CSCI 5410: Assignment 1

Part A: Build, Deploy and Run a Containerized Application using GCP

Screenshots of the steps:

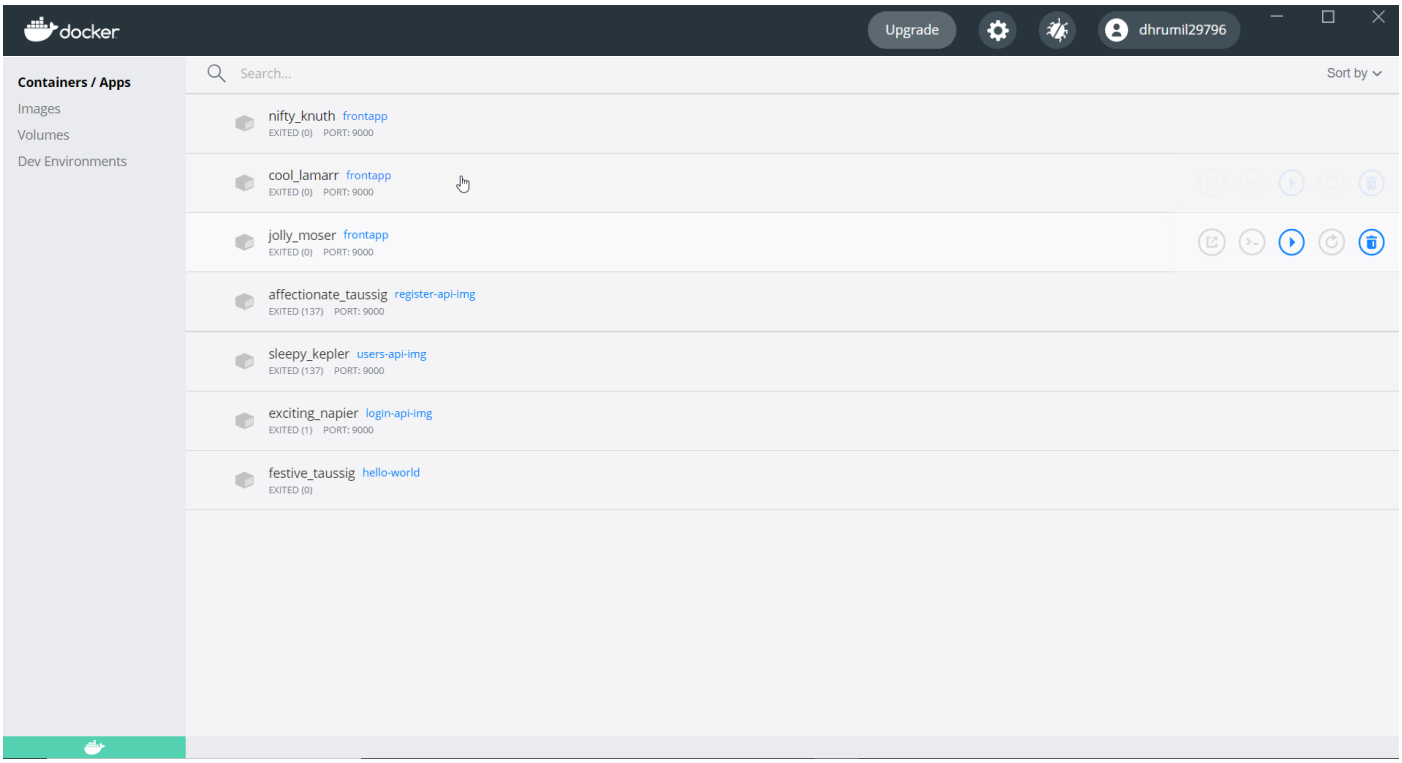

Figure 1: Docker Images

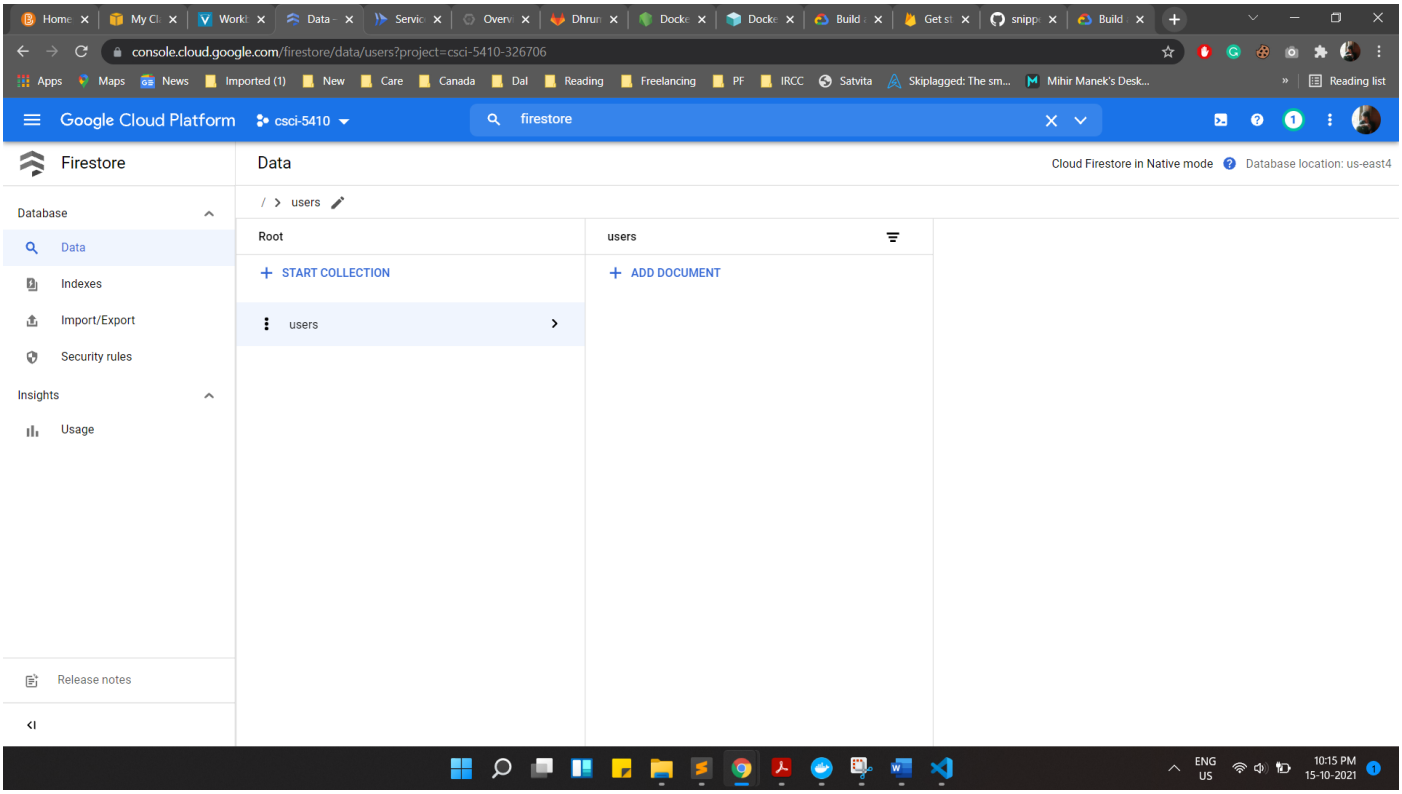

Figure 2: Firestore Empty Database

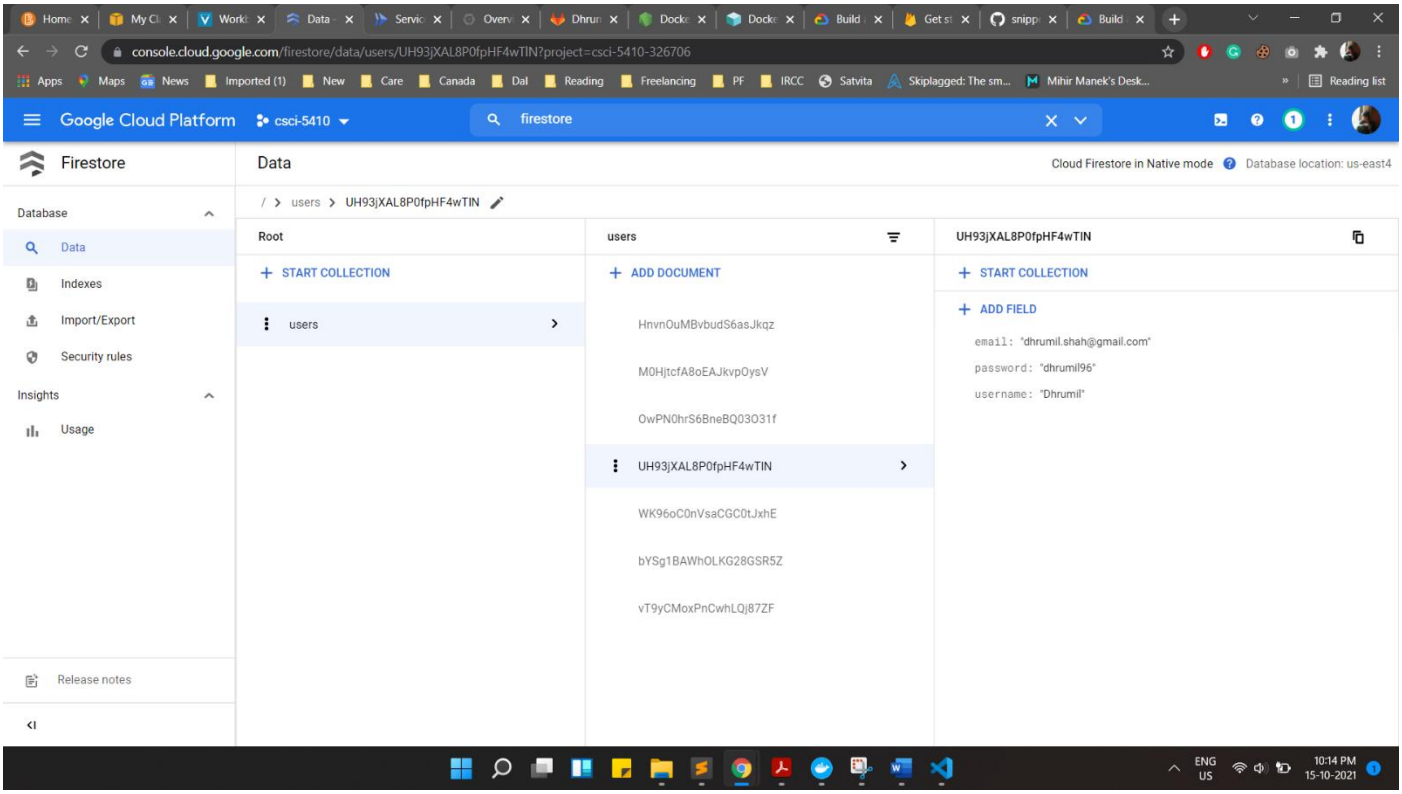

Figure 3: Firestore Filled Database

Figure 4: Application Home Page with Navbar

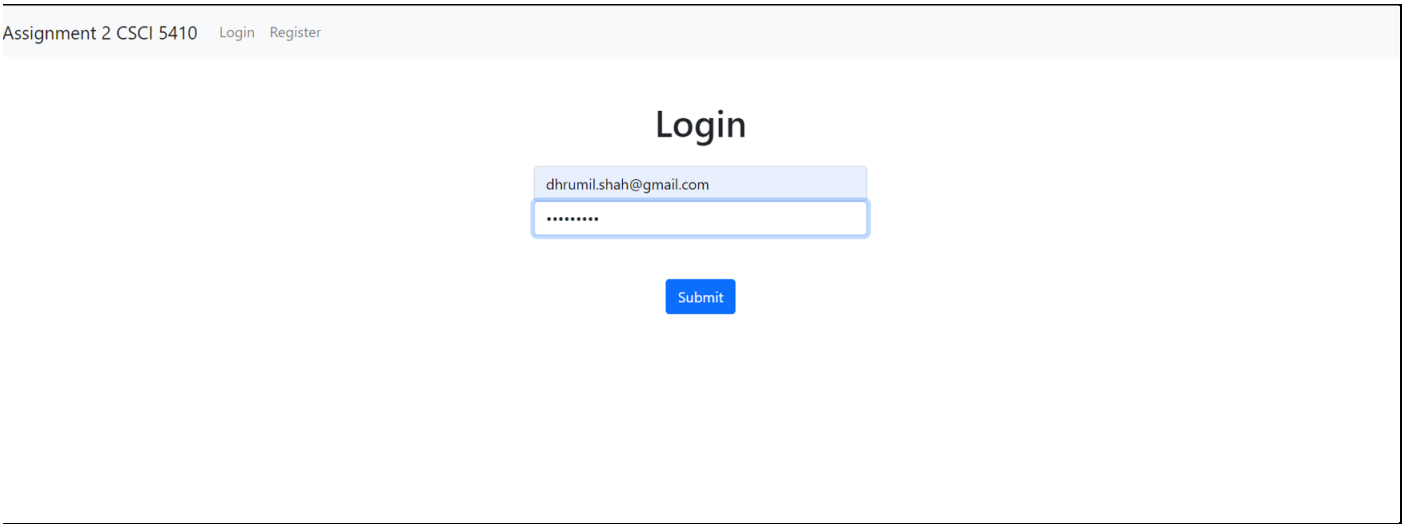

Figure 5: Application Login Page

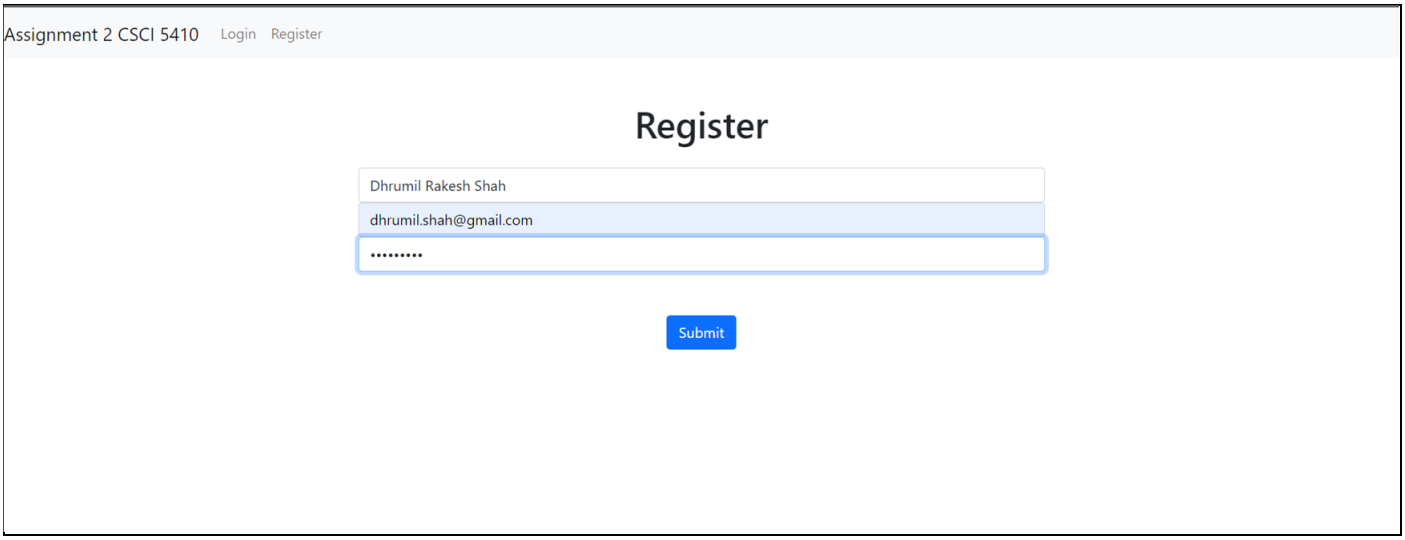

Figure 6: Application Register Page

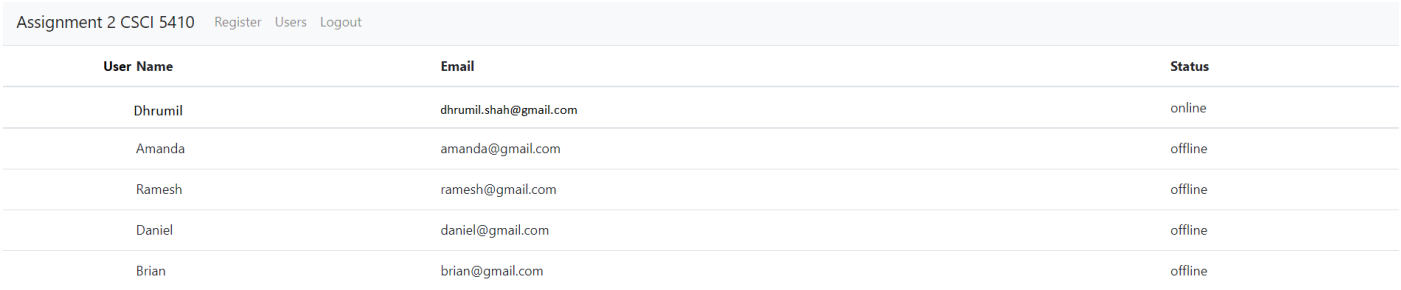

Figure 7: Application Users List Page

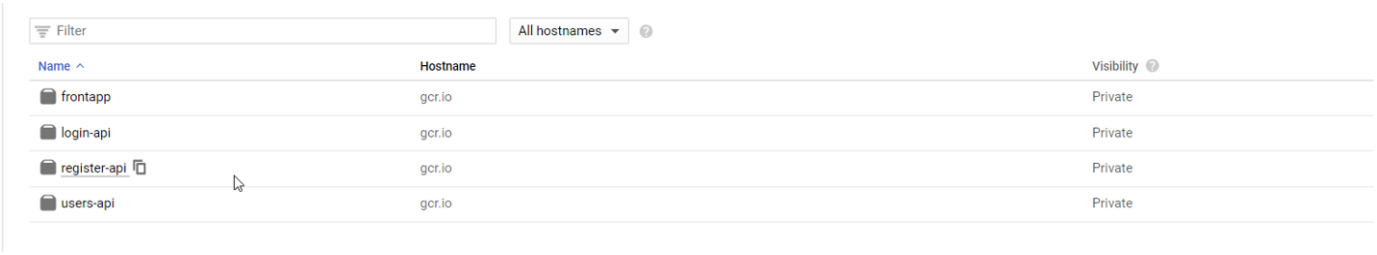

Figure 8: GCP Container Registry

Pseudo Code of the Application:

Login:

1. User opens the webpage and selects the Login option on the top Navbar.
2. The user then fils his/her data in the form and clicks on Submit button.
    a. The Login page is a part of Frontend.

3. The Login page or Frontend utilizes a service which invokes the Backend API.
4. The Backend API then makes use of the Firestore collection database provided by Google Cloud Platform (GCP).
    a. The Firestore collection database connection is established which verifies the user entered credentials with the credentials present in the database.
    b. If the credentials don't match then the Backend API responds with an error.
    c. Else the username of the user is returned.
5. If the Backend API returns the username then the login of the user is successful and the returned username is stored in the backend.
6. On successful login the state of the user in the database is updated.
7. The Frontend of the application verifies and changes itself if the user is logged in.

Logout:

1. We assume that the user is already logged in to the application.
2. The user selects the option to Logout from the application.
3. On clicking the logout link the API in the Backend is invoked which in turn changes the state of the user in the database.
4. On the other hand if the state is successfully modified by the API then the username from the local storage is removed as it is no longer needed as the session of that user has ended.

Register:

1. The user visits the webpage and then he/she selects the Register option.
    a. This then loads the Register page.
2. The user then fills his/her details in the form and then clicks on submit.
    a. On clicking submit the API is called by the frontend service.
    b. The backend then fetches the data from the request and invokes the Firestore collection database using the established connection.
    c. This in turn adds the details of the new user into the Firestore collection database.
3. On successful registration the user is redirected to the login page.

Users:

1. We assume that the user is already logged in to the application.
2. Then if he/she selects the Users Page on the webpage,
    a. Then the Frontend API has a table in bootstrap that gets populated by the data fetched from the Backend API.
3. The React application invokes the API of the backend which in turn fetches all the data of the existing users in the Firestore collection database.
4. The data is sent as a response back to the request being made.
5. Finally the frontend displays the data in the table format which can be viewed.

Summary:

To develop, build and deploy this application I have made use of the following technologies:

- React JS + Node JS + Express JS
- Google Cloud Platform
- Firestore Collection Database

- Docker
- Google Cloud Run

The basic operation that the application performs is lets new users register and existing users log in. The application also fetches the users details and displays in on the frontend. As listed above various technologies have been used in order to develop, build and deploy the application.

The frontend of the application is built using React JS which has a file that does the work of accessing the data using Node JS and REST API calls. So in simple words there are API calls in the backend of the applicatioln which are invoked by the frontend servies or more like functions. The Node JS APIs have business logic written inside them which helps them in communicating with the Firestore Collection Database in Google Cloud Platform (GCP). The application works in sessions by creating and maintaining them to update the states of the users. The session is maintained mainly by storing the user details in the browsers local storage when the user logs in to the application and if the user logs out then the user details are removed from the local storage.

Moving on to the backend of the application the technologies mainly used are the Node JS and Express JS along with APIs being used. Along with them the Docker and Google Cloud Run are being used for containerization and deployment of those containers. In the application I have created docker files for all the three major operations that are login, register and users list. Also using docker I created images for all the three APIs which I mentioned above. I have made use of the docker desktop application to create images and containers. On successful creation and testing of the docker images, I deployed all the images and backend APIs on Google Cloud Run by using the Google Cloud Registry. Once the images are successfully uploaded in the Google Cloud Registry then I can directly run them using Google Cloud Run. After I deployed all the backend on the Google Cloud, I tested all the APIs using the Postman application. After everything was working I tested the entire application and everything was working as expected.