Architecture Overview

The final project for this course has several steps that you must complete. The high-level step list given below will help you with an overview of the complete project. The project is divided into smaller labs with detailed instructions for each step. You must complete all labs to complete the project successfully.

Project breakdown

Fork the GitHub repo containing the project template. The main web application is a predefined Django application. You will need to add some new features, and then build and run your project implementation.

1. Fork the repository in your account.
2. Clone the repository in the Cloud IDE environment.
3. Create static pages to finish the user stories.
4. Run the application locally.

Add user management to the Django application.

1. Implement user management using the Django user authentication system and create a REACT frontend.

Implement backend services.

1. Create Node.js server to manage dealers and reviews using MongoDB and dockerize it.
2. Deploy sentiment analyzer on Code Engine.
3. Create Django models and views to manage car model and car make.
4. Create Django proxy services and views to integrate dealers and reviews together.

Add dynamic pages with Django templates.

1. Create a page that displays all the dealers.
2. Create a page that displays reviews for a selected dealer.
3. Create a page that lets the end user add a review for a selected dealer.

Implement CI/CD, and then run and test your application

1. Set up continuous integration and delivery for code linting.
2. Run your application on Cloud IDE.
3. Test the updated application locally.
4. Deploy the application on Kubernetes.

Solution architecture

The solution will consist of multiple technologies

1. The user interacts with the "Dealerships Website", a Django website, through a web browser.

2. 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

3. The Django application uses SQLite database to store the `Car Make` and the `Car Model` data.

4. The "Dealerships and Reviews Service" is an Express Mongo service running in a Docker container. It provides the following services::

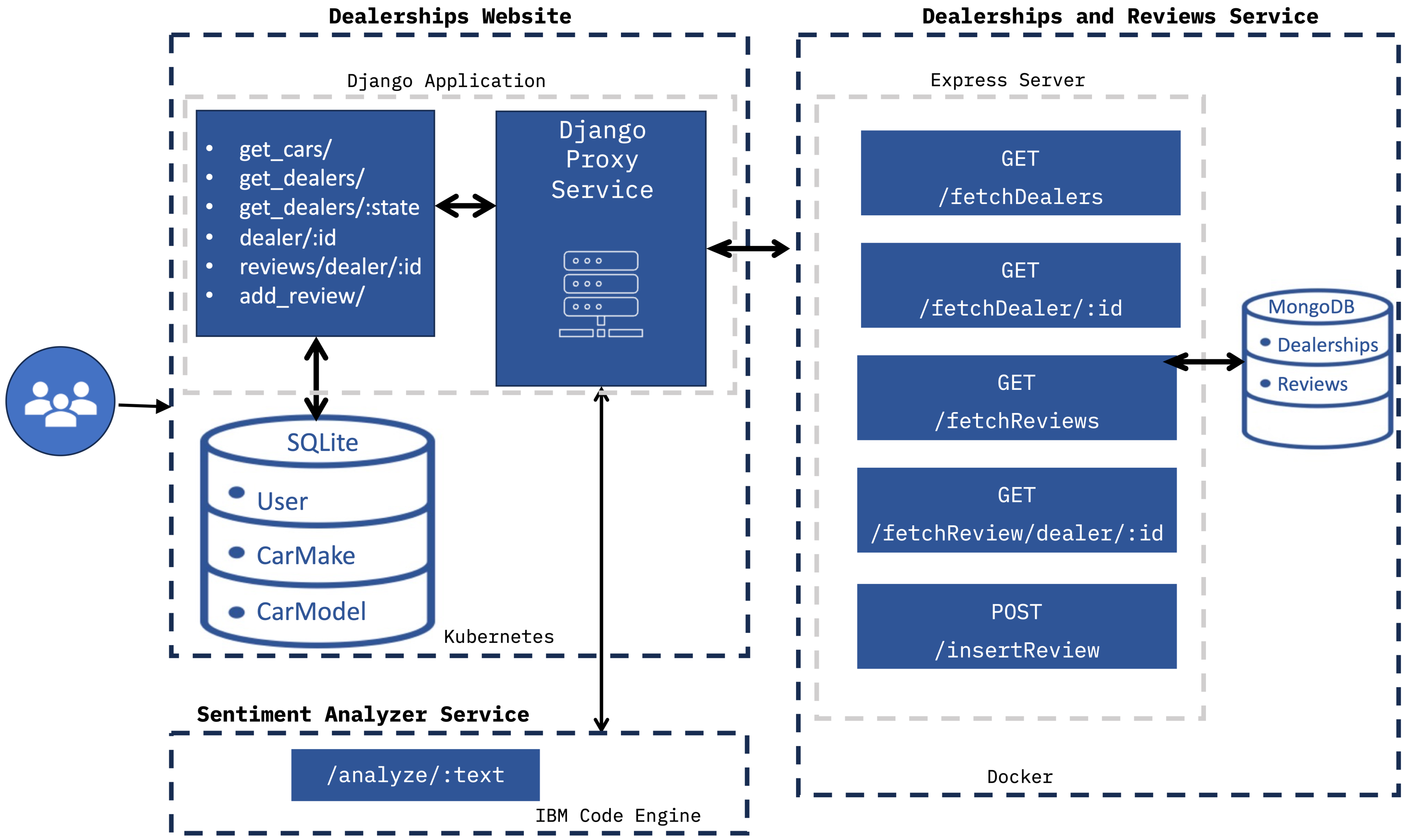
- **/fetchDealers** - To fetch the dealers
- **/fetchDealer/:id** - To fetch the dealer by id
- **fetchReviews** - To fetch all the reviews
- **fetchReview/dealer/:id** - To fetch reviews for a dealer by id
- **/insertReview** - To insert a review

5. "Dealerships Website" interacts with the "Dealership and Reviews Service" through the "Django Proxy Service" contained within the Django Application.

6. 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*.

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



Submit your project for grading

You will need to submit several screenshots for each module. The instructions will give you filenames and samples for each screenshot. Make sure to use the specified filename, as your reviewer will identify the correct screenshot based on the filename.

Next Steps

To complete the lab, follow the step-by-step instructions given in the lab, starting with the capstone prework.

Author(s)

[Upkar Lidder]

[Lavanya]

Skills Network

IBM