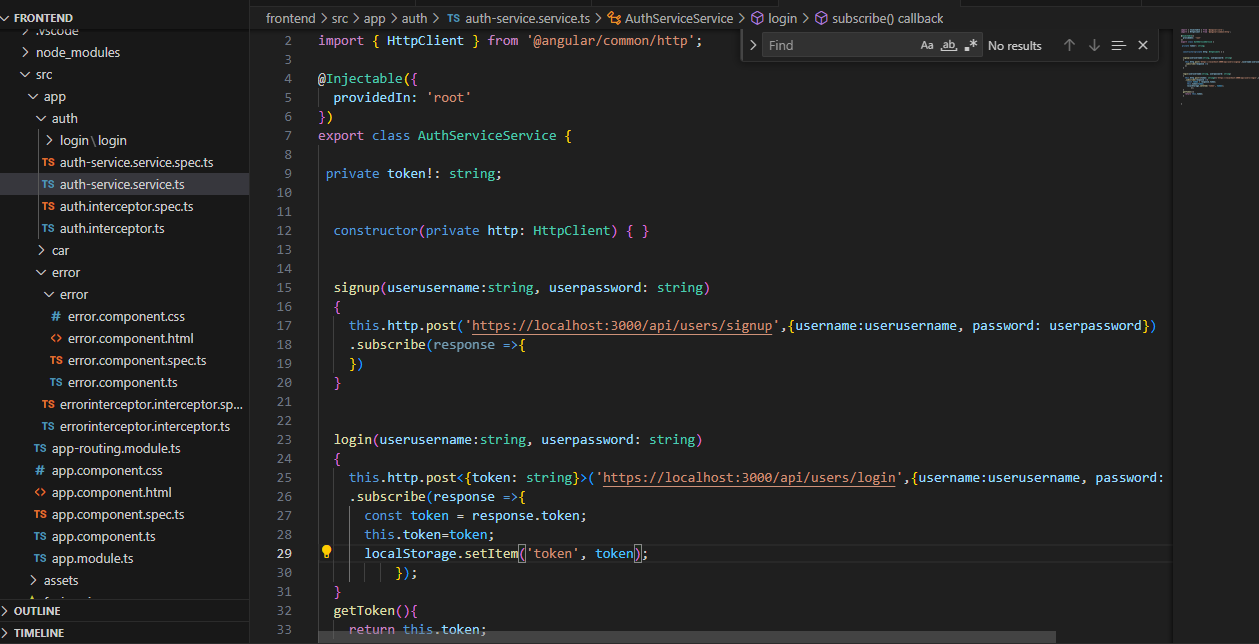


Figure 46 : shows the auth service files code in the front

**No table of contents entries found.**

end