### 

### **Car Sell and Services API with Flask and MongoDB**

### Introduction

Car sell and services is a web based application in which services and selling of a car can be made where old or new car will be submitted to sell faster on this platform. New user can register themselves on this application and can submit car information to sell quickly . Information like car model, year, condition, expected price and image of a car can be submitted on the website which directly displays on the used car page where user will come and purchase it directly.Also loginned user can sell car as well as edit and delete their cars but non loginned user can only view or buy the cars. This application has been implemented using HTML, CSS and JS as a frontend stack and MongoDB and Python flask for backend stack.

# Case study:

The project implements a Car Sell and Services web-based application through Flask interface while integrating the Python-based web framework. Users access an interactive interface through this application for effective car selling and purchasing through CRUD operations. After completing user registration they can add model and year information with expected price, car condition, and picture which the application displays for shopping users. The system uses MongoDB as a database to store critical data while providing scalable and efficient storage and manipulation functionalities.

The Car Sell and Services web-based application implements Flask as its intuitive Python web framework to design and deploy the application. Through its user friendly interface the application program enables users to efficiently handle car listings through supported CRUD which stands for Create, Read, Update, Delete functionality. The system enables users to complete operations by signing up for an account then logging in to submit cars or edit current listings or remove them from the database. The application utilizes MongoDB as its data management solution since it provides scalable and flexible storage for structured user and vehicle data.

#### **1. CRUD Operations for Task Management**

* **Create**: :
  + - User can be added in user database using /register endpoint with post method
    - Car information while selling like model, year, condition, price and image using /sellcar route with POST method
* **Read**:
  + - User can be read from user database where they stored at the time of register
    - Also cars which has been stored at the time of create can be read from same database using /fetchcars route with get method
* **Update**:
  + - By using car database we can manipulate all the data and update all the car information also using PUT method to update or edit the car information using /update car route.
* **Delete**:
  + - Car information can be delete on used cars page by the loginned user by using /deletecar route with GET method.

#### **2. Database Integration**

* MongoDB as a part of this project has been initialized to store all the database and tables of it.
* For integration of MongoDB we have used MongoClient library where we configure mongo URL to store all the collections.

#### 

#### **4. Error Handling**

* For error handling we have used try catch block in most of the route which helps in throwing the error to the user if error will be found.
* Incase user is already present user database but new user with email has been found then it will throw the error and send to the user.

#### **6. User Interface (UI)**

* For rendering the pages like index.html page for homepage, login page we have used render\_template feature of flask to render directly to the user when they directly to specific route..
* Also, whenever query or id found in the route they it will treat dynamically and response according to it.

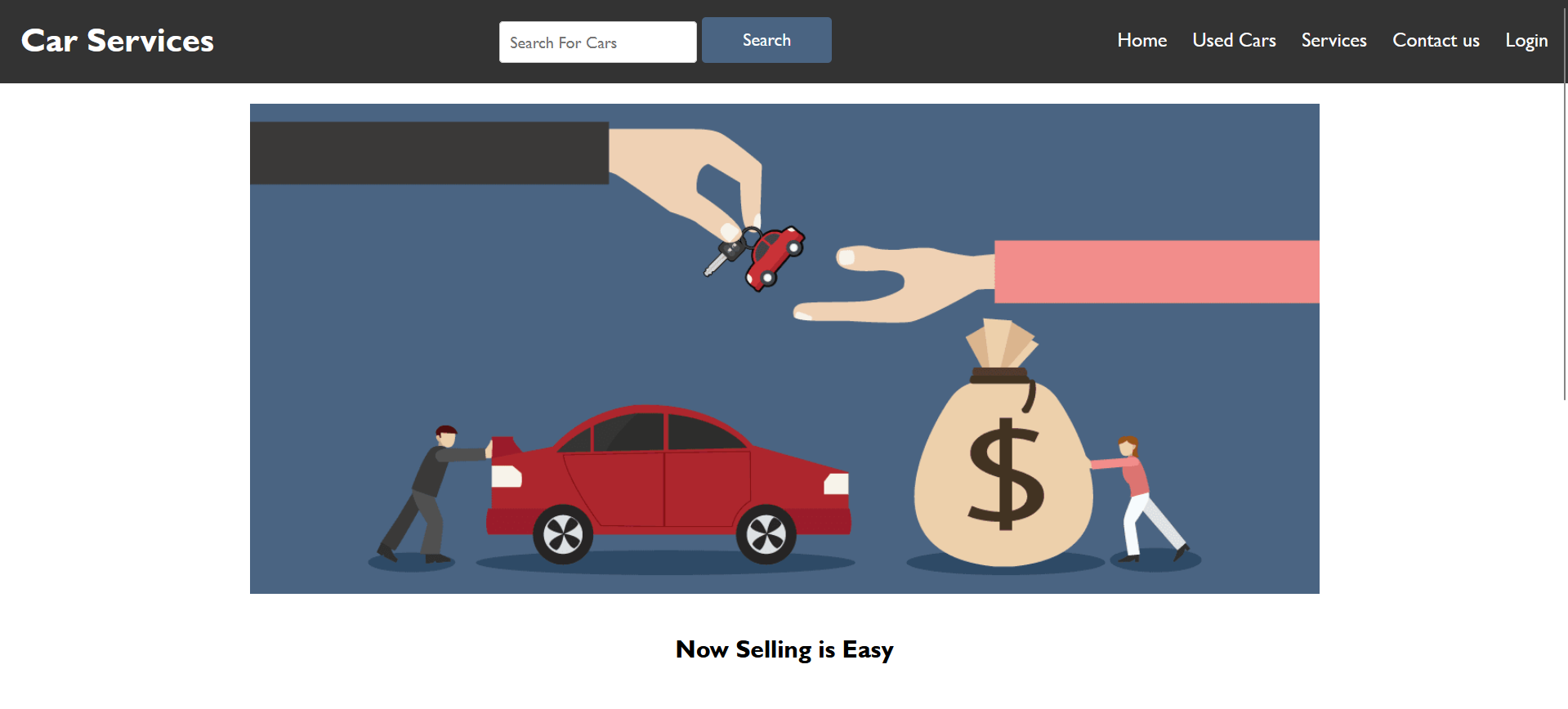
#### **Key Features Implemented:**

1. **Car Creation:**
   * Users can create car by providing:
     + Car Model
     + Car Price
     + Car Condition
     + Car Image
   * Validation has been done on frontend side so that it can be check at the time of form submission to rectify error fast and wait time will become less.
2. **Car Listing:**
   * Displays a list of all cars stores in mongoDB database
   * Task which displayed will container price, year and description and image of car.
3. **Car Update:**
   * Allows users to update a car model with specific document ID
   * Frontend display edit icon to edit the car.
4. **Car Deletion:**
   * Delete a car with specific ID.
   * Car deletion will automatically update a car list
5. **Interactive Interface:**
   * Routes like /sellcar, /updatecar, /deletecar, and /used car ensures a navigation to specific routes
6. **RESTful API Integration:**
   * Exposes RESTful endpoints for task management:
     + POST /sellcar for car creation
     + GET /usedcar to fetch all the cars contained in a database..
     + DELETE /deletecar used to delete a car
     + PUT /updatecar used to update a car .

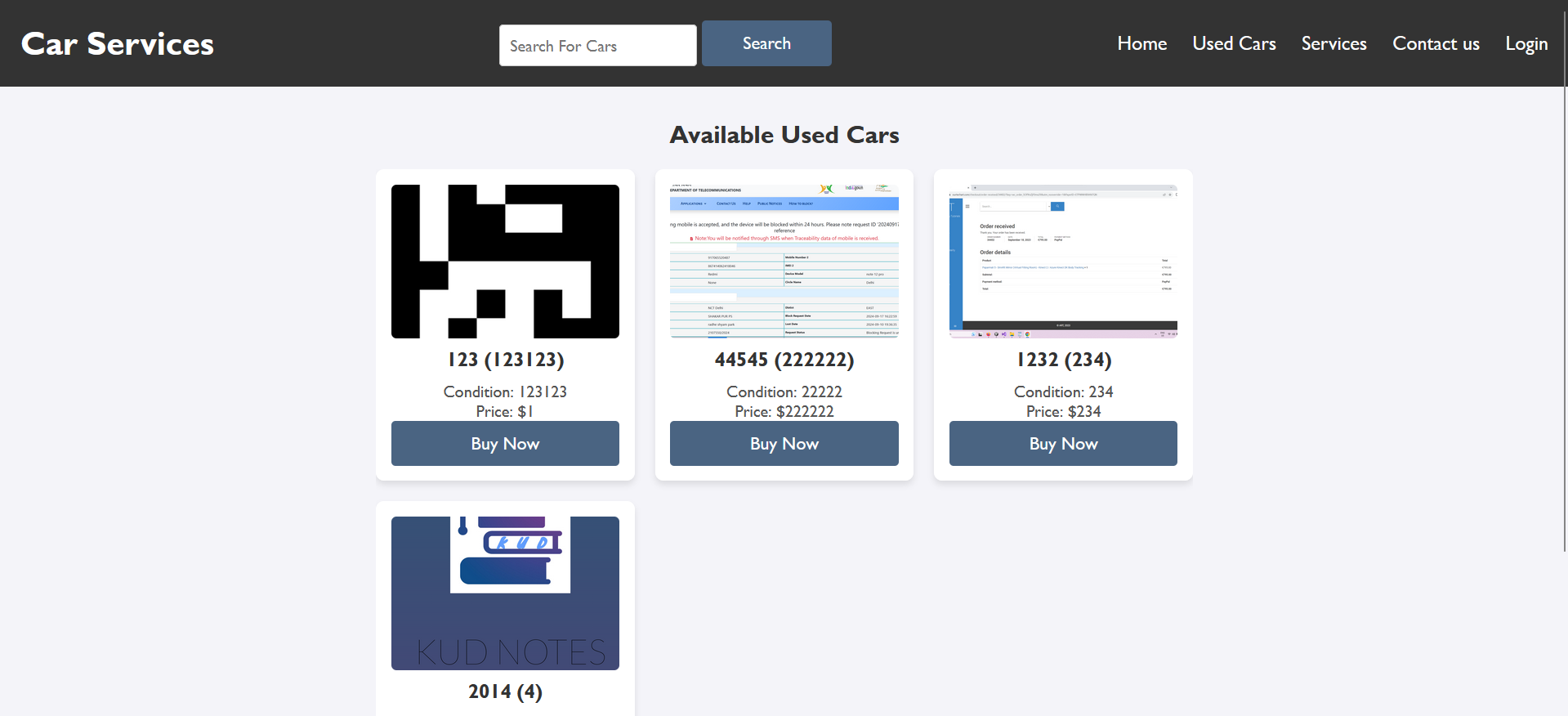
#### **2. Functional Requirements**

**2.1 Application Views**

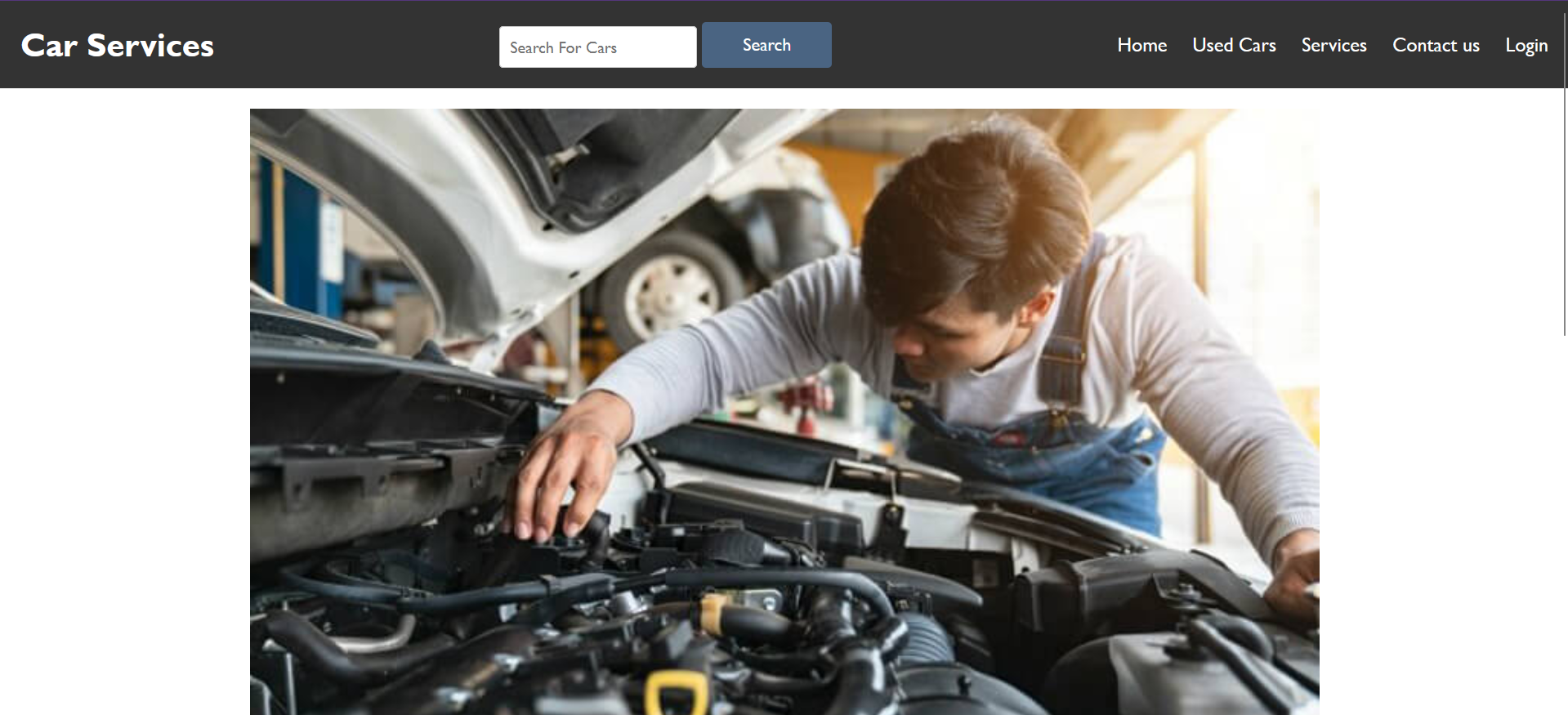
1. **Home Page (/):**
   * To display a homepage where use can navigate pages and also information of car page



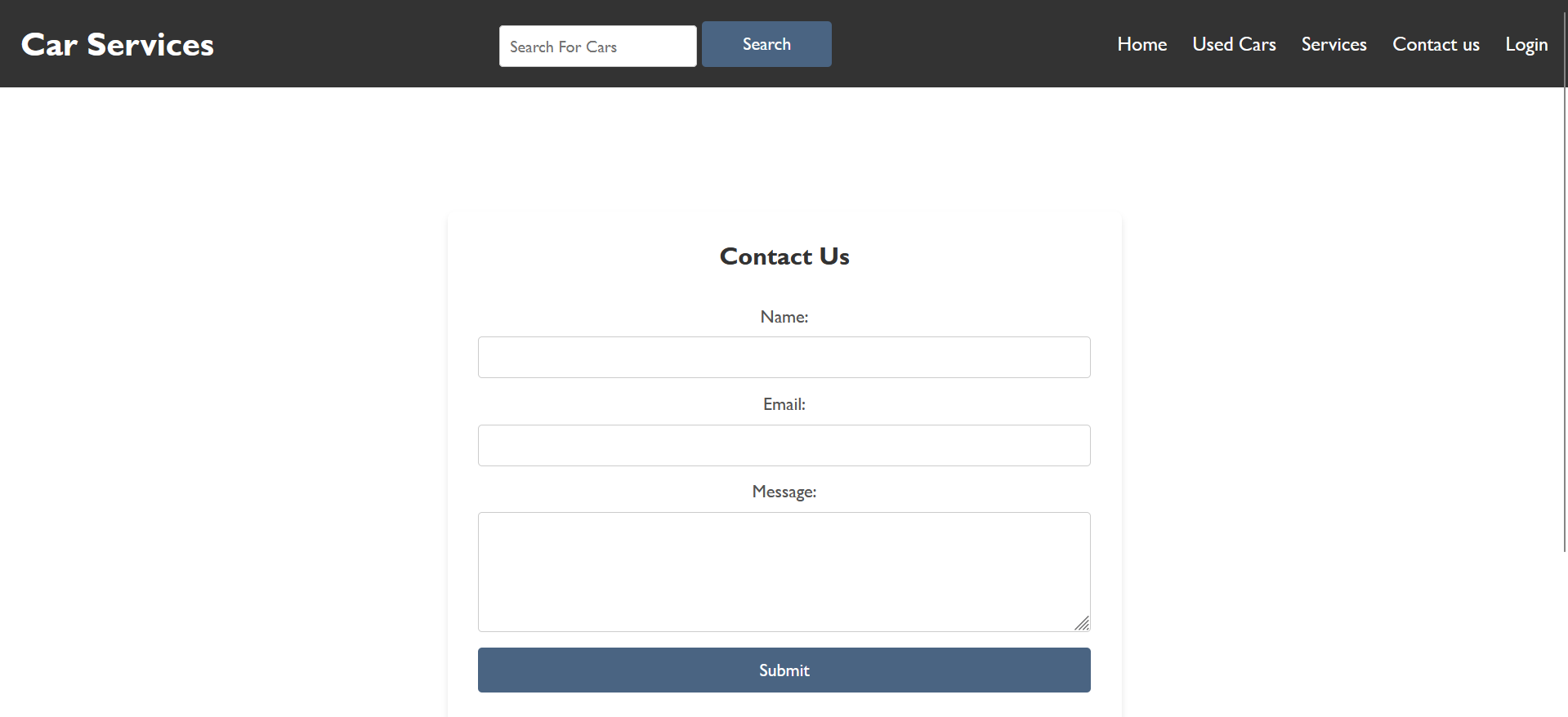
1. **To display a uploaded car model (/usedcar):**
   * Displays a cars inserted by a user



1. **Car View (/usedcarforusers):**
   * To display a car inserted by user by without editing options as it can be edited without logined.
2. **Service Page View (/services):**
   * To display a services provided a car company



1. **Contact us Page (/contactus):**
   * User can contact company if there is any pre booking or for any feedback



**3.1 Performance**

The application should be able to accommodate concurrent users without experiencing any delays. Under normal usage our expectation is for API response to be put back in to the callback in < 500ms.

**3.2 Usability**

In the system, the route naming should be clear and intuitive. Error message must be user friendly and it should provide clear guidence when error occurs.

**3.3 Scalability**

MongoDB is first used for simplicity but the system is designed to be migrated to more scalable database such as PostgreSQL data base as the application upgrades.

**3.4 Security**

All user inputs should be validated properly. It also means that any potential security threats common as SQL injection, XSS and CSRF should be protected against to ensure the safe data handling.

**3.5 Portability**

Application must be platform independent and should easily deployable on any system which supports Python and Flask (on cloud as well as in local environment).

#### **4. Data Requirements**

**4.1 Database Schema**Table: cars

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraint** |
| id | INTEGER | Primary Key |
| car\_name | TEXT | Not Null |
| car\_price | INTEGER | Not Null |
| car\_model | INTEGER | Not Null |
| car\_image | Text | Not Null |
| car\_condtion | Text | Not Null |

#### **5. Constraints**

1. Mongodb is used for database and table creation so that can be scale up easily
2. Flask application is integrated in python servers which serves locally and not deployed yet
3. No domain has been assigned to serve a frontend

#### **6. Deliverables**

* Fully functional Car sell Application with:
  + Web interface for car operations.

# **5.** **Implementation:**

### **5.1** **Development Environment:**

Developing the Car Sell and Services web application, we used Git to manage the code’s versions and keep track of smooth progress. In each update, commits were pushed to GitHub with adding additional features such as car listings, car editings, and user authentication to maintain a clean development history. GitHub served as central code repository to collaborate, to take backups, and to showcase the project in a structured and professional way.

#### **Backend:**

● It was Python because it’s simple, readable and has a lot of support for web development via flask framework.

● Flask is usually considered to be a lightweight and flexible web framework for python that provides the required tools and libraries to create a web application.

#### **Frontend:**

● HTML: To create a basic structure of pages for basic designing of tempaltes HTML has been used

● CSS: To style webpages cascading style sheets has been used to create advance designs..

● JavaScript: To give functionality and connectivity with backend javascript has been used.

#### **Database:**

● All the car related data, user information and transaction records has been stored in the database using MongoDB. It is a NoSQL database and since the nature of the application is dynamic, storing the car images and description in an unstructured way suits it. It also makes it easy to implement the CRUD operations on MongoDB with the Flask backend for managing data of the car listings, user accounts, as well as the purchase history.

#### **Development Tools:**

● Visual Studio Code

● Git and GitHub

The Car Sell and Services web application was developed utilizing Git, which was used to control the code versions and track the progress smoothly. Each update was accompanied by commits to GitHub to keep a clear history of development, and adding features like car listing, editing, and user authentication. As the central code repository GitHub was it made it much easier to collaborate, store backups and showcase the project in a controlled and professional manner.