COM519_AE1

Link to github repository: https://shrouded-caverns-19836.herokuapp.com/
Hosted on Heroku: https://github.com/Radoslav22/COM519_AE1
Contributer: **Radoslav Takvoryan**

How to Setup the Project

- Download the zip file of the project.
- Unzip it wherever you want.
- Open the folder using Visual Studio Code.
- Open new terminal.
- Run

```
npm install
```

- After that run

```
npm run dev
```

- In the terminal you will see Example app listening at: The URL of the application. Press ctrl and click on the URl to start the application.

Introduction

Cars are one of the most important factors in our society, nowadays. Every day, a lot of cars are sold, but not many people know how much they cost. My application aims to provide information about BMW price lists so that the people can compare the prices. Therefore, they will be able to decide at what price to sell their cars, and after they sell the cars, they can upload the data of the vehicles to my applications. Currently, the database contains around 4300 records only for BMW, however, they can upload other car makers.

System Overview

My application aims to provide information on what model is being sold for what price based on how many miles are driven. The primary function of my application is to carry out the CRUD functions (create, read, update and delete). The application is using MVC structure. Diagram
The application is separated into three main parts Model, Views, and Controllers (MVC). The Model creates and reads from the database. The views are displayed to the user which contains all of the functionality that interacts with the User. However, between the Model and the Views are the Controllers. The controllers

convert the user input from the views into requests to retrieve or update data using the model. The application is using non-relational Database MongoDB on one collection.

Key Design Decisions

I have tried to do my application user-friendly and created a navigation bar from where all of the users can navigate through the views. I have three main views, search, list, update. Nav-bar
Under my search view, I have created a search panel that shows the record list for which I have limited them to 10 per page so that it is easier to read. It is only to search by model. If you type model the list will be available only with the model you have typed, otherwise all records will be displayed. Also, I have implemented Regex for the search text, because there is some model with letters. SearchView On the listing view, it is displayed the whole records and I have implemented filters to filter the records in ascending or descending order. There are edit and delete buttons on every record. The delete is removing the record from the MongoDB database. ListingView
The edit is sending you to the update view, from where you can edit the record. UpdateView
The last view is the Create, it is created user-friendly, on every field I have put a placeholder to show how it is fine to create a record. CreateView

Database Design

The database design is simple because it is on one collection. All of the records have Object ID which was generated by MongoDB when I upload the database. The database is downloaded from Kaggle. It was with 10.0 Usability. The format of the database is simple. Database

Security and Scalability

In terms of security, I am sure that it is not secured. In the create view, there are required fields if you are not filling them you cannot create a new record. The scalability of the database is a non-relational database.
Security

Conclusion and reflection

Overall, i am joyful that i have the opportunity to create basic CRUD application with MVC architecture. The work with MVC was very advantageous for further projects, everything was in good order and easy for creating it. On the other hand, the learning of Node JS and Express was very helpful to develop the basic knowledge for one full-stack developer. Despite the fact that my application is a basic one, i am happy with the outcomes. There are a lot of aspects to improve on the application, such as creating another collection for dealers and creating user login, where the car dealer can login and create records for their cars as a result to read the record and compare the price. It is not common knowledge but most of the car dealers are using one small book with check list to determine the price. When you try to implement everything on one run, always everything is a failure. When you are creating something similar to that application i would suggest to do it step by step.