# Car Dealerships – Full stack Capstone Project

A Dissertation

Submitted by

**Abir Sait**

**(AA.SC.P2MCA2207108)**

in partial fulfilment of the requirements for the award of the degree of

**MASTER OF COMPUTER APPLICATIONS**

**August 2024**

# BONAFIDE CERTIFICATE

This is to certify that this dissertation titled " Car Dealerships – Full stack Capstone Project," submitted in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Applications**, by [Abir Yusuf Mohd Sait] (AA.SC.P2MCA2207108), is a bona fide record of the work carried out by him/her under my supervision during the academic year 2023-2024 and that it has not been submitted, to the best of my knowledge, in part or in full, for the award of any other degree or diploma.

Project Guide's name: Ms Susmita C

Coordinator's name: Ms Vidyalekshmi Vinod

Reviewer: Ms Krishnapriya P S

Date: 27/08/2024

# DECLARATION

I do hereby declare that this dissertation titled **"ABIR YUSUF MOHD SAIT"**, submitted in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications,** is a true record of work carried out by me and that all information contained herein, which do not arise directly from my work, have been properly acknowledged and cited, using acceptable international standards. Further, I declare that the contents of this thesis have not been submitted, in part or in full, for the award of any other degree or diploma.

[illegible]

Signature of the student

Date: 27/08/2024

**NAME OF THE STUDENT**

Abir Yusuf Mohd Sait

## Acknowledgements

I would like to express my sincere gratitude to Amrita Vishwa Vidyapeetham for granting me the opportunity to participate in the Full Stack Application Development Capstone Project course. I am thankful to our Program Head, Dr. Manjusha Nair, and Academic Head of  MCA, Ms. Jayashree Narayanan, for their support throughout the duration of this project.

I extend my heartfelt appreciation to my Project Coordinator, Ms. Vidyalekshmi Vinod, for her continuous support, insightful feedback, and dedicated mentorship, which played an important role in the successful completion of this project.

I am also deeply appreciative of the Coursera team for their significant contributions to this course. Special thanks to Rav Ahuja, the Project Lead, and the Primary Instructors Upkar Lidder, Lavanya Thiruvali Sunderarajan, and Yan Luo for their expert guidance and insightful instruction. My gratitude extends to Priya Kapoor, the Instructional Designer, as well as Lab Authors Lavanya Thiruvali Sunderarajan, Upkar Lidder, and Yan Luo for their valuable input.

I would also like to acknowledge Bryan Hayes, the Project Coordinator from Skill-Up Technologies, and Nancy Varkey, the Project Manager from Skill-Up Technologies, for their roles in managing the project. Thanks are also due to the Publishing Team members Grace Barker from IBM, and Rachael Jones, Santosh Kumar Sahu, and Madhumita Pati from Skill-Up Technologies. Additionally, I appreciate the efforts of the QA team, including Pradnya B and Kunal Merchant, as well as the Teaching Assistants and Forum Moderators Anamika Agarwal, Lavanya Rajalingam, K. Sundararajan, and Pallavi Rai for their support and assistance.

Thank you all for your dedication and for making this learning journey both rewarding and enriching.

## Abstract

The IBM Full Stack Software Developer Professional Certificate program is designed to prepare individuals for careers in software development with a comprehensive approach. Offered through Coursera, this series covers a wide range of topics in full stack development, including both front-end and back-end technologies, as well as cloud-native application development. Participants will gain hands-on experience with essential tools such as HTML, CSS, JavaScript, React, and Bootstrap for front-end development, while also learning to deploy and scale applications using advanced cloud-native methodologies like Containers, Kubernetes, Microservices, and Serverless architectures.

The curriculum is structured to ensure a balanced acquisition of theoretical knowledge and practical skills through various labs and projects. This hands-on approach allows participants to build a comprehensive GitHub portfolio, showcasing their ability to solve real-world problems effectively.

Upon successful completion of the program, participants earn a professional certificate from IBM, highlighting proficiency in full stack cloud-native application development. This credential is highly esteemed in the industry and positions certificate holders for roles such as Application Developer and Cloud Application Developer.

With a focus on current industry practices and tools, the program aims to equip graduates with the necessary skills to thrive in the fast-paced software development field. By emphasizing practical application and industry relevance, participants are prepared to contribute meaningfully to projects and navigate the complexities of cloud application development with confidence and proficiency.

# Contents

# Chapter 1
# Introduction

## 1.1 Course Overview

| **Sl. No.** | **Courses** | **Total Duration (in hours)** | **Project-based Duration (in hours)** | **Completed/Not Completed** | **Outcomes** |
|---|---|---|---|---|---|
| **1** | *Introduction to Cloud Computing* | *12 hours* | *0.5 hours* | *Completed* | *Understand cloud computing fundamentals, service models (IaaS, PaaS, SaaS), deployment models (public, private, hybrid), and emerging trends* |
| **2** | *Introduction to Web Development with HTML, CSS, JavaScript* | *12 hours* | *1.5 hours* | *Completed* | *Gain foundational web development skills using HTML for structure, CSS for styling, and JavaScript for interactivity.* |
| **3** | *Getting Started with Git and GitHub* | *10 hours* | *1.5 hours* | *Completed* | *Master version control with Git, managing* |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | *repositories, branches, and pull requests on GitHub for collaborative coding.* |
| **4** | *Developing Front-End Apps with React* | *13 hours* | *3 hours* | *Completed* | *Develop user interfaces and dynamic applications using React, focusing on components, state management, and hooks.* |
| **5** | *Developing Back-End Apps with Node.js and Express* | *12 hours* | *4 hours* | *Completed* | *Create server-side applications with Node.js and Express.js, including routing, middleware, and RESTful APIs.* |
| **6** | *Python for Data Science, AI & Development* | *25 hours* | *2.8 hours* | *Completed* | *Learn Python for data science, utilizing libraries like Pandas, NumPy, and Matplotlib for analysis and visualization.* |
| **7** | *Developing AI Applications with Python and Flask* | *11 hours* | *2.25 hours* | *Completed* | *Build AI-powered applications using Python and Flask, integrating natural language processing and* |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  |  |  |  | *machine learning.* |
| **8** | *Django Application Development with SQL and Databases* | *14 hours* | *2 hours* | *Completed* | *Develop Django web applications, managing models, views, templates, and performing SQL database operations.* |
| **9** | *Introduction to Containers w/ Docker, Kubernetes & OpenShift* | *13 hours* | *4 hours* | *Completed* | *Understand containerization with Docker, manage applications with Kubernetes, and deploy on OpenShift.* |
| **10** | *Application Development using Microservices and Serverless* | *14 hours* | *3 hours* | *Completed* | *Develop applications using microservices architecture and serverless technologies, deploying functions on the cloud.* |
| **11** | *Full Stack Application Development Capstone Project* | *16 hours* | *18 hours* | *Completed* | *Apply full-stack skills in a real-world project, developing and deploying a comprehensive cloud-based application.* |

| | | | | | |
|---|---|---|---|---|---|
| **12** | *Generative AI: Elevate your Software Development Career* | *17 hours* | | | *Learn new concepts from industry experts, gain a foundational understanding of a subject or tool* |
| **13** | *Software Developer Career Guide and Interview Preparation* | *11 hours* | | | *Learn new concepts from industry experts, gain a foundational understanding of a subject or tool* |
| **14** | *Full Stack Software Developer Assessment* | *6 hours* | *-* | *Completed* | *Validate your full-stack development skills through a final assessment, demonstrating mastery in various technologies.* |
| | | ***186 hours*** | ***42.5 hours*** | | |

Table 1.1

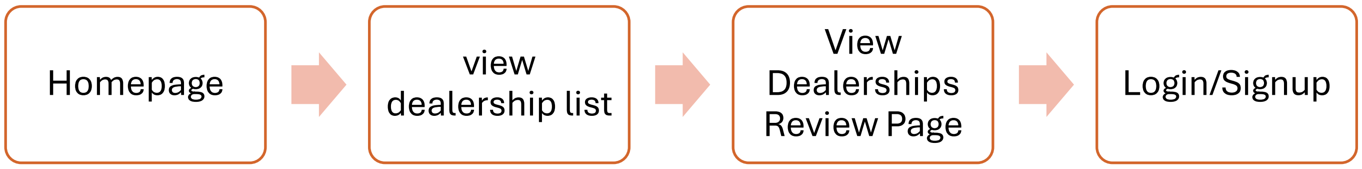
## 1.2 Project Overview

In the final practical course of IBM's Full Stack Software Developer Professional Certificate track, I completed the Coursera course "Full Stack Application Development Capstone Project". The objective was to leverage the skills acquired throughout the program to develop a comprehensive full-stack web application for a car dealership.

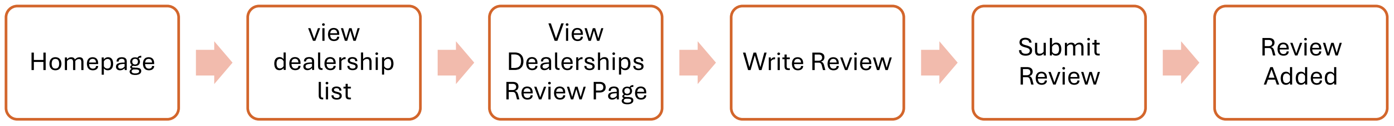
The project was structured into five modules, each with specific tasks and hands-on labs:

- Module 1: Focused on updating static web pages and running them using Django. I enhanced the appearance of stock images and text using Bootstrap, created a Django app, and configured the necessary settings.
- Module 2: Involved managing users through a dynamic front end with React and Django. The lab provided basic code for user login and logout, which required troubleshooting issues related to browser compatibility. I also developed user registration functionality.
- Module 3: Emphasized creating API endpoints using Express and MongoDB, implementing Django models for car makes and models, and developing Django proxy services for backend APIs. This included deploying a sentiment analysis service on IBM Cloud Code Engine.
- Module 4: Entailed adding dynamic front-end pages using React for various components, such as Dealers, Dealer Details, and Post Review, and ensuring proper integration with the backend services.
- Module 5: Covered CI/CD integration with GitHub Actions, which involved addressing numerous linting errors and configuring automated formatters. The second part of the module focused on containerizing the application with Docker and deploying it using Kubernetes.

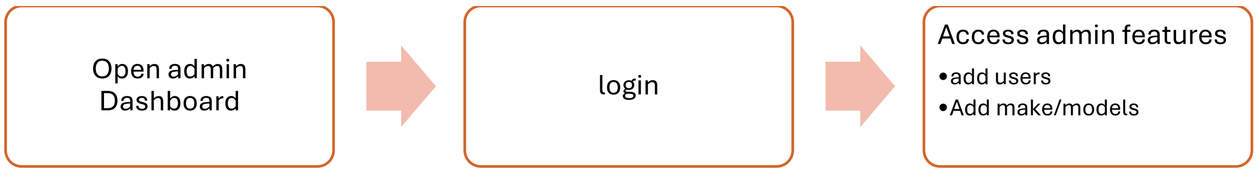
## Anonymous  user interaction



## Authorized/logged in  user interaction



### Admin user flow



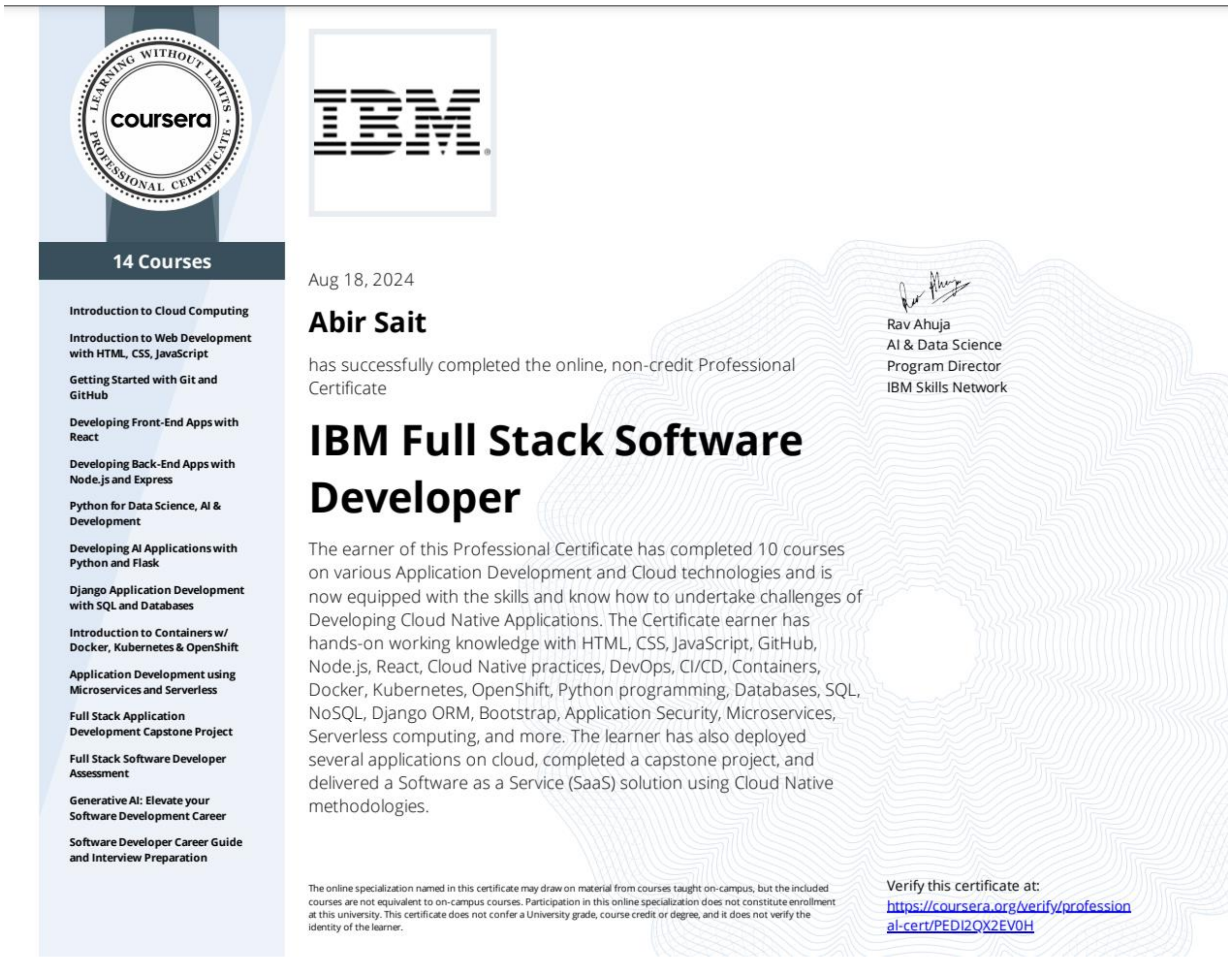
## 1.3 Project Outcomes

The capstone project resulted in a fully functional car dealership web application that effectively combines front-end and back-end technologies. Key outcomes included:

- Enhanced User Experience: Dynamic and interactive user interfaces were developed using React, providing a seamless experience for managing users and interacting with the dealership data.
- Robust Back-End: Implemented API endpoints and services with Express, MongoDB, and Django, ensuring efficient data handling and integration.
- Deployment and Scalability: Successfully containerized the application using Docker and deployed it on Kubernetes, demonstrating a strong understanding of modern DevOps practices and cloud technologies.
- CI/CD Integration: Integrated CI/CD pipelines with GitHub Actions, improving the development workflow and ensuring code quality through automated linting and formatting processes.

The project served as a comprehensive application of full-stack development skills, demonstrating proficiency in modern web technologies and deployment practices.

## 1.4 Course Certification



14 Courses

- Introduction to Cloud Computing
- Introduction to Web Development with HTML, CSS, JavaScript
- Getting Started with Git and GitHub
- Developing Front-End Apps with React
- Developing Back-End Apps with Node.js and Express
- Python for Data Science, AI & Development
- Developing AI Applications with Python and Flask
- Django Application Development with SQL and Databases
- Introduction to Containers w/ Docker, Kubernetes & OpenShift
- Application Development using Microservices and Serverless
- Full Stack Application Development Capstone Project
- Full Stack Software Developer Assessment
- Generative AI: Elevate your Software Development Career
- Software Developer Career Guide and Interview Preparation

Aug 18, 2024

**Abir Sait**

has successfully completed the online, non-credit Professional Certificate

**IBM Full Stack Software Developer**

The earner of this Professional Certificate has completed 10 courses on various Application Development and Cloud technologies and is now equipped with the skills and know how to undertake challenges of Developing Cloud Native Applications. The Certificate earner has hands-on working knowledge with HTML, CSS, JavaScript, GitHub, Node.js, React, Cloud Native practices, DevOps, CI/CD, Containers, Docker, Kubernetes, OpenShift, Python programming, Databases, SQL, NoSQL, Django ORM, Bootstrap, Application Security, Microservices, Serverless computing, and more. The learner has also deployed several applications on cloud, completed a capstone project, and delivered a Software as a Service (SaaS) solution using Cloud Native methodologies.

Rav Ahuja
AI & Data Science
Program Director
IBM Skills Network

The online specialization named in this certificate may draw on material from courses taught on-campus, but the included courses are not equivalent to on-campus courses. Participation in this online specialization does not constitute enrollment at this university. This certificate does not confer a University grade, course credit or degree, and it does not verify the identity of the learner.

Verify this certificate at:
https://coursera.org/verify/professional-cert/PEDI2QX2EV0H

**Link of certificate: here**

# Chapter 2
# Problem Definition

## 2.1 Context and Background

A national car dealership with local branches spread across the United States recently conducted a market survey. One of the suggestions that emerged from the survey was that customers would find it beneficial if they could access a central database of dealership reviews across the country. This feedback highlighted a need for a centralized platform where customers can view and submit reviews for various dealerships nationwide. By providing a comprehensive and accessible platform, the dealership aims to increase transparency and build customer trust.

The dealership has tasked its new hire with developing a website that allows new and existing customers to look up different branches by state and read customer reviews of the various branches. Customers should also be able to create an account and add their reviews for any of the branches. The management hopes this will bring transparency to the system and increase the trust customers have in the dealership.

To achieve this, the development team has created use cases for anonymous, authorized, and admin users. Anonymous users can view the Contact Us and About Us pages, browse the list of dealerships, filter dealerships by state, and view detailed reviews for each dealership. Authorized users, in addition to the aforementioned capabilities, can write reviews for any dealership. Admin users can log in to the admin site to manage car makes, models, and other attributes.

## 2.2 Problem Statement

The challenge is to develop a full-stack web application that enables customers to:

- View Dealership Information: Access a list of dealerships, filter by state, and view detailed reviews.
- Submit Reviews: Create accounts, submit reviews with detailed attributes, and view submitted reviews.
- Manage Dealership Data: Allow administrators to manage dealership attributes, including car makes and models.

The project involves several key steps to achieve these goals. First, static web pages need to be created and styled using Django and Bootstrap. User management will be implemented using Django's built-in user authentication system and a React front-end. Backend services will be developed using Node.js and Express to manage dealer and review data, with the services containerized using Docker. A sentiment analysis service will be deployed on IBM Cloud Code Engine to analyze review sentiments.

To ensure continuous integration and delivery, CI/CD pipelines will be set up using GitHub Actions. The application will be containerized with Docker and deployed using Kubernetes. Each component will be thoroughly tested to ensure functionality and reliability.

# Chapter 3
# Requirements

## 3.1 Hardware Requirements:

- Development Environment:
    - A modern computer with at least 8 GB of RAM and a multi-core processor.
    - Stable internet connection for accessing cloud services and repositories.
- Deployment Environment:
    - Servers or cloud infrastructure to deploy the application (e.g., IBM Cloud, Kubernetes cluster).
    - Storage for database and application containers.

## 3.2 Software Requirements:

The successful implementation and deployment of the car dealership web application requires specific software resources. The following table outlines the essential software requirements needed for the development, testing, and deployment phases of the project.

| **Software Requirements** | |
|---|---|
| Backend | Python, Django, Node.js, Express, MongoDB |
| Frontend | React.js, Bootstrap, HTML, CSS |
| Containerization | Docker, |
| Version Control | Git, GitHub |

| | |
|---|---|
| CI/CD Tools | GitHub Actions, Linter |
| Deployment Tools | Kubernetes, IBM Cloud Code Engine |
| Testing Tools | Automated test scripts |

# Chapter 4
# Proposed System

## 4.1 Architecture Overview

The architecture of the full-stack web application is designed to manage and review car dealerships, incorporating various technologies and services. The project is implemented through several stages, each addressing specific requirements and functionalities. Below is a detailed description of the project's architecture, including the development process and integration of various components.

- Repository Setup:
    - Fork the GitHub Repository: Start by forking the GitHub repository containing the project template here. This repository provides a predefined Django application that serves as the foundation for the project.
    - Clone the Repository: Clone the repository to your local development environment or Cloud IDE.
- Initial Setup:
    - Create Static Pages: Develop static pages to meet initial user requirements. These pages are created using Django and styled with Bootstrap.
    - Run the Application Locally: Test the application locally to ensure that static pages are correctly displayed.
- User Management:
    - Add User Management Features: Integrate Django's user authentication system to handle user registration, login, and management.
    - Implement React Frontend: Develop the frontend using React to manage user interactions and communicate with the Django backend.
- Backend Services:

    - Create Node.js Server: Develop a Node.js server to manage dealership data and reviews. Use MongoDB for data storage and Dockerize the Node.js application for containerization.
    - Deploy Sentiment Analyzer: Deploy a sentiment analysis service on IBM Cloud Code Engine to analyze the sentiment of reviews.
- Django Models and Views:
    - Create Django Models: Define Django models to manage car makes and models.
    - Develop Django Views: Implement views to handle car model and make management, and integrate them with the existing Django application.
- Integration of Services:
    - Django Proxy Services: Create Django proxy services to interface with the Node.js server for managing dealers and reviews.
    - Dynamic Pages with Django Templates: Develop dynamic pages using Django templates to display all dealerships, review details for selected dealerships, and allow users to submit reviews.
- CI/CD Implementation:
    - Set Up CI/CD Pipelines: Configure continuous integration and delivery pipelines using GitHub Actions for code linting, testing, and deployment.
    - Run and Test Application: Deploy the application on a Cloud IDE for testing and validation.
    - Deploy on Kubernetes: Finally, deploy the application using Kubernetes for production use.

## 4.2 Solution architecture

The user interacts with the "Dealerships Website", a Django website, through a web browser. The Django application provides the following microservices for the end user:

- get_cars/ - To get the list of cars from

- get_dealers/ - To get the list of dealers
- get_dealers/:state - To get dealers by state
- dealer/:id - To get dealer by id
- review/dealer/:id - To get reviews specific to a dealer
- add_review/ - To post review about a dealer

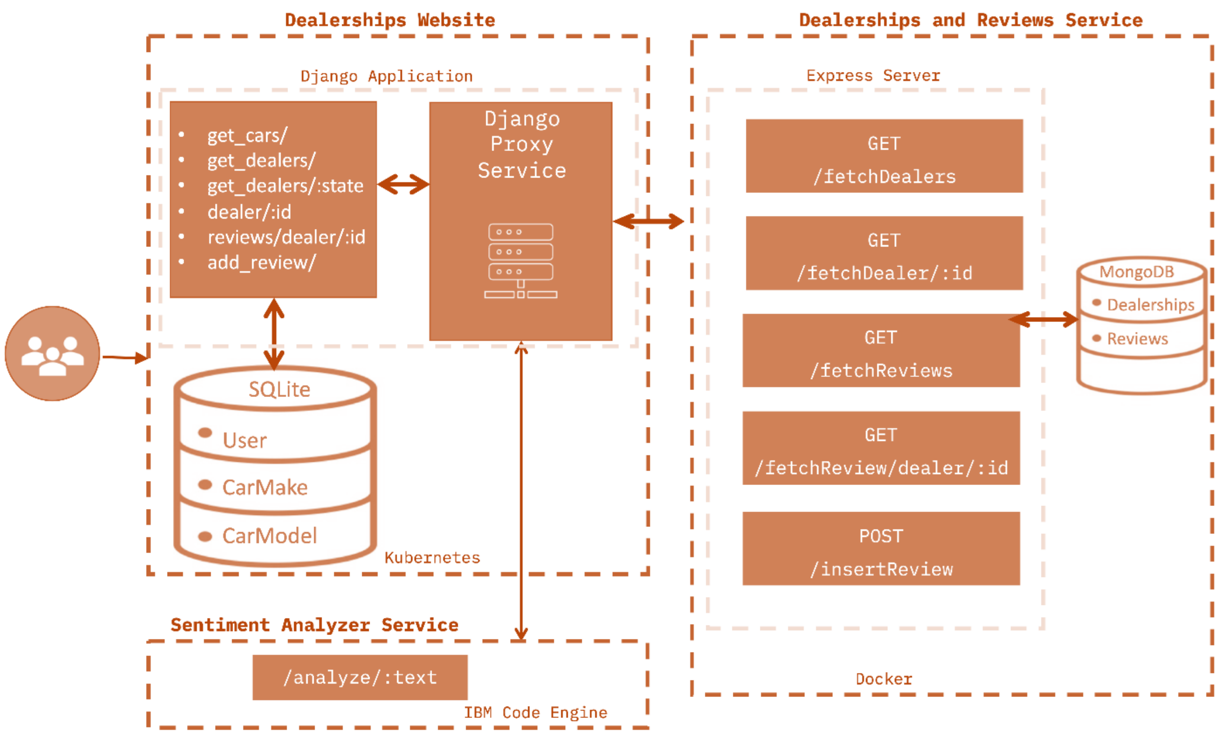
The application uses SQLite database to store the Car Make and the Car Model data. The "Dealerships and Reviews Service" is an Express Mongo service running in a Docker container. It provides the following services::

- /fetchDealers - To fetch the dealers
- /fetchDealer/:id - To fetch the dealer by id
- fetchReviews - To fetch all the reviews
- fetchReview/dealer/:id - To fetch reviews for a dealer by id
- /insertReview - To insert a review

"Dealerships Website" interacts with the "Dealership and Reviews Service" through the "Django Proxy Service" contained within the Django Application.
The "Sentiment Analyzer Service" is deployed on IBM Cloud Code Engine, it provides the following service:

- /analyze/:text - To analyze the sentiment of the text passed. It returns positive, negative or neutral.

The "Dealerships Website" consumes the "Sentiment Analyzer Service" to analyze the sentiments of the reviews through the Django Proxy contained within the Django application.



## 4.3 Detailed Module Implementation

### 4.3.1 Module 1: Static Pages

**Overview: Application - Static Pages**

The Django app will be mainly used for user management and authentication, managing car models and makes, and routing other MongoDB services for dealership and customer reviews.

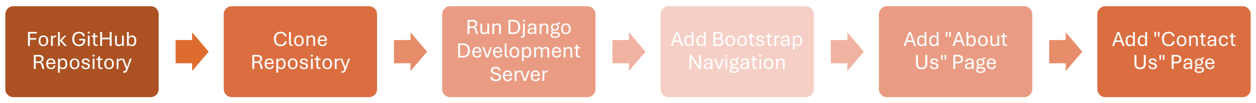
- Tasks for Module 1:
  - Fork the GitHub repo containing the project template.
  - Clone your forked repository.
  - Run the Django app on the development server.
  - Add navigation to the website using Bootstrap.
  - Add an "About Us" static page.
  - Add a "Contact Us" static page.

**Implementation Details:**

- Step 1: Fork the GitHub Repository
  - Description: Fork the repository to create a personal copy of the project. This allows for modifications and updates without affecting the original repository.
- Step 2: Clone Your Forked Repository
  - Description: Clone the forked repository to your local development environment to work on it.
  - Process:
    - Use the command git clone <your_forked_repo_url> to clone the repository.
- Step 3: Run the Django App on the Development Server
  - Description: Set up and run the Django development server to ensure the application is working correctly.
  - Process
    - Navigate to the project directory.
    - Install the required dependencies using pip install -r requirements.txt.
    - Run the Django development server using python manage.py runserver.
- Step 4: Add Navigation to the Website Using Bootstrap
  - Description: Enhance the user interface by adding a navigation bar to the website using Bootstrap.
  - Process:
    - Integrate Bootstrap into the project
    - Update the HTML templates to include a navigation bar
- Step 5: Add an "About Us" Static Page
  - Description: Create a static page that provides information about the dealership
  - Process: Create an HTML template for the "About Us" page.

- Step 6: Add a "Contact Us" Static Page
  - Description: Create a static page that provides contact information for the dealership.
  - Process: Create an HTML template for the "Contact Us" page.

**Module 1 Process:**



### 4.3.2 Module 2: User Management

**Overview: User Management**

Now that the initial Django application is built and deployed, the next step involves reviewing the app to identify users and manage their access based on roles (such as anonymous users or registered users). To accomplish this, you need to add authentication and authorization features to the app.

- Tasks for Module 2:
  - Create a superuser for your app.
  - Build the client side and configure it.
  - Check the client configuration.
  - Add a login view to handle login requests.
  - Add a logout view to handle logout requests.
  - Add a registration view to handle sign-up requests.

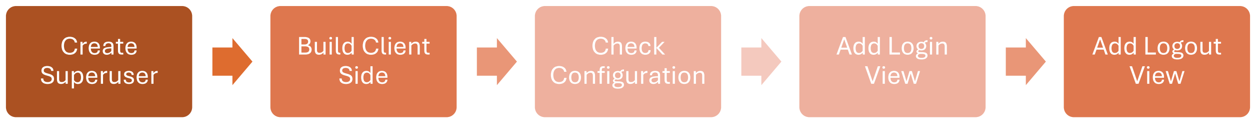
**Implementation Details:**

- Step 1: Create a Superuser for Your App
  - Description: Create an admin user who can manage the application and its users.
  - Process:
    - Run python manage.py createsuperuser and follow the prompts to create a superuser.

- Step 2: Build the Client Side and Configure It
  - Description: Develop the client-side application and ensure it communicates correctly with the Django backend.

  - Process:
    - Set up the frontend framework (e.g., React).
    - Configure the frontend to interact with the backend using APIs.
- Step 3: Check the Client Configuration
  - Description: Verify the client-side configuration to ensure it works as expected.
  - Process:
    - Test the client-side application locally.
    - Ensure it communicates properly with the Django backend.
- Step 4: Add a Login View to Handle Login Requests
  - Description: Implement a login view to manage user login requests.
  - Process:
    - Create a login form.
    - Handle form submissions and authenticate users using Django's authentication system.
- Step 5: Add a Logout View to Handle Logout Requests
  - Description: Implement a logout view to manage user logout requests.
  - Process:
    - Create a logout view that invalidates user sessions.
    - Redirect users to the login page after logout.
- Step 6: Add a Registration View to Handle Sign-Up Requests
  - Description: Implement a registration view to manage user sign-up requests.
  - Process:
    - Create a registration form.
    - Handle form submissions and create new user accounts.

Module 2 Process:



## 4.3.3 Module 3: Backend Services

**Overview**

In this module, you will implement some endpoints in the Express application that interact with MongoDB. You will then containerize the MongoDB and Express server using Docker and run it. Further, you will set up Car Make and Car Model with Django Models and populate the database. Finally, you will deploy a sentiment analyzer to the IBM Code Engine and create proxy services to access these external services.

- Tasks for Module 3:
    - Develop backend services for your Django application using JavaScript.
    - Set-up a new Express MongoDB server in a Docker container.
    - Deploy sentiment analyzer on IBM Code Engine.
    - Create data models in your Django application.
    - Create proxy services to call cloud functions in Django.
    - Create CarModel and CarMake Django models
    - Register CarModel and CarMake models with the admin site
    - Create new car models objects with associated car makes and dealerships

**Implementation Details**

Step 1 : Prepare MongoDB Data

- Use provided schema files for Reviews and Dealerships.
- Load JSON data into MongoDB
    - server/database/data/dealerships.json
    - server/database/data/reviews.json

- Mongoose Integration
    - Use mongoose to interact with MongoDB.
    - Define schemas in review.js and dealership.js.
- Create API Endpoints in app.js
    - fetchReviews: Fetch all reviews.
    - fetchReviews/dealer/:id: Fetch reviews for a particular dealer.

    - fetchDealers: Fetch all dealerships.
    - fetchDealers/:state: Fetch dealerships in a particular state.
    - fetchDealer/:id: Fetch dealer by ID.
    - insert_review: Insert reviews.
- Build and Run Docker App
    - Build Docker app.
        - docker build . -t nodeapp
    - Run Docker containers for MongoDB and Node app.
        - docker-compose up
- Create Additional Endpoints
    - Implement fetchDealers, fetchDealers/:state, and fetchDealer/:id.
- Test Endpoints
    - Test endpoints and take screenshots.

Step 2: Create Django Proxy Services of Backend APIs

- Set Up Django Environment
    - Open a terminal and set up Django environment.

```
cd /home/project/xrwvm-fullstack_developer_capstone/server
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
```

- Install required packages.

```
python3 -m pip install -U -r requirements.txt
```

- Create Proxy Functions
    - Open djangoapp/restapis.py and add a get_request method to interact with backend APIs.

Step 3 : Deploy Sentiment Analysis on IBM Code Engine

- Build and Push Docker Image
- Change to microservices directory.

```
cd xrwvm-fullstack_developer_capstone/server/djangoapp/microservices
```

- Build and push Docker image.

  ```
  docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/senti_analyzer
  docker push us.icr.io/${SN_ICR_NAMESPACE}/senti_analyzer
  ```
- Deploy Application on Code Engine
- Deploy the sentiment analyzer.

  ```
  ibmcloud ce application create --name sentianalyzer --image
  us.icr.io/${SN_ICR_NAMESPACE}/senti_analyzer --registry-secret icr-secret --
  port 5000
  ```
- Configure Django to Use Sentiment Analyzer
  - Update djangoapp/.env with the deployment URL.

    ```
    sentiment_analyzer_url=your_code_engine_deployment_url
    ```
- Update djangoapp/restapis.py with functions to consume the microservice.

Step 4: Create Django Views for Dealers and Reviews

- Get Dealerships
  - Update get_dealerships view in djangoapp/views.py.
  - Configure route in urls.py.
- Get Dealer Details
  - Create get_dealer_details method in views.py.
  - Map it in urls.py.
- Post Dealer Review
  - Add post_review method in restapis.py.
  - Create add_review method in views.py.
  - Configure route in urls.py.

Module 3

- backend services

Develop Backend Services

- Using JavaScript
- Setup Express Server
- use Mongoose for MongoDB

Set Up Express MongoDB Server

- Containerize with Docker
- Docker-compose for orchestration

Deploy Sentiment Analyzer

- IBM Code Engine

Create Django Data Models

- Car Make and Car Model

Create Proxy Services

- Integrate External Services

Integrate External Services

- fetchReviews
- fetchReviews/dealer/:id
- fetchDealers
- fetchDealers/:state
- fetchDealer/:id
- insert_review

### 4.3.4 Module 4: Dynamic pages

**Overview**:

In this module,  functionality to manage car dealerships is added. This includes creating, updating, and deleting dealerships, as well as listing them for users to view.

Tasks for Module 4:

- Build frontend pages to present backend services to end users.
- Create a component to list the dealerships.
- Develop a dealer details and reviews component.
- Create a review submission page.

**Implementation Details**

Step 1 : Add and Set Up React Component for Dealers Page

- Import and add the Dealers component in frontend/src/App.js:

```
import Dealers from './components/Dealers/Dealers';

<Route path="/dealers" element={<Dealers />} />
```

- Add routes for Dealers and Dealer in server/djangoproj/urls.py:

```
path('dealers/', TemplateView.as_view(template_name="index.html")),
```

- Build the front end:

```
cd /home/project/xrwvm-fullstack_developer_capstone/server/frontend

npm install

npm run build
```

- Check if the server is running without errors. Restart if necessary.
- Test the get_dealers view:
  - Launch the application with the development server on port 8000.

Step 2: Add React Component for Dealer Showing Reviews

- Import and add the route to the Dealer React component in frontend/src/App.js:

  import Dealer from "./components/Dealers/Dealer"

  <Route path="/dealer/:id" element={<Dealer />} />

- Build the front end again
- Add the path for showing the dealer page in server/djangoproj/urls.py:

  path('dealer/<int:dealer_id>',
  TemplateView.as_view(template_name="index.html")),

- Refresh the application, go to the View Reviews page, and click any dealer's name to see its reviews.

Step 3: Create a Dealer Details or Reviews Page

- Open and view frontend/src/components/Dealers/PostReview.jsx. Make any desired changes.

- Import PostReview component and add the route in frontend/src/App.js:

  import PostReview from "./components/Dealers/PostReview"

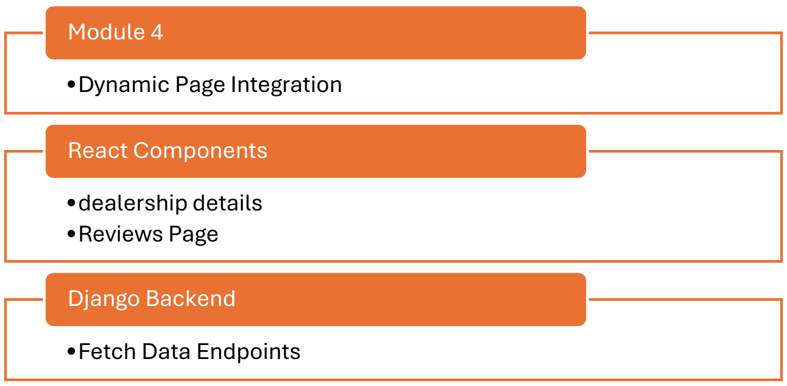
  <Route path="/postreview/:id" element={<PostReview />} />

- Build the front end again:
- Add the path to the post review page in server/djangoproj/urls.py:

  path('postreview/<int:dealer_id>',
  TemplateView.as_view(template_name="index.html")),

- Log in and test the Post Review link by adding a review to a dealership.
  - Enter the review details and take a screenshot before submitting it. Save as dealership_review_submission.png or dealership_review_submission.jpg.

- If successful, you will see the updated page with your review along with the sentiment.



### 4.3.5 Module 5: CI/CD, Containerize & Deploy to Kubernetes

Overview

In this module, Continuous Integration (CI) and Continuous Delivery (CD) are set up for the source code. Additionally, the application is containerized and deployed to Kubernetes for better scalability and management.

Tasks for Module 5:

- Enable GitHub Actions and run the Linting workflow
- Add the ability to your application to run in a container
- Add deployment artifacts for your application so it can be managed by Kubernetes

**Implementation details:**

GitHub Actions

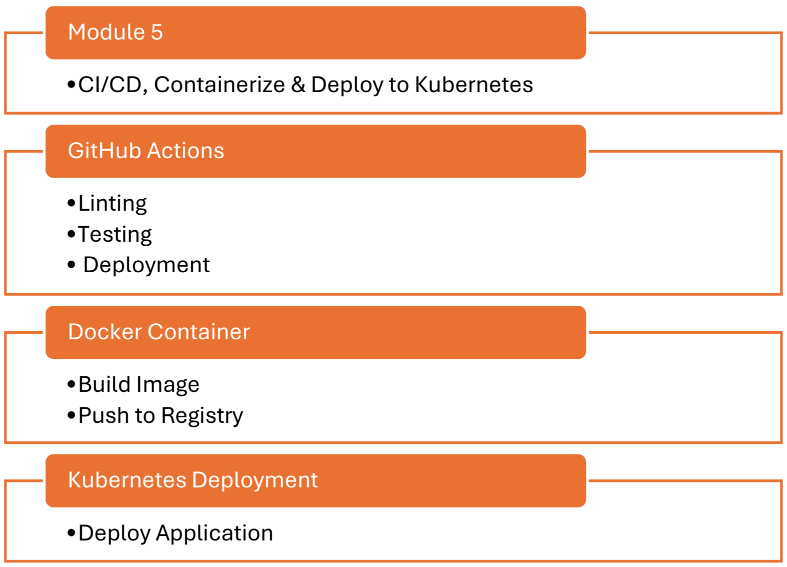
GitHub actions provide an event-driven way to automate tasks in your project. There are several kinds of events you can listen to. Here are a few examples:

- push: Runs tasks when someone pushes to a repository branch.
- pull_request: Runs tasks when someone creates a pull request (PR). You can also start tasks when certain activities happen, such as:
  - PR opened
  - PR closed

    - PR reopened
- create: Run tasks when someone creates a branch or a tag.
- delete: Run tasks when someone deletes a branch or a tag.
- manually: Jobs are kicked off manually.

<u>GitHub Action Components</u>

- Workflows: A collection of jobs you can add to your repository.
- Events: An activity that launches a workflow.
- Jobs: A sequence of one or more steps. Jobs are run in parallel by default.
- Steps: Individual tasks that can run in a job. A step may be an action or a command.
- Actions: The smallest block of a workflow.



<u>GitHub Workflow Template</u>

Filename: main.yml

```
name: 'Lint Code'

on:
  push:
    branches: [master, main]
  pull_request:
    branches: [master, main]

jobs:
  lint_python:
    name: Lint Python Files
    runs-on: ubuntu-latest

    steps:

    - name: Checkout Repository
      uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: 3.12

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install flake8

    - name: Print working directory
      run: pwd

    - name: Run Linter
      run: |
        pwd
        # This command finds all Python files recursively and runs flake8 on them
        find . -name "*.py" -exec flake8 {} +
        echo "Linted all the python files successfully"

  lint_js:
      name: Lint JavaScript Files
      runs-on: ubuntu-latest

      steps:
      - name: Checkout Repository
        uses: actions/checkout@v3

      - name: Install Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 14

      - name: Install JSHint
        run: npm install jshint --global

      - name: Run Linter
        run: |
          # This command finds all JavaScript files recursively and runs JSHint on them
          find ./server/database -name "*.js" -exec jshint {} +
          echo "Linted all the js files successfully"
```

Workflow Details

- Events: Triggers on push and pull requests to master and main branches.
- Jobs:
  - lint_python:
    - Sets up Python environment.
    - Installs dependencies.
    - Runs flake8 linter on Python files.
  - lint_js:
    - Sets up Node.js environment.
    - Installs jshint.
    - Runs jshint linter on JavaScript files.

Step 1: Enabling GitHub Actions

- Log into GitHub and open your repository.
- Go to the Actions tab and click "Set up a workflow yourself".
- Paste the above YAML code into main.yml and commit it.
- Check the Actions tab to see the lint workflow run automatically.
- Review workflow results: Green tick for success, red cross for errors.

Step 2: Common Linting Errors

- Python: Ensure all Python files adhere to PEP8 standards.
- JavaScript: Ensure all JS files follow best practices and coding standards.

Containerizing Your Application

To meet your company's hybrid cloud strategy and leverage Kubernetes, follow these steps to containerize your Django application.

Step 3: Add Dockerfile: Create a Dockerfile in the server directory with the following content:

Step 4: Build and Push Image to Container Registry

Step 5: Create an entrypoint.sh file in the server directory with the following content:

Step 6: Create a deployment.yaml file in the server directory with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: dealership
  name: dealership
spec:
  replicas: 1
  selector:
    matchLabels:
      run: dealership
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: dealership
    spec:
      containers:
      - image: us.icr.io/your-name-space/dealership:latest
        imagePullPolicy: Always
        name: dealership
        ports:
        - containerPort: 8000
          protocol: TCP
      restartPolicy: Always
```

Step 7: deploy the application

- Apply deployment:
    - kubectl apply -f deployment.yaml
- Port forward to access the application:
    - kubectl port-forward deployment.apps/dealership 8000:8000

Step 8: Launch Application

- Click the "Dealership Application" button or navigate to the "Skills Network Toolbox".
- Click OTHER, then "Launch Application".
- Enter port 8000 to see the running application.

# Chapter 5
# Results and Analysis

In this section, the results of the Full Stack Application Development Capstone Project is presented, highlighting the successful implementation of various tasks and features. Each task demonstrates a critical component of the application, from setting up the development environment to deploying a fully functional web application. The accompanying screenshots provide visual evidence of the tasks performed, showcasing the functionality and design of the application at different stages of development.

Below are the detailed descriptions of each task and the corresponding screenshots that illustrate the progress and achievements throughout the project.

**Task 1**

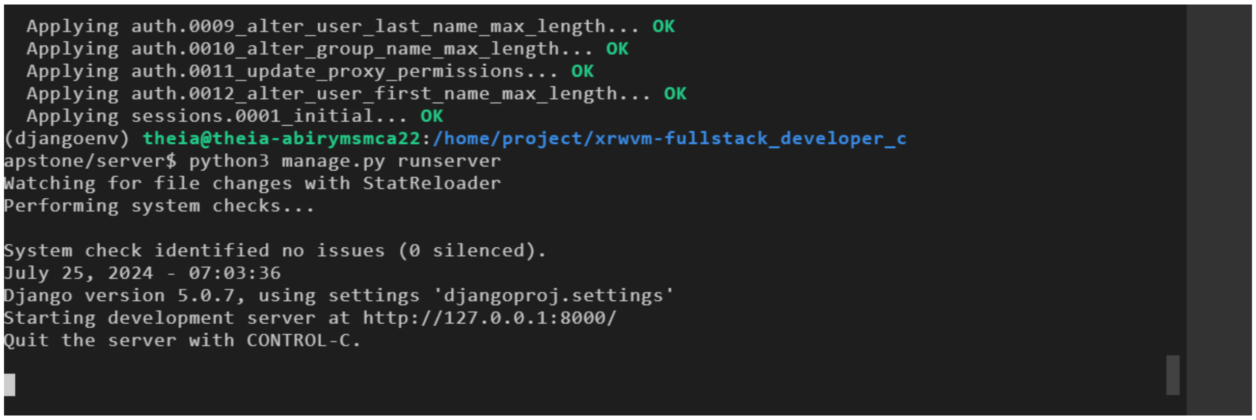
Submit your public GitHub Project repo URL with all code changes.

The source code for the entire project is available on GitHub. This repository contains all the code changes made during the development process, including updates to static web pages using Django, styling with Bootstrap, user management using React and Django, API endpoints with Express and MongoDB, and the implementation of CI/CD pipelines with GitHub Actions. The repository URL is: GitHub Repository.

**Task 2**

Submit the screenshot (django_server.png or django_server.jpg) showing the Django server running.

This screenshot demonstrates the successful deployment and execution of the Django server. It confirms that the backend environment is correctly set up and functioning, enabling the management of dynamic content and user interactions.



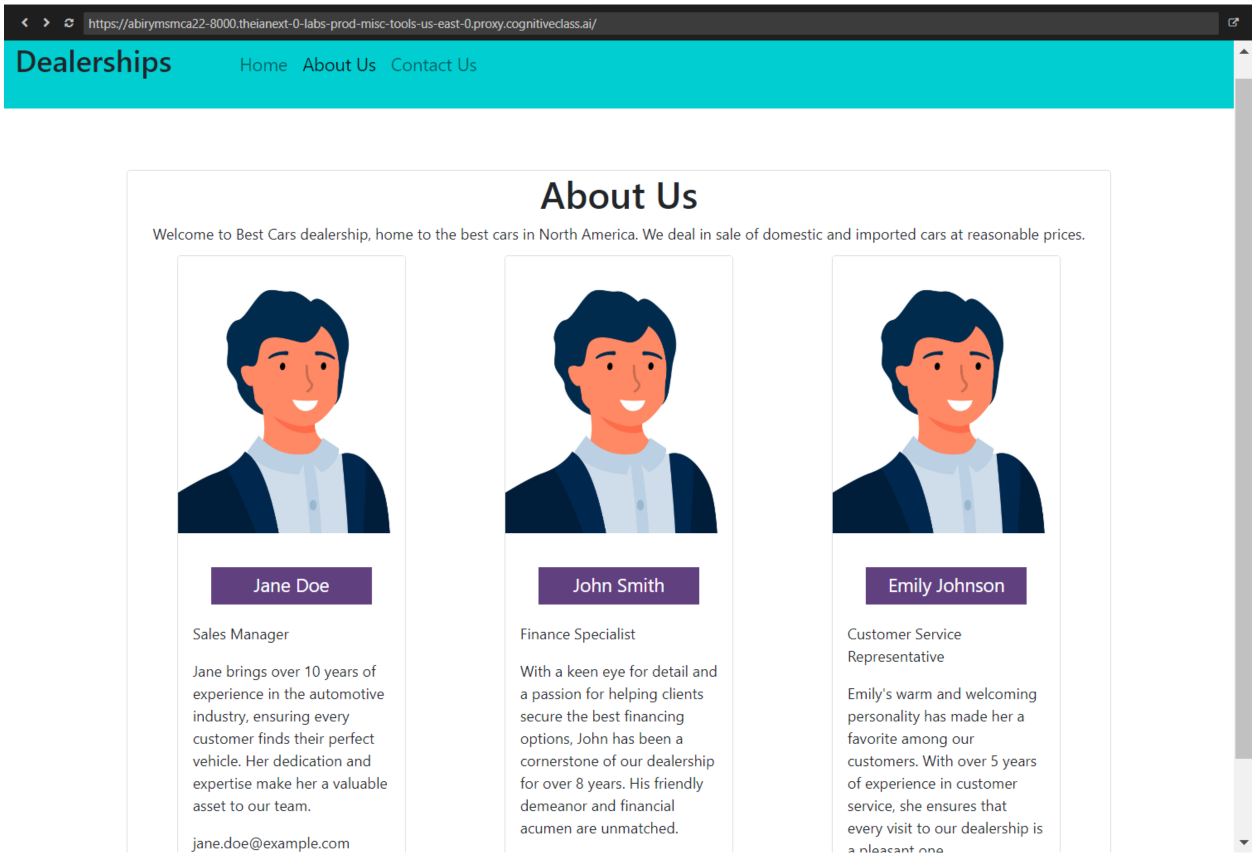
```
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(djangoenv) theia@theia-abirymsmca22:/home/project/xrwvm-fullstack_developer_c
apstone/server$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 25, 2024 - 07:03:36
Django version 5.0.7, using settings 'djangoproj.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

**Task 3**

Submit the screenshot (about_us.png or about_us.jpg) of the “About Us” page of the Django application.

The "About Us" page showcases the company's mission, values, and team information. This static component of the Django application is styled using Bootstrap, ensuring a consistent and professional appearance.



https://abirymsmca22-8000.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/

About Us

Welcome to Best Cars dealership, home to the best cars in North America. We deal in sale of domestic and imported cars at reasonable prices.

Jane Doe

Sales Manager

Jane brings over 10 years of experience in the automotive industry, ensuring every customer finds their perfect vehicle. Her dedication and expertise make her a valuable asset to our team.

jane.doe@example.com

John Smith

Finance Specialist

With a keen eye for detail and a passion for helping clients secure the best financing options, John has been a cornerstone of our dealership for over 8 years. His friendly demeanor and financial acumen are unmatched.

Emily Johnson

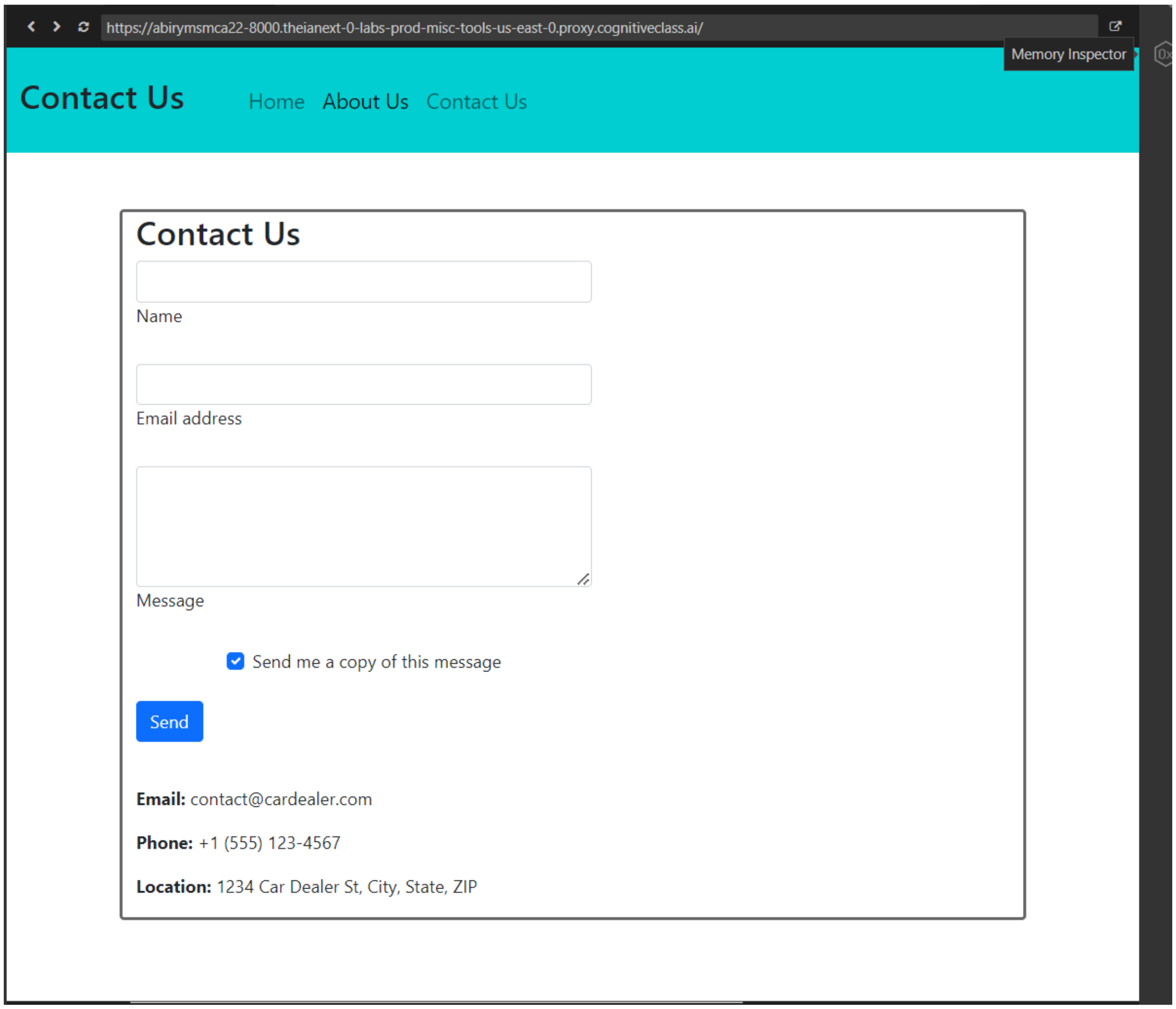
Customer Service Representative

Emily's warm and welcoming personality has made her a favorite among our customers. With over 5 years of experience in customer service, she ensures that every visit to our dealership is a pleasant one

**Task 4:**

Submit the screenshot (contact_us.png or .jpg) of the “Contact Us” page of the Django application.

The "Contact Us" page provides users with a form to reach out to the company. This feature is crucial for customer support and engagement, allowing users to submit inquiries directly through the application.


https://abirymsmca22-8000.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/
Memory Inspector
Contact Us
Home
About Us
Contact Us

Contact Us

Name

Email address

Message

✓ Send me a copy of this message

Send

**Email:** contact@cardealer.com

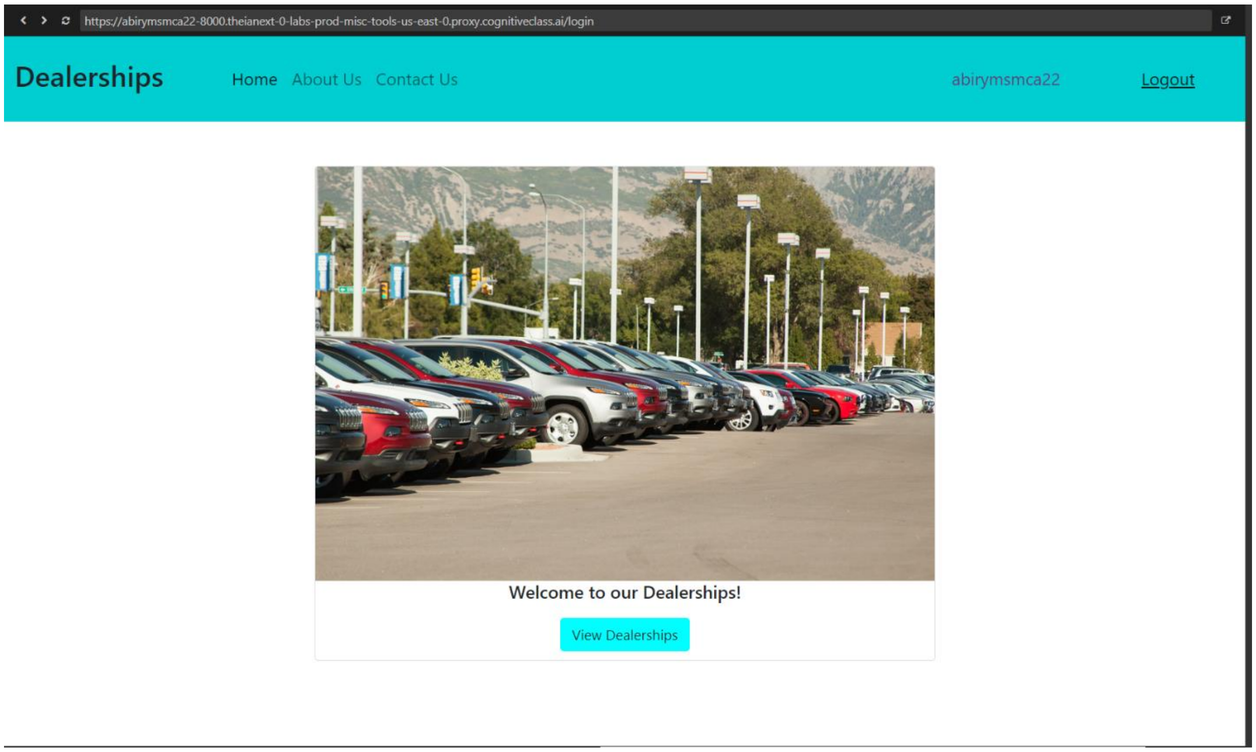
**Phone:** +1 (555) 123-4567

**Location:** 1234 Car Dealer St, City, State, ZIP

**Task 5:**

Submit the screenshot (login.png or login.jpg) of the "Login" page of the Django application.

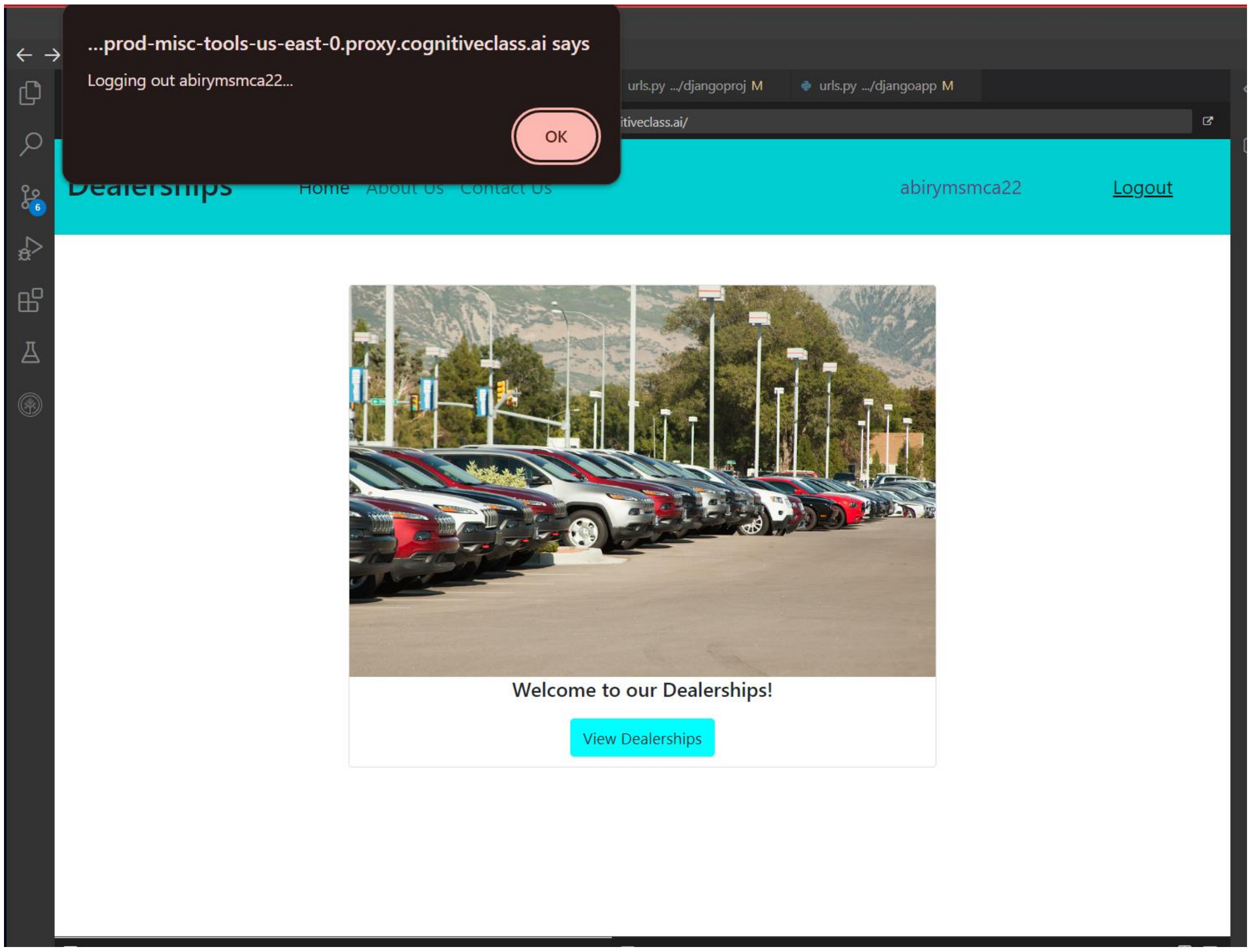
The login page allows users to authenticate themselves before accessing restricted features of the application. This task verifies the implementation of the user authentication mechanism, ensuring secure access to user-specific functionalities.


https://abirymsmca22-8000.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/login
Dealerships
Home
About Us
Contact Us
abirymsmca22
Logout
Welcome to our Dealerships!
View Dealerships

**Task 6:**

Submit the screenshot (logout.png or logout.jpg) of the "Logout alert" of the Django application.

The logout alert confirms that the user has successfully logged out of the application, ensuring proper session management and security.

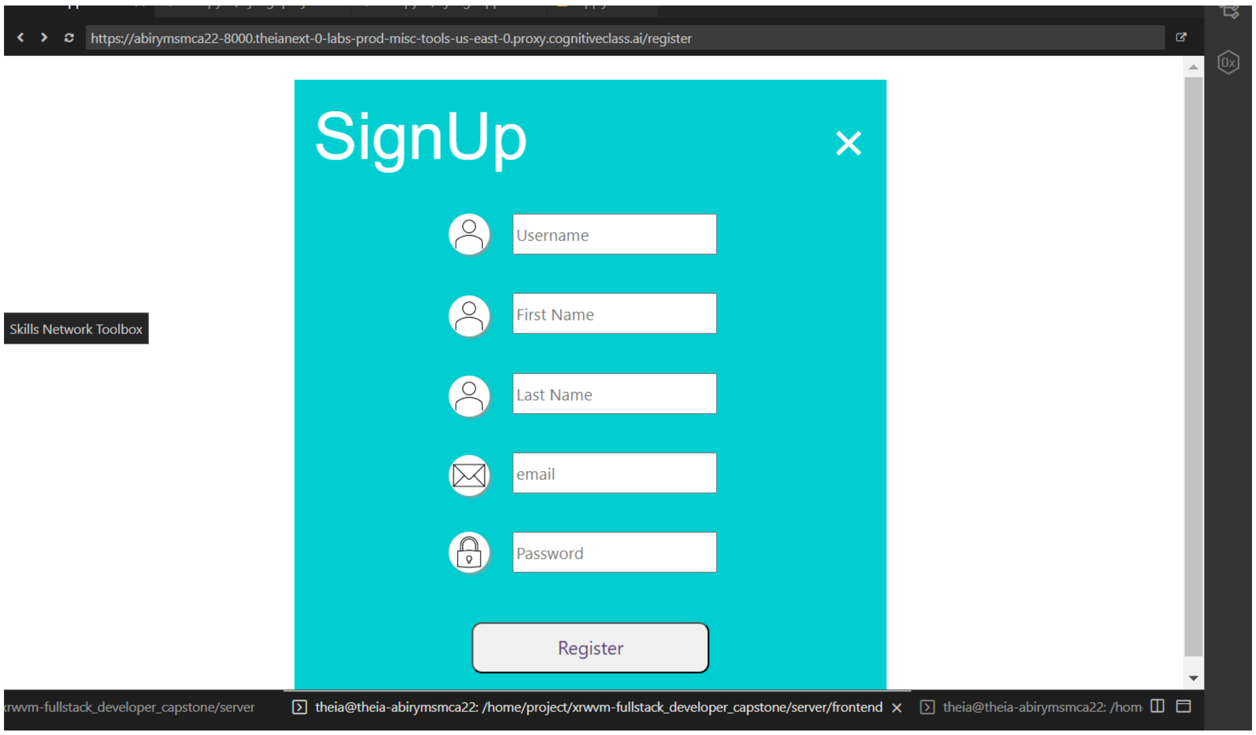

...prod-misc-tools-us-east-0.proxy.cognitiveclass.ai says

Logging out abirymsmca22...

**Task 7:**

Submit the screenshot (sign-up.png or sign-up.jpg) of the “Sign-up” page of the Django application.

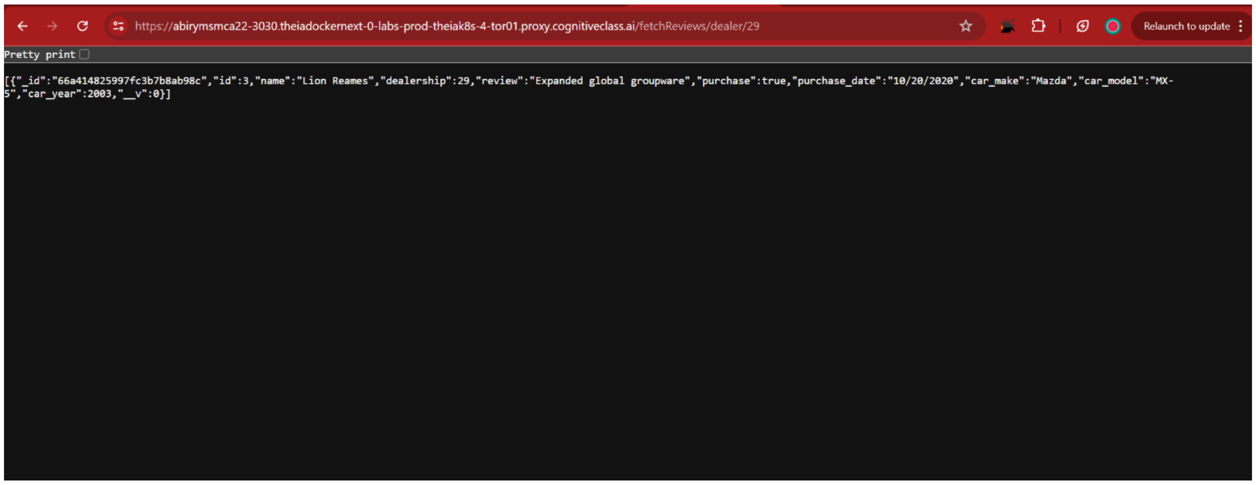
The sign-up page enables new users to create accounts. This task ensures that the user registration process is working as intended, allowing the application to expand its user base.


https://abirymsmca22-8000.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/register
SignUp
Username
First Name
Last Name
email
Password
Register
Skills Network Toolbox
xrwvm-fullstack_developer_capstone/server
theia@theia-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server/frontend
theia@theia-abirymsmca22: /hom

**Task 8:**

Submit the screenshot (dealer_review.png or dealer_review.jpg) showing dealer reviews through the Express-Mongo application endpoint.

This screenshot displays the dealer reviews fetched through the Express-Mongo backend. It confirms the successful integration of the backend service with the front-end React application, demonstrating the application’s capability to handle and display user-generated content.

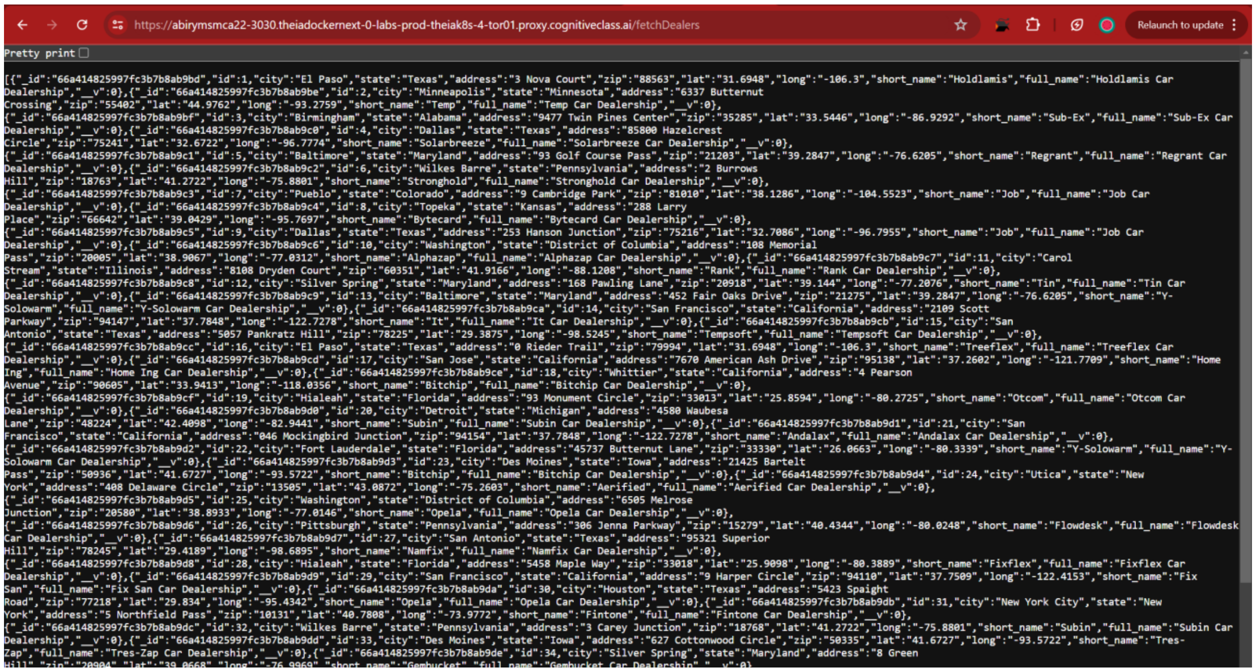

https://abirymsmca22-3030.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/fetchReviews/dealer/29
Relaunch to update
Pretty print

```
[{"_id":"66a414825997fc3b7b8ab98c","id":3,"name":"Lion Reames","dealership":29,"review":"Expanded global groupware","purchase":true,"purchase_date":"10/20/2020","car_make":"Mazda","car_model":"MX-
5","car_year":2003,"__v":0}]
```

**Task 9:**

Submit the screenshot (dealerships.png or dealerships.jpg) showing all dealers through the Express-Mongo application endpoint.

This screenshot shows a list of all dealers retrieved from the MongoDB database via the Express application. It demonstrates the correct data retrieval and display functionality, ensuring users can view comprehensive dealership information.

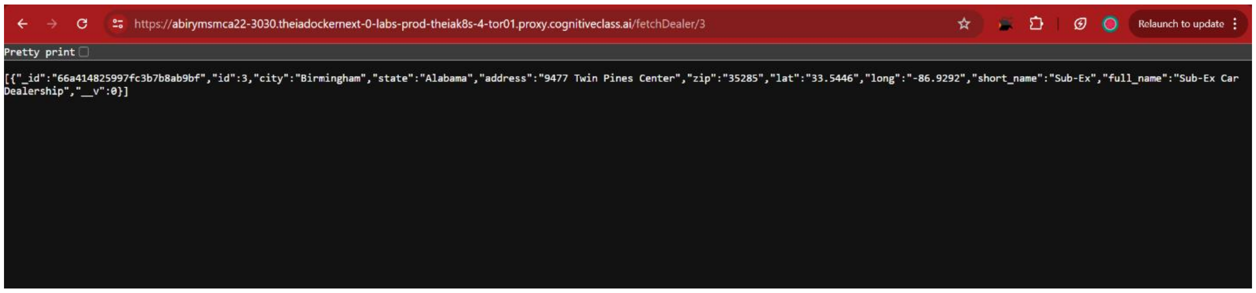

https://abirymsmca22-3030.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/fetchDealers
Relaunch to update
Pretty print

```
[{"_id":"66a414825997fc3b7b8ab9bd","id":1,"city":"El Paso","state":"Texas","address":"3 Nova Court","zip":"88563","lat":"31.6948","long":"-106.3","short_name":"Holdlamis","full_name":"Holdlamis Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9be","id":2,"city":"Minneapolis","state":"Minnesota","address":"6337 Butternut
Crossing","zip":"55402","lat":"44.9762","long":"-93.2759","short_name":"Temp","full_name":"Temp Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9bf","id":3,"city":"Birmingham","state":"Alabama","address":"9477 Twin Pines Center","zip":"35285","lat":"33.5446","long":"-86.9292","short_name":"Sub-Ex","full_name":"Sub-Ex Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c0","id":4,"city":"Dallas","state":"Texas","address":"85800 Hazelcrest
Circle","zip":"75241","lat":"32.6722","long":"-96.7774","short_name":"Solarbreeze","full_name":"Solarbreeze Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9c1","id":5,"city":"Baltimore","state":"Maryland","address":"93 Golf Course Pass","zip":"21203","lat":"39.2847","long":"-76.6205","short_name":"Regrant","full_name":"Regrant Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c2","id":6,"city":"Wilkes Barre","state":"Pennsylvania","address":"2 Burrows
Hill","zip":"18763","lat":"41.2722","long":"-75.8801","short_name":"Stronghold","full_name":"Stronghold Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9c3","id":7,"city":"Pueblo","state":"Colorado","address":"9 Cambridge Park","zip":"81010","lat":"38.1286","long":"-104.5523","short_name":"Job","full_name":"Job Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c4","id":8,"city":"Topeka","state":"Kansas","address":"288 Larry
Place","zip":"66642","lat":"39.0429","long":"-95.7697","short_name":"Bytecard","full_name":"Bytecard Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9c5","id":9,"city":"Dallas","state":"Texas","address":"253 Hanson Junction","zip":"75216","lat":"32.7086","long":"-96.7955","short_name":"Job","full_name":"Job Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c6","id":10,"city":"Washington","state":"District of Columbia","address":"108 Memorial
Pass","zip":"20005","lat":"38.9067","long":"-77.0312","short_name":"Alphazap","full_name":"Alphazap Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c7","id":11,"city":"Carol
Stream","state":"Illinois","address":"8108 Dryden Court","zip":"60351","lat":"41.9166","long":"-88.1208","short_name":"Rank","full_name":"Rank Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9c8","id":12,"city":"Silver Spring","state":"Maryland","address":"168 Pawling Lane","zip":"20918","lat":"39.144","long":"-77.2076","short_name":"Tin","full_name":"Tin Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9c9","id":13,"city":"Baltimore","state":"Maryland","address":"452 Fair Oaks Drive","zip":"21275","lat":"39.2847","long":"-76.6205","short_name":"Y-
Solowarm","full_name":"Y-Solowarm Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9ca","id":14,"city":"San Francisco","state":"California","address":"2109 Scott
Parkway","zip":"94147","lat":"37.7848","long":"-122.7278","short_name":"It","full_name":"It Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9cb","id":15,"city":"San
Antonio","state":"Texas","address":"5057 Pankratz Hill","zip":"78225","lat":"29.3875","long":"-98.5245","short_name":"Tempsoft","full_name":"Tempsoft Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9cc","id":16,"city":"El Paso","state":"Texas","address":"0 Rieder Trail","zip":"79994","lat":"31.6948","long":"-106.3","short_name":"Treeflex","full_name":"Treeflex Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9cd","id":17,"city":"San Jose","state":"California","address":"7670 American Ash Drive","zip":"95138","lat":"37.2602","long":"-121.7709","short_name":"Home
Ing","full_name":"Home Ing Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9ce","id":18,"city":"Whittier","state":"California","address":"4 Pearson
Avenue","zip":"90605","lat":"33.9413","long":"-118.0356","short_name":"Bitchip","full_name":"Bitchip Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9cf","id":19,"city":"Hialeah","state":"Florida","address":"93 Monument Circle","zip":"33013","lat":"25.8594","long":"-80.2725","short_name":"Otcom","full_name":"Otcom Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d0","id":20,"city":"Detroit","state":"Michigan","address":"4580 Waubesa
Lane","zip":"48224","lat":"42.4098","long":"-82.9441","short_name":"Subin","full_name":"Subin Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d1","id":21,"city":"San
Francisco","state":"California","address":"046 Mockingbird Junction","zip":"94154","lat":"37.7848","long":"-122.7278","short_name":"Andalax","full_name":"Andalax Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9d2","id":22,"city":"Fort Lauderdale","state":"Florida","address":"45737 Butternut Lane","zip":"33330","lat":"26.0663","long":"-80.3339","short_name":"Y-Solowarm","full_name":"Y-
Solowarm Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d3","id":23,"city":"Des Moines","state":"Iowa","address":"21425 Bartelt
Pass","zip":"50936","lat":"41.6727","long":"-93.5722","short_name":"Bitchip","full_name":"Bitchip Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d4","id":24,"city":"Utica","state":"New
York","address":"408 Delaware Circle","zip":"13505","lat":"43.0872","long":"-75.2603","short_name":"Aerified","full_name":"Aerified Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9d5","id":25,"city":"Washington","state":"District of Columbia","address":"6505 Melrose
Junction","zip":"20580","lat":"38.8933","long":"-77.0146","short_name":"Opela","full_name":"Opela Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9d6","id":26,"city":"Pittsburgh","state":"Pennsylvania","address":"306 Jenna Parkway","zip":"15279","lat":"40.4344","long":"-80.0248","short_name":"Flowdesk","full_name":"Flowdesk
Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d7","id":27,"city":"San Antonio","state":"Texas","address":"95321 Superior
Hill","zip":"78245","lat":"29.4189","long":"-98.6895","short_name":"Namfix","full_name":"Namfix Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9d8","id":28,"city":"Hialeah","state":"Florida","address":"5458 Maple Way","zip":"33018","lat":"25.9098","long":"-80.3889","short_name":"Fixflex","full_name":"Fixflex Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9d9","id":29,"city":"San Francisco","state":"California","address":"9 Harper Circle","zip":"94110","lat":"37.7509","long":"-122.4153","short_name":"Fix
San","full_name":"Fix San Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9da","id":30,"city":"Houston","state":"Texas","address":"5423 Spaight
Road","zip":"77218","lat":"29.834","long":"-95.4342","short_name":"Opela","full_name":"Opela Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9db","id":31,"city":"New York City","state":"New
York","address":"5 Northfield Pass","zip":"10131","lat":"40.7808","long":"-73.9772","short_name":"Fintone","full_name":"Fintone Car Dealership","__v":0},
{"_id":"66a414825997fc3b7b8ab9dc","id":32,"city":"Wilkes Barre","state":"Pennsylvania","address":"3 Carey Junction","zip":"18768","lat":"41.2722","long":"-75.8801","short_name":"Subin","full_name":"Subin Car
Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9dd","id":33,"city":"Des Moines","state":"Iowa","address":"627 Cottonwood Circle","zip":"50335","lat":"41.6727","long":"-93.5722","short_name":"Tres-
Zap","full_name":"Tres-Zap Car Dealership","__v":0},{"_id":"66a414825997fc3b7b8ab9de","id":34,"city":"Silver Spring","state":"Maryland","address":"8 Green
Hill","zip":"20904","lat":"39.0668","long":"-76.9969","short_name":"Gembucket","full_name":"Gembucket Car Dealership","__v":0},
```

**Task 10:**

Submit the screenshot (dealer_details.png or dealer_details.jpg) showing all dealer details through the Express-Mongo application endpoint.

This task shows detailed information about a specific dealer, including their reviews, ensuring the detailed data retrieval and display works correctly. This feature allows users to make informed decisions based on dealer information.

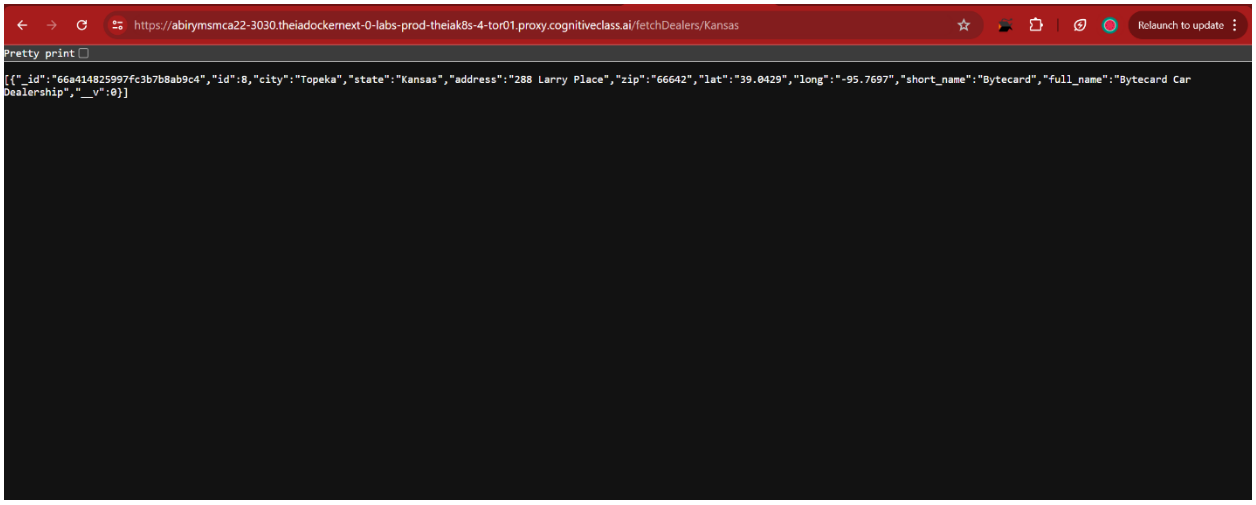

https://abirymsmca22-3030.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/fetchDealer/3
Relaunch to update
Pretty print

```
[{"_id":"66a414825997fc3b7b8ab9bf","id":3,"city":"Birmingham","state":"Alabama","address":"9477 Twin Pines Center","zip":"35285","lat":"33.5446","long":"-86.9292","short_name":"Sub-Ex","full_name":"Sub-Ex Car Dealership","__v":0}]
```

**Task 11:**

Submit the screenshot (kansasDealers.png or kansasDealers.jpg) showing all dealers in Kansas through the Express-Mongo application endpoint.

The screenshot shows the dealers filtered by state (Kansas) from the Express-Mongo endpoint, verifying the filtering functionality of the API. This feature enhances user experience by allowing state-specific dealer searches.

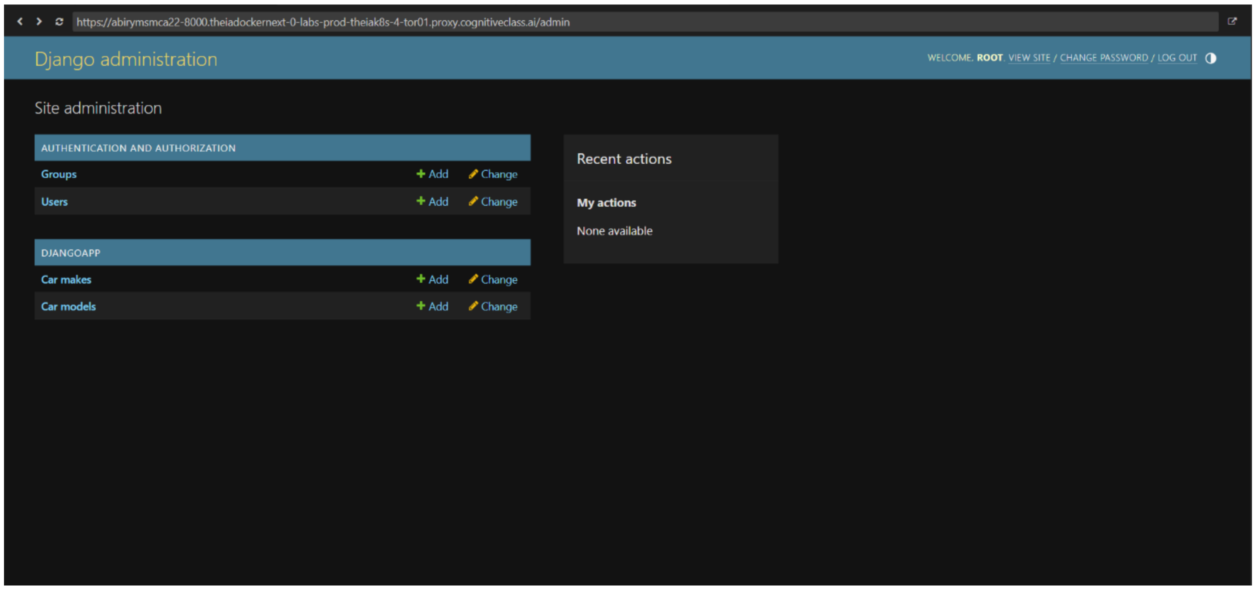

https://abirymsmca22-3030.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/fetchDealers/Kansas
Relaunch to update
Pretty print

```
[{"_id":"66a414825997fc3b7b8ab9c4","id":8,"city":"Topeka","state":"Kansas","address":"288 Larry Place","zip":"66642","lat":"39.0429","long":"-95.7697","short_name":"Bytecard","full_name":"Bytecard Car Dealership","__v":0}]
```

**Task 12:**

Submit the screenshot (admin_login.png or admin_login.jpg) showing the root user login on the admin page.

This task verifies the administrator's ability to log in to the Django admin interface. The admin interface is crucial for managing application data, including user accounts and dealership information.



https://abirymsmca22-8000.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/admin

Django administration

WELCOME, **ROOT**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

| AUTHENTICATION AND AUTHORIZATION | | |
|---|---|---|
| Groups | + Add | Change |
| Users | + Add | Change |

| DJANGOAPP | | |
|---|---|---|
| Car makes | + Add | Change |
| Car models | + Add | Change |

Recent actions

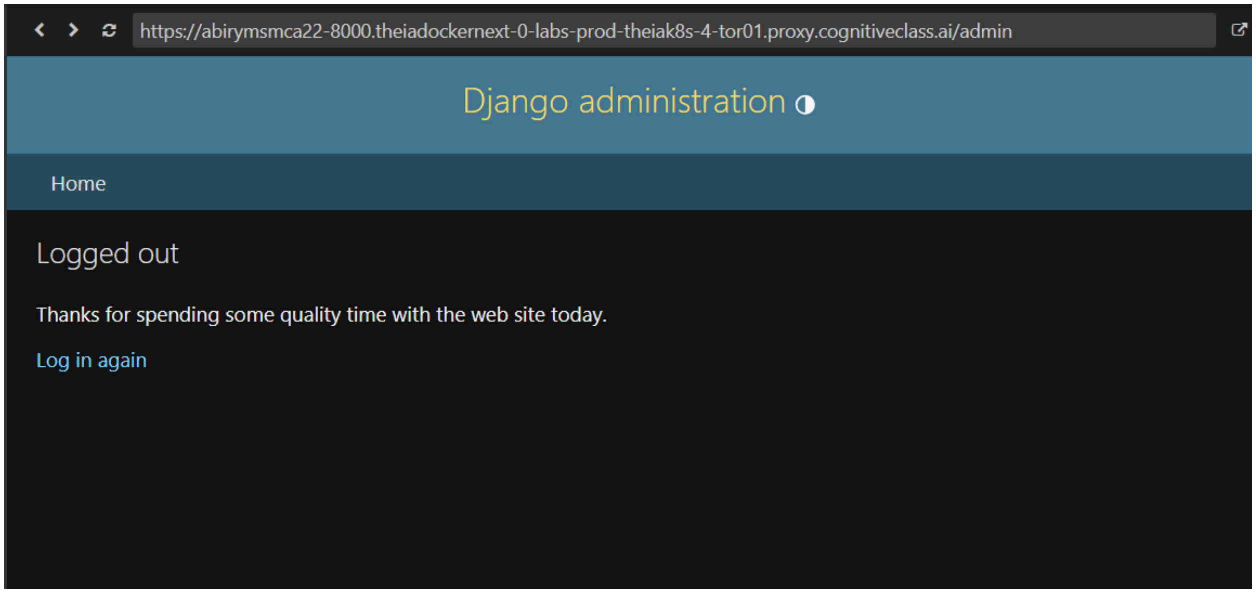
**My actions**

None available

**Task 13:**

Submit the screenshot (admin_logout.png or admin_logout.jpg) showing the root user logged out from the admin page.

This screenshot demonstrates the successful logout process from the Django admin interface, ensuring secure session management for administrators.


https://abirymsmca22-8000.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/admin

Django administration

Home

Logged out

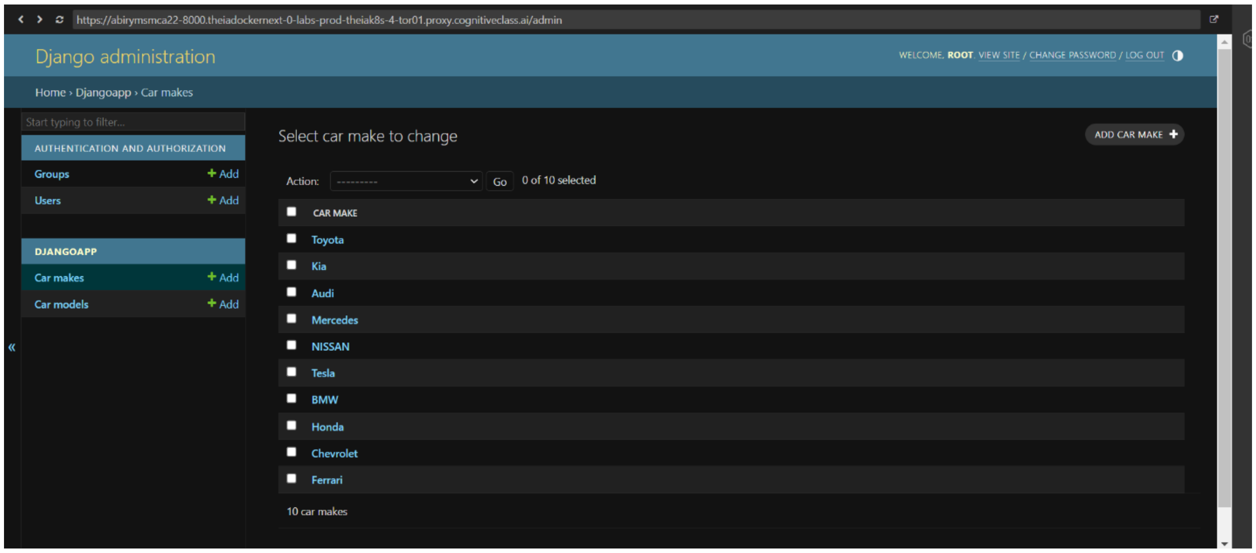
Thanks for spending some quality time with the web site today.

Log in again

**Task 14:**

Submit the screenshot (cars.png or cars.jpg) showing car makes (after adding or populating the car makes) from the admin page.

This task ensures that car makes can be added and viewed in the Django admin interface. It confirms the correct implementation of this feature, allowing administrators to manage the car inventory effectively.


https://abirymsmca22-8000.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/admin

Django administration

WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Djangoapp › Car makes

Select car make to change

ADD CAR MAKE

Action: --------- Go 0 of 10 selected

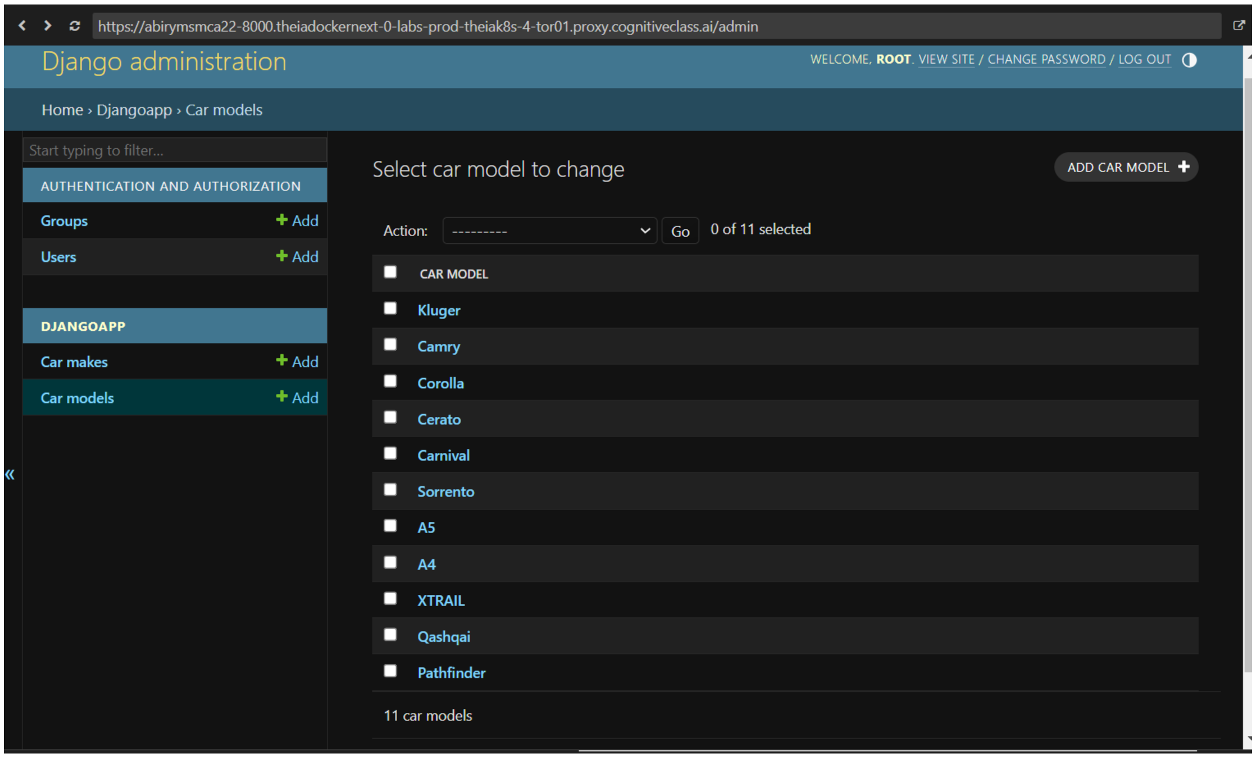
| CAR MAKE |
|---|
| Toyota |
| Kia |
| Audi |
| Mercedes |
| NISSAN |
| Tesla |
| BMW |
| Honda |
| Chevrolet |
| Ferrari |

10 car makes

**Task 15:**

Submit the screenshot (car_models.png or car_models.jpg) showing car models (after adding or populating the car makes) from the admin page.

This screenshot demonstrates the functionality of adding and viewing car models, verifying the relationship between car makes and models. This feature ensures comprehensive management of the car dealership's inventory.



https://abirymsmca22-8000.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/admin

Django administration

WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Djangoapp › Car models

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

DJANGOAPP

Car makes + Add

Car models + Add

Select car model to change

ADD CAR MODEL +

Action: --------- Go 0 of 11 selected

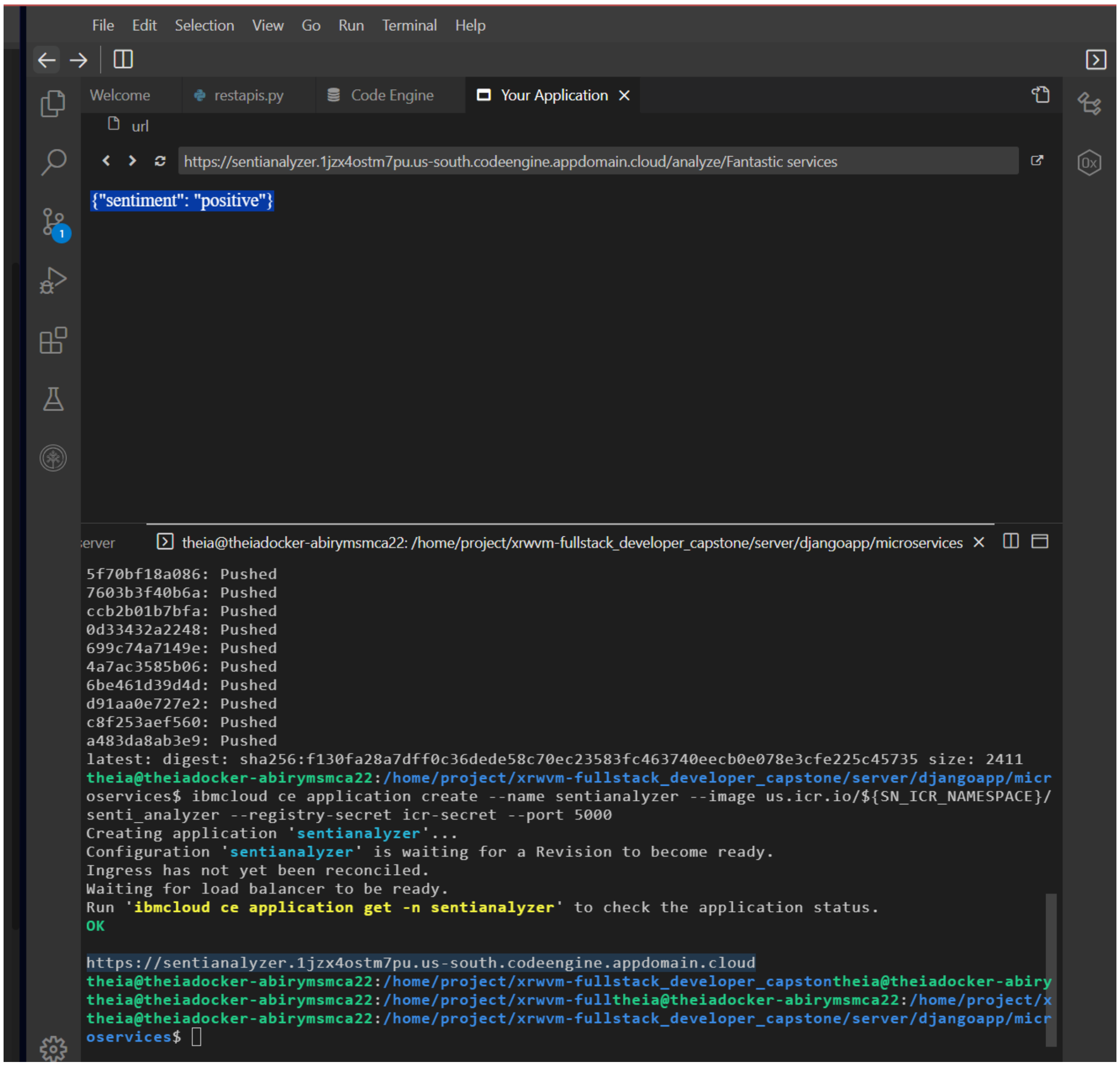
| CAR MODEL |
|---|
| Kluger |
| Camry |
| Corolla |
| Cerato |
| Carnival |
| Sorrento |
| A5 |
| A4 |
| XTRAIL |
| Qashqai |
| Pathfinder |

11 car models

**Task 16:**

Submit the screenshot (sentiment_analyzer.png or sentiment_analyzer.jpg) showing the sentiment analyzer working through the deployed URL.

This task demonstrates the sentiment analysis feature, which processes reviews and determines their sentiment. This showcases the application's machine learning capabilities, providing valuable insights into customer feedback.


File Edit Selection View Go Run Terminal Help
Welcome
restapis.py
Code Engine
Your Application
url
erver
theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server/djangoapp/microservices

https://sentianalyzer.1jzx4ostm7pu.us-south.codeengine.appdomain.cloud/analyze/Fantastic services

{"sentiment": "positive"}

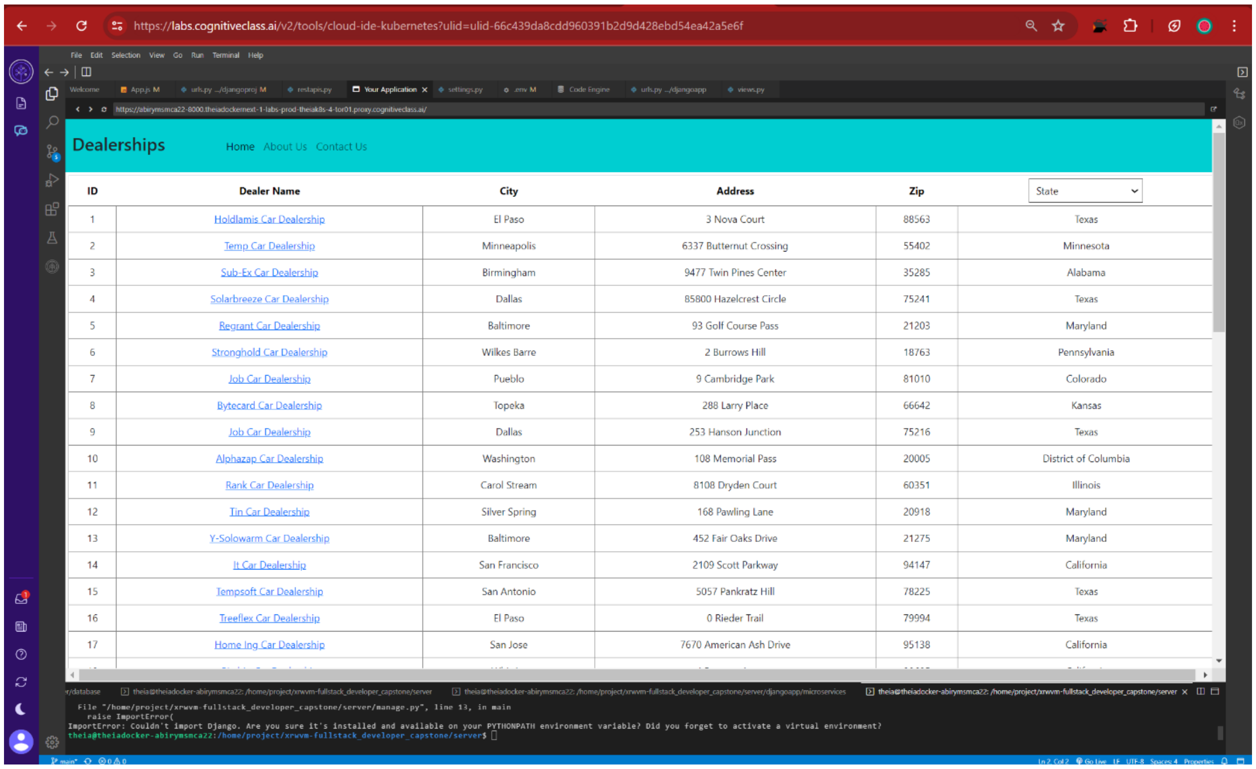
```
5f70bf18a086: Pushed
7603b3f40b6a: Pushed
ccb2b01b7bfa: Pushed
0d33432a2248: Pushed
699c74a7149e: Pushed
4a7ac3585b06: Pushed
6be461d39d4d: Pushed
d91aa0e727e2: Pushed
c8f253aef560: Pushed
a483da8ab3e9: Pushed
latest: digest: sha256:f130fa28a7dff0c36dede58c70ec23583fc463740eecb0e078e3cfe225c45735 size: 2411
theia@theiadocker-abirymsmca22:/home/project/xrwvm-fullstack_developer_capstone/server/djangoapp/micr
oservices$ ibmcloud ce application create --name sentianalyzer --image us.icr.io/${SN_ICR_NAMESPACE}/
senti_analyzer --registry-secret icr-secret --port 5000
Creating application 'sentianalyzer'...
Configuration 'sentianalyzer' is waiting for a Revision to become ready.
Ingress has not yet been reconciled.
Waiting for load balancer to be ready.
Run 'ibmcloud ce application get -n sentianalyzer' to check the application status.
OK

https://sentianalyzer.1jzx4ostm7pu.us-south.codeengine.appdomain.cloud
theia@theiadocker-abirymsmca22:/home/project/xrwvm-fullstack_developer_capstontheia@theiadocker-abiry
theia@theiadocker-abirymsmca22:/home/project/xrwvm-fulltheia@theiadocker-abirymsmca22:/home/project/x
theia@theiadocker-abirymsmca22:/home/project/xrwvm-fullstack_developer_capstone/server/djangoapp/micr
oservices$ 
```

## Task 17:

Submit the screenshot (get_dealers.png or get_dealers.jpg) showing the dealers on the home page of the Django application before logging in.

The screenshot shows the list of dealers visible to unauthenticated users, ensuring that public access to dealer information is properly implemented. This feature provides potential customers with essential dealership information.


https://labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-66c439da8cdd960391b2d9d428ebd54ea42a5e6f
File Edit Selection View Go Run Terminal Help
Welcome
App.js M
urls.py .../djangoproj M
restapis.py
Your Application
settings.py
.env M
Code Engine
urls.py .../djangoapp
views.py
https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
Dealerships
Home
About Us
Contact Us

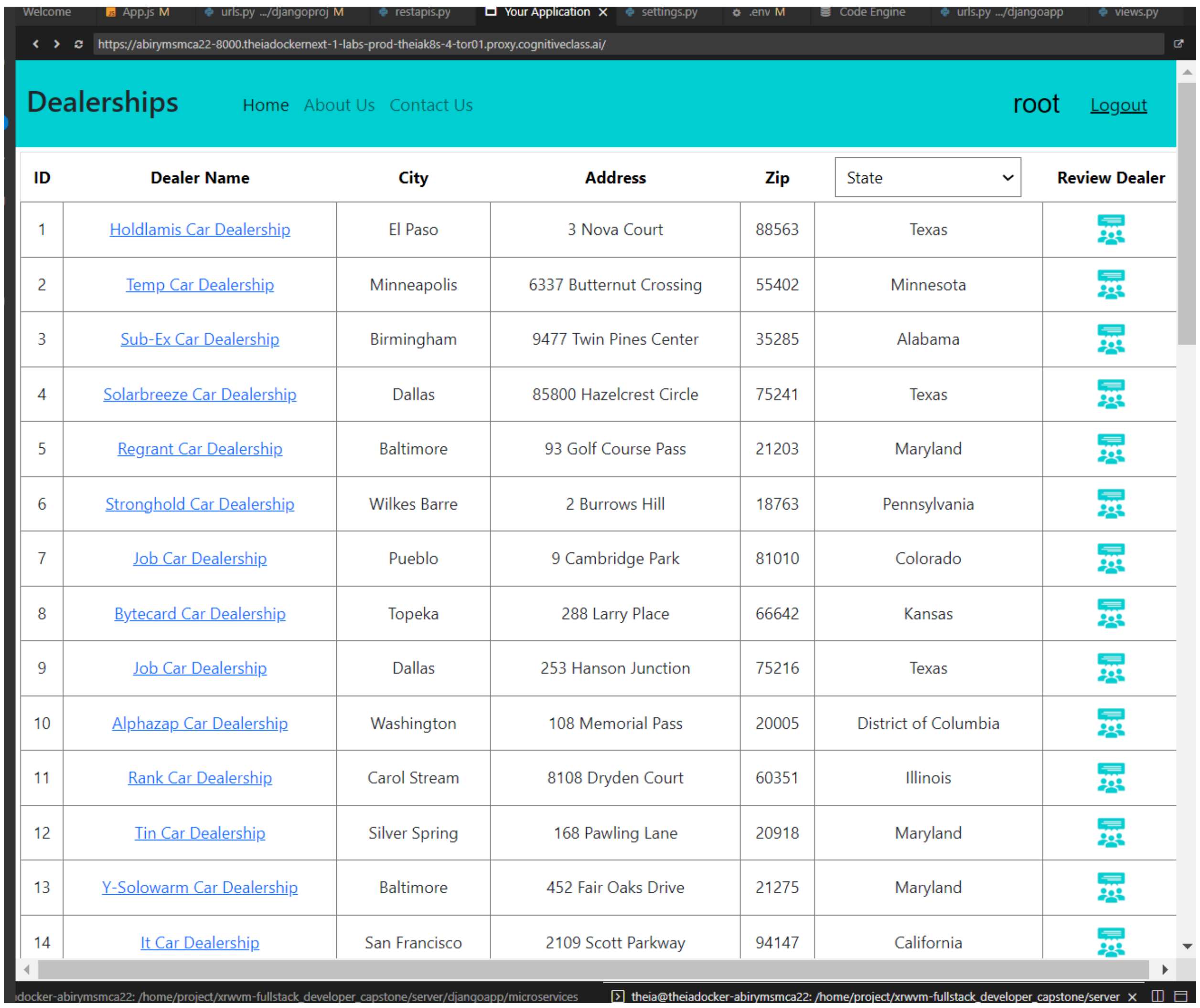
| ID | Dealer Name | City | Address | Zip | State |
|---|---|---|---|---|---|
| 1 | Holdlamis Car Dealership | El Paso | 3 Nova Court | 88563 | Texas |
| 2 | Temp Car Dealership | Minneapolis | 6337 Butternut Crossing | 55402 | Minnesota |
| 3 | Sub-Ex Car Dealership | Birmingham | 9477 Twin Pines Center | 35285 | Alabama |
| 4 | Solarbreeze Car Dealership | Dallas | 85800 Hazelcrest Circle | 75241 | Texas |
| 5 | Regrant Car Dealership | Baltimore | 93 Golf Course Pass | 21203 | Maryland |
| 6 | Stronghold Car Dealership | Wilkes Barre | 2 Burrows Hill | 18763 | Pennsylvania |
| 7 | Job Car Dealership | Pueblo | 9 Cambridge Park | 81010 | Colorado |
| 8 | Bytecard Car Dealership | Topeka | 288 Larry Place | 66642 | Kansas |
| 9 | Job Car Dealership | Dallas | 253 Hanson Junction | 75216 | Texas |
| 10 | Alphazap Car Dealership | Washington | 108 Memorial Pass | 20005 | District of Columbia |
| 11 | Rank Car Dealership | Carol Stream | 8108 Dryden Court | 60351 | Illinois |
| 12 | Tin Car Dealership | Silver Spring | 168 Pawling Lane | 20918 | Maryland |
| 13 | Y-Solowarm Car Dealership | Baltimore | 452 Fair Oaks Drive | 21275 | Maryland |
| 14 | It Car Dealership | San Francisco | 2109 Scott Parkway | 94147 | California |
| 15 | Tempsoft Car Dealership | San Antonio | 5057 Pankratz Hill | 78225 | Texas |
| 16 | Treeflex Car Dealership | El Paso | 0 Rieder Trail | 79994 | Texas |
| 17 | Home Ing Car Dealership | San Jose | 7670 American Ash Drive | 95138 | California |

```
r/database   theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server   theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server/djangoapp/microservices   theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server
  File "/home/project/xrwvm-fullstack_developer_capstone/server/manage.py", line 13, in main
    raise ImportError(
ImportError: Couldn't import Django. Are you sure it's installed and available on your PYTHONPATH environment variable? Did you forget to activate a virtual environment?
theia@theiadocker-abirymsmca22:/home/project/xrwvm-fullstack_developer_capstone/server$ 
```

**Task 18**:

Submit the screenshot (get_dealers_loggedin.png or get_dealers_loggedin.jpg) showing the dealers on the home page of the Django application after logging in.

This task verifies that logged-in users can see additional functionality, such as the "Post Review" button, ensuring proper role-based access control. This feature allows authenticated users to contribute reviews.



| ID | Dealer Name | City | Address | Zip | State | Review Dealer |
|---|---|---|---|---|---|---|
| 1 | Holdlamis Car Dealership | El Paso | 3 Nova Court | 88563 | Texas | |
| 2 | Temp Car Dealership | Minneapolis | 6337 Butternut Crossing | 55402 | Minnesota | |
| 3 | Sub-Ex Car Dealership | Birmingham | 9477 Twin Pines Center | 35285 | Alabama | |
| 4 | Solarbreeze Car Dealership | Dallas | 85800 Hazelcrest Circle | 75241 | Texas | |
| 5 | Regrant Car Dealership | Baltimore | 93 Golf Course Pass | 21203 | Maryland | |
| 6 | Stronghold Car Dealership | Wilkes Barre | 2 Burrows Hill | 18763 | Pennsylvania | |
| 7 | Job Car Dealership | Pueblo | 9 Cambridge Park | 81010 | Colorado | |
| 8 | Bytecard Car Dealership | Topeka | 288 Larry Place | 66642 | Kansas | |
| 9 | Job Car Dealership | Dallas | 253 Hanson Junction | 75216 | Texas | |
| 10 | Alphazap Car Dealership | Washington | 108 Memorial Pass | 20005 | District of Columbia | |
| 11 | Rank Car Dealership | Carol Stream | 8108 Dryden Court | 60351 | Illinois | |
| 12 | Tin Car Dealership | Silver Spring | 168 Pawling Lane | 20918 | Maryland | |
| 13 | Y-Solowarm Car Dealership | Baltimore | 452 Fair Oaks Drive | 21275 | Maryland | |
| 14 | It Car Dealership | San Francisco | 2109 Scott Parkway | 94147 | California | |

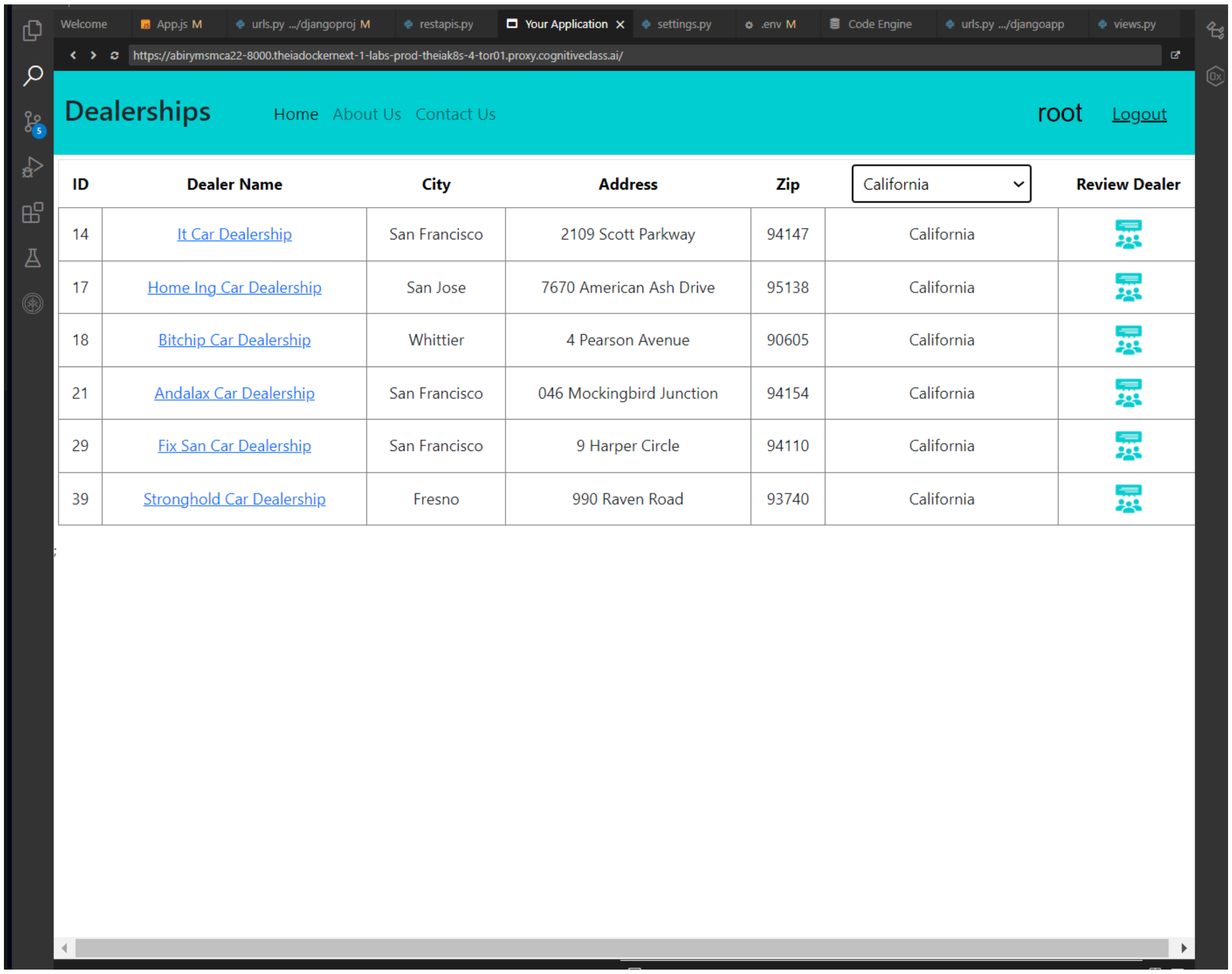
```
idocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server/djangoapp/microservices
theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server
```

46

**Task 19:**

Submit the screenshot (dealersbystate.png or dealersbystate.jpg) showing the dealers filtered by the State on the home page of the Django application.

This screenshot demonstrates the filtering functionality on the frontend, allowing users to view dealers by state. This enhances user experience by providing customized search results.



https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

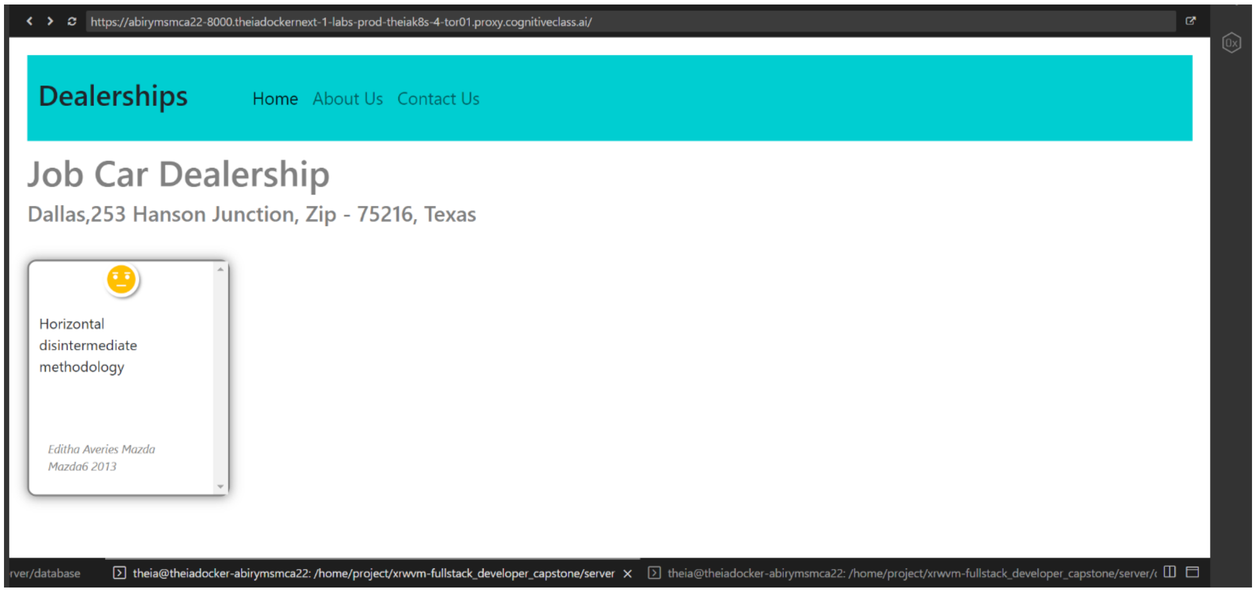
Dealerships Home About Us Contact Us root Logout

| ID | Dealer Name | City | Address | Zip | California | Review Dealer |
|---|---|---|---|---|---|---|
| 14 | It Car Dealership | San Francisco | 2109 Scott Parkway | 94147 | California | |
| 17 | Home Ing Car Dealership | San Jose | 7670 American Ash Drive | 95138 | California | |
| 18 | Bitchip Car Dealership | Whittier | 4 Pearson Avenue | 90605 | California | |
| 21 | Andalax Car Dealership | San Francisco | 046 Mockingbird Junction | 94154 | California | |
| 29 | Fix San Car Dealership | San Francisco | 9 Harper Circle | 94110 | California | |
| 39 | Stronghold Car Dealership | Fresno | 990 Raven Road | 93740 | California | |

**Task 20:**

Submit the screenshot (dealer_id_reviews.png or dealer_id_reviews.jpg) showing the selected dealer details on the dealer page along with the reviews.

This task confirms that users can view detailed dealer information and their reviews, ensuring the front-end correctly displays detailed data. This feature helps users make informed decisions based on dealer reviews.


https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Dealerships Home About Us Contact Us

Job Car Dealership

Dallas,253 Hanson Junction, Zip - 75216, Texas

Horizontal disintermediate methodology

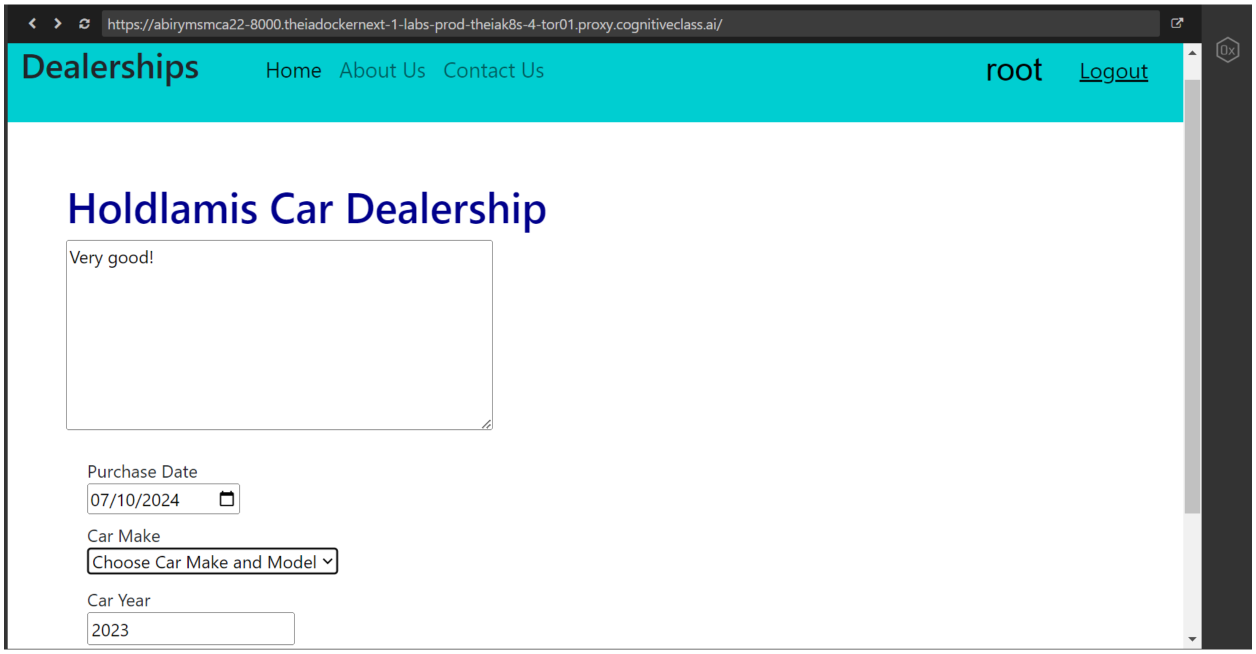
*Editha Averies Mazda Mazda6 2013*

rver/database theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server/

**Task 21**:

Submit the screenshot (dealership_review_submission.png or dealership_review_submission.jpg) showing the "Post Review" page after adding the details before you submit.

The screenshot shows the review submission form filled out, ready to be submitted. This ensures the front-end form handling is functioning correctly, allowing users to submit reviews.


https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
Dealerships
Home
About Us
Contact Us
root
Logout

Holdlamis Car Dealership

Very good!

Purchase Date
07/10/2024

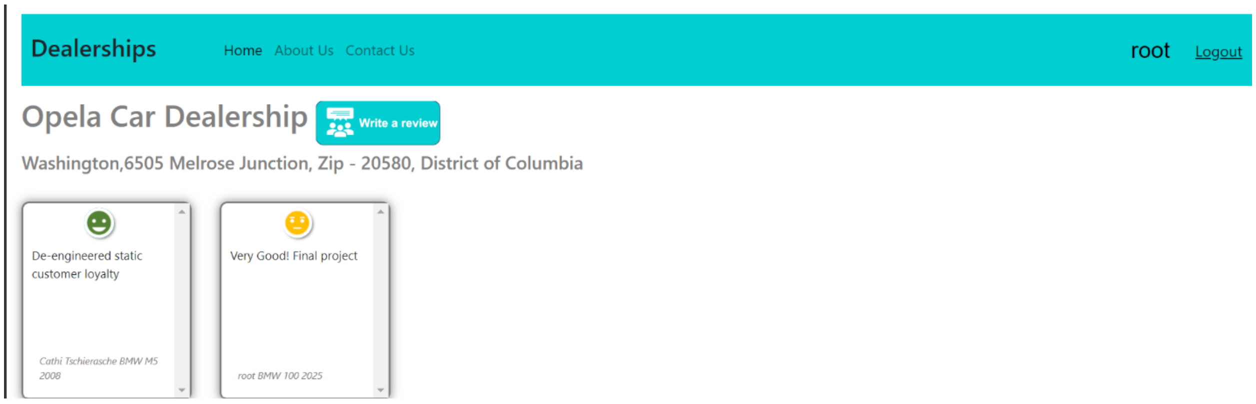
Car Make
Choose Car Make and Model

Car Year
2023

**Task 22:**

Submit the screenshot (added_review.png or added_review.jpg) showing the review you posted. The details should match with the input shown in the previous task.

This task confirms that reviews are correctly posted and stored in the database, demonstrating the end-to-end functionality of the review feature. This provides users with a platform to share their experiences.



Opela Car Dealership

Washington,6505 Melrose Junction, Zip - 20580, District of Columbia

De-engineered static customer loyalty

Cathi Tschierasche BMW M5 2008

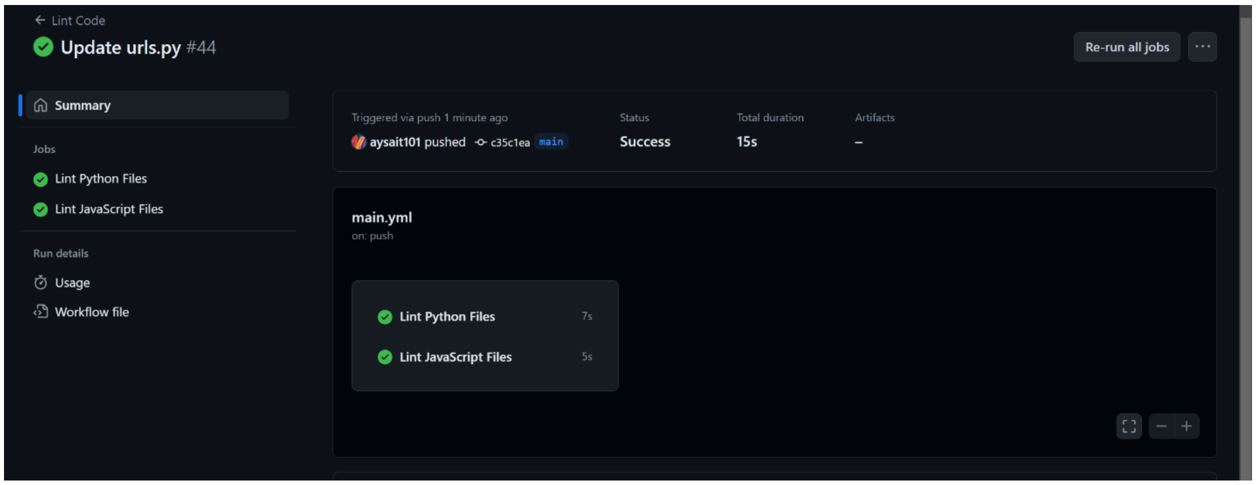
Very Good! Final project

root BMW 100 2025

**Task 23:**

Submit the screenshot (CICD.png or CICD.jpg) of the successful implementation of CI/CD on GitHub.

This screenshot shows the continuous integration and continuous deployment (CI/CD) pipeline, ensuring automated testing and deployment are set up correctly. This improves the development workflow and ensures consistent deployment.



Triggered via push 1 minute ago
aysait101 pushed c35c1ea main

| Status | Total duration | Artifacts |
|---|---|---|
| Success | 15s | – |

main.yml
on: push

Lint Python Files 7s
Lint JavaScript Files 5s

**Task 24:**

Submit the deployment URL for your Django application.

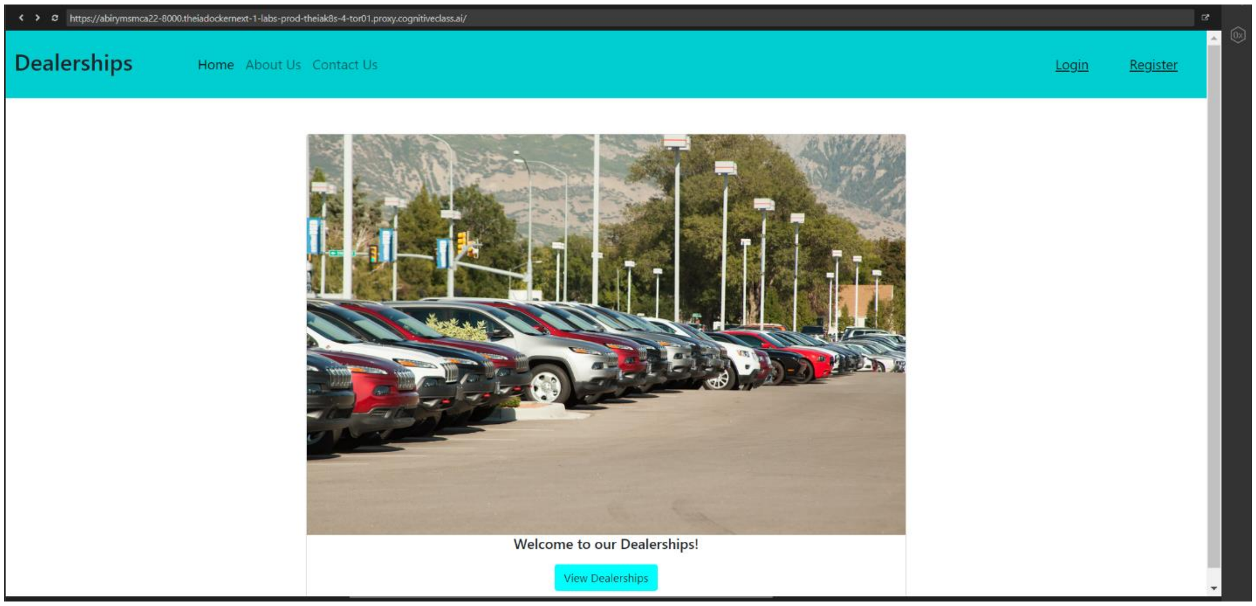
The deployment URL provides access to the live version of the Django application. This makes the application accessible for testing and demonstration purposes, showcasing the successful deployment.

https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

**Task 25:**

Submit the screenshot (deployed_landingpage.png or deployed_landingpage.jpg) of the landing page opened through your deployment.

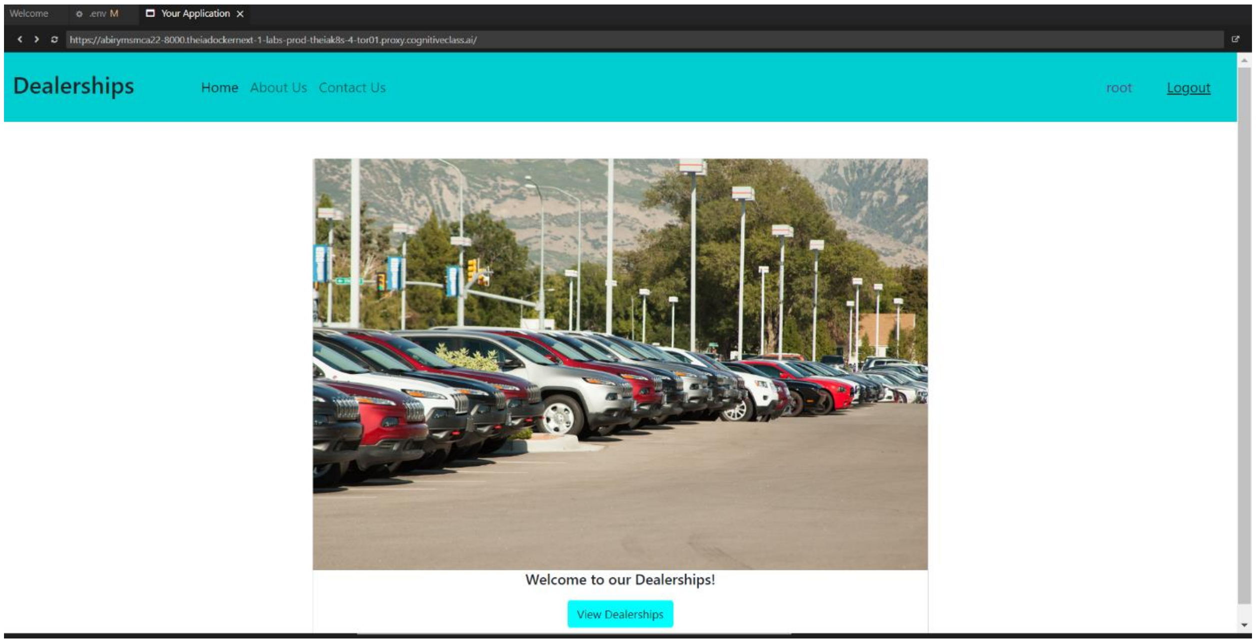
This task confirms that the landing page of the deployed application is accessible and functional, showcasing the successful deployment. It provides users with the initial interface of the application.


https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
Dealerships
Home
About Us
Contact Us
Login
Register
Welcome to our Dealerships!
View Dealerships

**Task 26:**

Submit the screenshot (deployed_loggedin.png or deployed_loggedin.jpg) of the logged-in page opened through your deployment.

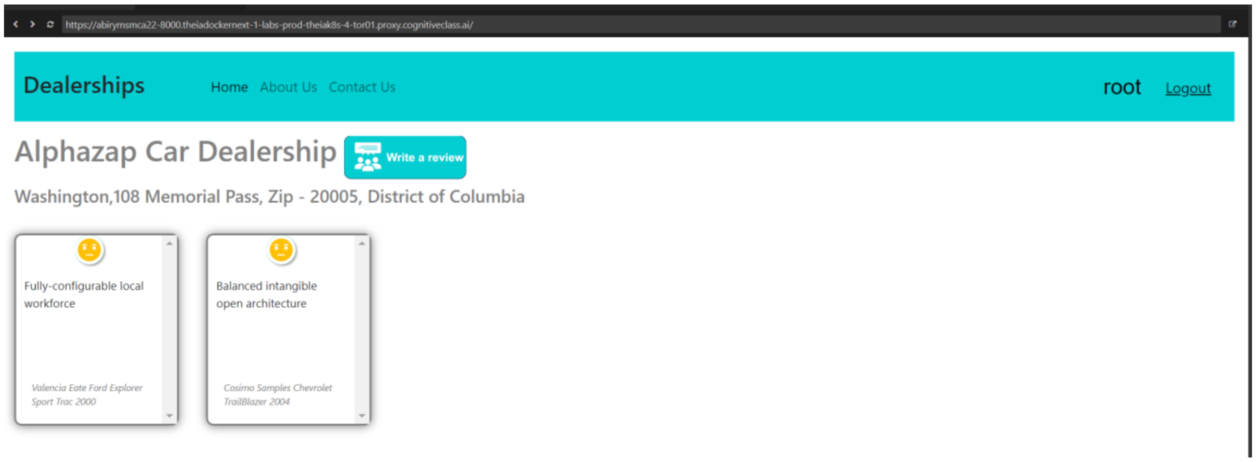
The screenshot demonstrates a user successfully logging into the deployed application. This ensures that authentication works on the live site, providing secure access to user-specific features.


Welcome
.env M
Your Application
https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
Dealerships
Home
About Us
Contact Us
root
Logout
Welcome to our Dealerships!
View Dealerships

**Task 27:**

Submit the screenshot (deployed_dealer_detail.png or deployed_dealer_detail.jpg) of a dealer details page opened through your deployment.

This task confirms that detailed dealer information is correctly displayed on the deployed application. It verifies the functionality in the production environment, ensuring users can access comprehensive dealer details.


https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
Dealerships
Home
About Us
Contact Us
root
Logout
Write a review

Alphazap Car Dealership

Washington,108 Memorial Pass, Zip - 20005, District of Columbia

Fully-configurable local workforce

*Valencia Eate Ford Explorer Sport Trac 2000*

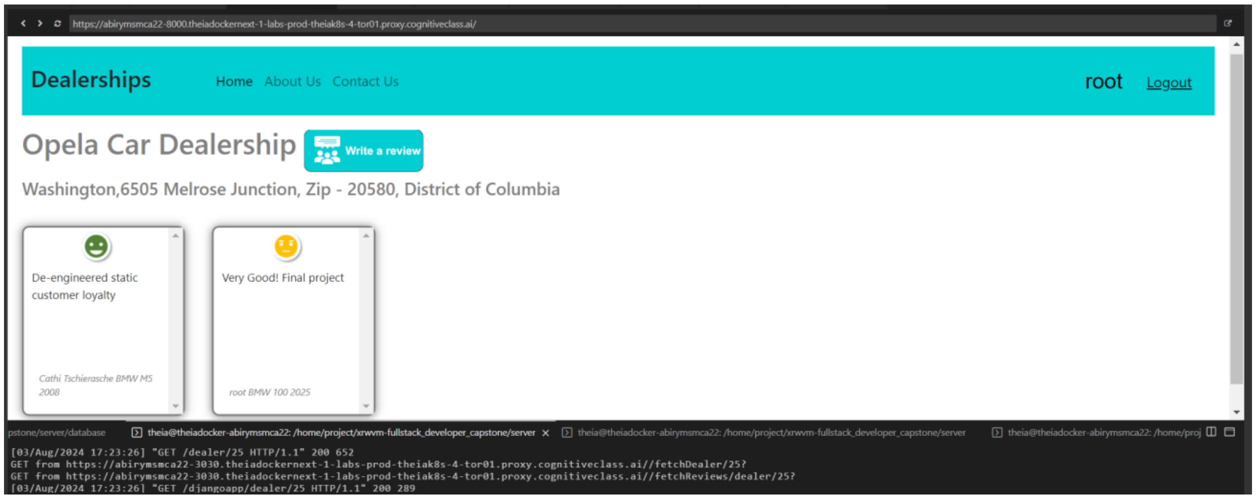
Balanced intangible open architecture

*Cosimo Samples Chevrolet TrailBlazer 2004*

**Task 28:**

Submit the screenshot (deployed_add_review.png or deployed_add_review.jpg) of a review added through your deployed application.

The screenshot demonstrates the functionality of adding a review on the live application. This ensures that the review feature works post-deployment, allowing users to share their experiences on the live platform.



https://abirymsmca22-8000.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Dealerships Home About Us Contact Us root Logout

Opela Car Dealership Write a review

Washington,6505 Melrose Junction, Zip - 20580, District of Columbia

De-engineered static customer loyalty

*Cathi Tschierasche BMW M5 2008*

Very Good! Final project

*root BMW 100 2025*

pstone/server/database theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server theia@theiadocker-abirymsmca22: /home/project/xrwvm-fullstack_developer_capstone/server theia@theiadocker-abirymsmca22: /home/proj

```
[03/Aug/2024 17:23:26] "GET /dealer/25 HTTP/1.1" 200 652
GET from https://abirymsmca22-3030.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai//fetchDealer/25?
GET from https://abirymsmca22-3030.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai//fetchReviews/dealer/25?
[03/Aug/2024 17:23:26] "GET /djangoapp/dealer/25 HTTP/1.1" 200 289
```

# Chapter 6
# Conclusion

Completing the IBM Full Stack Web Developer Professional Certificate from Coursera has been an enriching and transformative experience. This comprehensive program has provided a thorough understanding of both front-end and back-end web development, equipping us with the skills necessary to build, deploy, and maintain full stack web applications.

Throughout the course, we have gained proficiency in a variety of technologies and frameworks, including HTML, CSS, JavaScript, React, Node.js, Express, MongoDB, Python, Django, Docker, and Kubernetes. The structured curriculum and hands-on projects have allowed us to apply theoretical knowledge to practical scenarios, reinforcing our learning and providing valuable real-world experience.

Key highlights of the program include:

- Frontend Development: Learning to create responsive and interactive user interfaces using HTML, CSS, JavaScript, and React. We have mastered the principles of modern front-end development, including state management, component-based architecture, and responsive design.
- Backend Development: Gaining expertise in server-side programming with Node.js, Express, and Django. We have learned to design and implement RESTful APIs, manage databases with MongoDB, and handle user authentication and authorization.
- DevOps and Deployment: Understanding the importance of DevOps practices, including containerization with Docker and orchestration with Kubernetes. We have also implemented continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process and ensure the reliability of our applications.
- Project Management and Collaboration: Developing skills in project management, version control with Git, and collaborative development using GitHub. These skills are essential for working effectively in a team environment and managing complex projects.

The capstone project, which involved building a fully functional car dealership web application, was a significant milestone that demonstrated our ability to integrate various technologies and frameworks into a cohesive and scalable solution. This project allowed us to showcase our skills in both front-end and back-end development, as well as our ability to manage and deploy applications in a cloud environment.

Overall, completing the IBM Full Stack Web Developer Professional Certificate from Coursera has been a rigorous and rewarding journey. The program has provided a solid foundation for a career in web development, equipping us with the technical skills and practical experience needed to excel in the field. We are now well-prepared to tackle real-world challenges and contribute effectively to the dynamic and ever-evolving landscape of web development.

# References

**References**

1. Django Documentation:
    - Django Software Foundation. (2024). Django Documentation. Available at: https://docs.djangoproject.com/en/stable/
2. Express.js Documentation:
    - Express.js. (2024). Express - Node.js web application framework. Available at: https://expressjs.com/en/4x/api.html
3. MongoDB Documentation:
    - MongoDB, Inc. (2024). MongoDB Manual. Available at: https://docs.mongodb.com/manual/
4. Docker Documentation:
    - Docker, Inc. (2024). Docker Documentation. Available at: https://docs.docker.com/
5. Kubernetes Documentation:
    - The Kubernetes Authors. (2024). Kubernetes Documentation. Available at: https://kubernetes.io/docs/home/
6. GitHub Actions Documentation:
    - GitHub, Inc. (2024). GitHub Actions Documentation. Available at: https://docs.github.com/en/actions
7. IBM Cloud Documentation:
    - IBM. (2024). IBM Cloud Documentation. Available at: https://cloud.ibm.com/docs
8. Python Documentation:
    - Python Software Foundation. (2024). Python 3.12 Documentation. Available at: https://docs.python.org/3.12/
9. React Documentation:
    - Meta Platforms, Inc. (2024). React – A JavaScript library for building user interfaces. Available at: https://reactjs.org/docs/getting-started.html
10. Full-Stack Web Development with React and Django:
    - GitHub Repository for project: https://github.com/aysait101/xrwvm-fullstack_developer_capstone

11. IBM Full-Stack Software Developer Professional Certificate - Full-Stack Application Development:

    - Ober, T. (2023). IBM Full-Stack Software Developer Professional Certificate - Full-Stack Application Development. Available at: https://medium.com/@timothy.ober/ibm-full-stack-software-developer-professional-certificate-full-stack-application-development-94842bfebbd3

# Appendix A : Source code

The complete source code for this project is available in the GitHub repository linked below. This repository includes all the files and directories used in the development of the Full Stack Web Application for a national car dealership.

**GitHub Repository**: https://github.com/aysait101/xrwvm-fullstack_developer_capstone

Key Directories and Files:

1. server:
   - Dockerfile: Contains instructions for building the Docker image.
   - entrypoint.sh: Shell script for running initial setup commands.
   - requirements.txt: Lists all the dependencies required for the project.
   - manage.py: Command-line utility for administrative tasks.
   - djangoproj/: Main Django project directory containing settings, URLs, and WSGI configurations.

2. /client:
   - src/: Contains the React components and application logic.
   - public/: Static assets and the main HTML template.

3. /api:
   - server.js: Main server file for Express.js.
   - routes/: Contains route definitions for the API endpoints.
   - models/: Defines the data models used by MongoDB.

4. /scripts:
   - ci-cd/: Contains configuration files for setting up CI/CD pipelines using GitHub Actions.
   - kubernetes/: Contains YAML files for Kubernetes deployments and services.

Code Snippets:

- About.html

```
  <div class="card" style="width: 80%;margin: auto; margin-top:5%;">
      <div class="banner" name="about-header">
  <h1>About Us</h1>
  Welcome to Best Cars dealership, home to the best cars in North America. We
deal in sale of domestic and imported cars at reasonable prices.      </div>
      <div style="display: flex;flex-direction: row; margin:auto">
      <div class="card" style="width: 30%;">
        <img class="card-img-top" src="/static/realperson.png" alt="Card
image">
        <div class="card-body">
          <p class="title">Jane Doe
</p>
          <p>Sales Manager
</p>
          <p class="card-text">Jane brings over 10 years of experience in the
automotive industry, ensuring every customer finds their perfect vehicle. Her
dedication and expertise make her a valuable asset to our team.</p>
          <p>jane.doe@example.com</p>
        </div>
      </div>
      <div class="card" style="width: 30%;">
        <img class="card-img-top" src="/static/realperson.png" alt="Card
image">
        <div class="card-body">
          <p class="title">John Smith</p>
          <p>Finance Specialist</p>
          <p class="card-text">With a keen eye for detail and a passion for
helping clients secure the best financing options, John has been a
cornerstone of our dealership for over 8 years. His friendly demeanor and
financial acumen are unmatched.</p>
          <p>john.smith@example.com</p>
        </div>
      </div>
      <div class="card" style="width: 30%;">
        <img class="card-img-top" src="/static/realperson.png" alt="Card
image">
        <div class="card-body">
          <p class="title">Emily Johnson</p>
          <p>Customer Service Representative
          </p>
          <p class="card-text">Emily's warm and welcoming personality has
made her a favorite among our customers. With over 5 years of experience in
customer service, she ensures that every visit to our dealership is a
pleasant one.</p>
          <p>emily.johnson@example.com</p>
        </div>
      </div>
```

- Server/djangoproj/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.views.generic import TemplateView
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path('djangoapp/', include('djangoapp.urls')),
    path('', TemplateView.as_view(template_name="Home.html")),
    path('about/', TemplateView.as_view(template_name="About.html")),
    path('contact/', TemplateView.as_view(template_name="contact.html")),
    path('login/', TemplateView.as_view(template_name="index.html")),
    path('register/', TemplateView.as_view(template_name="index.html")),
        path('dealers/', TemplateView.as_view(template_name="index.html")),

path('dealer/<int:dealer_id>',TemplateView.as_view(template_name="index.html"
)),

path('postreview/<int:dealer_id>',TemplateView.as_view(template_name="index.h
tml")),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

- Server/djangoproj/settings.py

```
TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(BASE_DIR, 'frontend/static'),

            os.path.join(BASE_DIR, 'frontend/build'),

            os.path.join(BASE_DIR, 'frontend/build/static'),],

        'APP_DIRS': True,

        'OPTIONS': {

            'context_processors': [

                'django.template.context_processors.debug',

                'django.template.context_processors.request',

                'django.contrib.auth.context_processors.auth',

                'django.contrib.messages.context_processors.messages',
            ],},},]


STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'frontend/static'),
    os.path.join(BASE_DIR, 'frontend/build'),
    os.path.join(BASE_DIR, 'frontend/build/static'),
]
```

- server/frontend/static/Home.html

```
  let logout_url = window.location.origin+"/djangoapp/logout";
  const res = await fetch(logout_url, {
    method: "GET",
  });


  const json = await res.json();
  if (json) {
    let username = sessionStorage.getItem('username');
    sessionStorage.removeItem('username');
    window.location.href = window.location.origin;
    window.location.reload();
    alert("Logging out "+username+"...")
  }
  else {
    alert("The user could not be logged out.")
  }
```

- djangoapp/views.py

```
    logout(request)
    data = {"userName":""}
    return JsonResponse(data)


@csrf_exempt
```

```
def registration(request):
    context = {}

    data = json.loads(request.body)
    username = data['userName']
    password = data['password']
    first_name = data['firstName']
    last_name = data['lastName']
    email = data['email']
    username_exist = False
    email_exist = False
    try:
        # Check if user already exists
        User.objects.get(username=username)
        username_exist = True
    except:
        # If not, simply log this is a new user
        logger.debug("{} is new user".format(username))

    # If it is a new user
    if not username_exist:
        # Create user in auth_user table
        user = User.objects.create_user(username=username,
first_name=first_name, last_name=last_name,password=password, email=email)
        # Login the user and redirect to list page
        login(request, user)
        data = {"userName":username,"status":"Authenticated"}
        return JsonResponse(data)
    else :
        data = {"userName":username,"error":"Already Registered"}
        return JsonResponse(data)

def get_cars(request):
    count = CarMake.objects.filter().count()
    print(count)
    if(count == 0):
        initiate()
    car_models = CarModel.objects.select_related('car_make')
    cars = []
    for car_model in car_models:
        cars.append({"CarModel": car_model.name, "CarMake":
car_model.car_make.name})
    return JsonResponse({"CarModels":cars})
```

- server/djangoapp/models.py

```
class CarMake(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    # Other fields as needed

    def __str__(self):
        return self.name  # Return the name as the string representation
```

```
# <HINT> Create a Car Model model `class CarModel(models.Model):`:
class CarModel(models.Model):
    car_make = models.ForeignKey(CarMake, on_delete=models.CASCADE)  # Many-
to-One relationship
    name = models.CharField(max_length=100)
    CAR_TYPES = [
        ('SEDAN', 'Sedan'),
        ('SUV', 'SUV'),
        ('WAGON', 'Wagon'),
        # Add more choices as required
    ]
    type = models.CharField(max_length=10, choices=CAR_TYPES, default='SUV')
    year = models.IntegerField(default=2023,
        validators=[
            MaxValueValidator(2023),
            MinValueValidator(2015)
        ])
    dealer_id = models.IntegerField(default=1)
    # Other fields as needed
```

- server/frontend/src/App.js

```
import LoginPanel from "./components/Login/Login";
import Registerss from "./components/Register/Register";
import { Routes, Route } from "react-router-dom";
import Dealers from './components/Dealers/Dealers';
import Dealer from "./components/Dealers/Dealer";
import PostReview from "./components/Dealers/PostReview"

function App() {
  return (
    <Routes>
      <Route path="/login" element={<LoginPanel />} />
      <Route path="/register" element={<Registerss />} />
            <Route path="/dealers" element={<Dealers/>} />
            <Route path="/dealer/:id" element={<Dealer/>} />
             <Route path="/postreview/:id" element={<PostReview/>} />
    </Routes>
  );
}

export default App;
```

- server/Dockerfile

```
FROM python:3.12.0-slim-bookworm

ENV PYTHONBUFFERED 1
ENV PYTHONWRITEBYTECODE 1

ENV APP=/app

# Change the workdir.
WORKDIR $APP

# Install the requirements
COPY requirements.txt $APP

RUN pip3 install -r requirements.txt

# Copy the rest of the files
COPY . $APP

EXPOSE 8000

RUN chmod +x /app/entrypoint.sh

ENTRYPOINT ["/bin/bash","/app/entrypoint.sh"]

CMD ["gunicorn", "--bind", ":8000", "--workers", "3", "djangoproj.wsgi"]
```

- server/entrypoint.sh

```
#!/bin/sh


# Make migrations and migrate the database.

echo "Making migrations and migrating the database. "

python manage.py makemigrations --noinput

python manage.py migrate --noinput

python manage.py collectstatic --noinput

exec "$@"
```