
CSCI 5410 Serverless Data Processing

Assignment 2 Report

Prepared by:
Vikram Venkatapathi - B00936916

Master of Applied Computer Science (Summer'23)
Faculty of Computer Science
Dalhousie University

GitLab Repo link : https://git.cs.dal.ca/vikramv/csci5410-summer-23-b00936916/-/tree/A2?ref_type=heads

Contents

Table of Contents	2
I. PART A	1
1. Introduction	3
2. Summary	3
II. PART B	5
3. Procedure followed for the given experiment	7
3.1. Flowchart	7
3.2. Steps	8
3.2.1. Configure environment	8
3.2.2. Control and data flow of the deployments	8
4. Screenshots	10
4.1. Environment setup	10
4.1.1. Collections & Repositories	14
4.2. Demo	18
4.3. Test cases	32
4.4. My view on leveraging Google Cloud Run, GCR/Artifact Registry, and Docker Containers for Efficient Application Deployment	35
III. PART C	37
5. Bot RideRequest	39
5.1. Actions taken to perform the bot creation operation	39
5.2. Intents	39
5.2.1. TaxiRequest	39
5.2.2. SelfDriveRequest	41
5.3. Screenshots	43
5.4. Output	52

Part I.

PART A

Provide summary

1. Introduction

The given paper titled as *Performance Evaluation of Distributed Systems in Multiple Clouds using Docker Swarm* authored by "N. Naik", focuses on Docker Swarm-based Distributed System in Multiple Clouds[1].

This report attempts to review the given literature and then provides a summary.

2. Summary

The authors present a study on designing distributed systems in multiple clouds using Docker Swarm. They emphasize the benefits of multi-cloud infrastructure and discuss the challenges it entails. **Docker** is introduced as an **efficient platform for application development** through **containerization**. **Docker Swarm** [15] is highlighted as a **clustering tool** that addresses critical issues in provisioning, configuration management, load balancing, and migration. Overall, the paper provides insights into building robust distributed systems across multiple cloud environments using Docker Swarm.

The paper specifically addresses the challenge of designing distributed systems across multiple cloud environments. It highlights the **issues** related to **provisioning, configuration management, load balancing, and** migration in such setups. The authors propose Docker Swarm as a solution to these challenges, as it provides a clustering mechanism that enables efficient application development through containerization. By leveraging Docker Swarm, the paper presents a framework for building robust distributed systems that can seamlessly operate in multi-cloud infrastructures.

The paper describes several experiments and studies conducted to validate the proposed framework. The authors set up a test environment consisting of multiple cloud providers(**AWS, Azure, GCP, Digital Ocean & Softlayer**) and deployed a distributed application using Docker Swarm. They evaluated the **performance and scalability** of the system by **measuring response times, throughput, and resource utilization** under different workload conditions. Additionally, they conducted experiments to analyze the impact of network latency and node failures on the overall system performance. The **results** of these experiments demonstrated the **effectiveness** of the proposed framework in achieving efficient resource utilization, **fault tolerance**, and **seamless scalability** in multi-cloud environments.

The authors' analysis and findings demonstrated that their **framework improved resource utilization** in multi-cloud environments by distributing tasks effectively. The experiments showed scalability and fault tolerance, with the system adapting to workload changes and recovering from failures. **Network latency** was identified as an **influential factor**, underscoring the importance of efficient communication and task allocation. Overall, the framework proved advantageous for efficient resource management and fault tolerance in multi-cloud settings.

In conclusion, the authors presented a framework for efficient resource management in multi-cloud environments. Their approach effectively distributed tasks, improved resource utilization, and demonstrated scalability and fault tolerance. The experiments highlighted the significance of network latency and emphasized the need for efficient communication and task allocation. The findings underscored the framework's ability to adapt to workload changes and recover from failures. Overall, the study showcased the advantages of the proposed framework in enhancing resource management and fault tolerance in multi-cloud settings.

My suggestions on areas of improvement:

The paper could **discuss the limitations of its experimental setup**. For example, the authors only tested the system with a small number of nodes. But according to [16], Kubernetes is a good choice, when the scale, complexity, and management of the containers increase. Hence, It would be interesting to see how the system performs with a larger number of nodes.

The paper could **suggest potential areas for future exploration**. For example, the authors could explore how Docker Swarm could be used to build distributed systems that are more secure or more efficient.

The paper could **provide alternative perspectives on the findings**. For example, the authors could compare the performance of Docker Swarm to other container orchestration tools, such as Kubernetes[17], and K3s[18].

Part II.

PART B

Build, deploy, and run a Containerized Application using GCP

3. Procedure followed for the given experiment

3.1. Flowchart

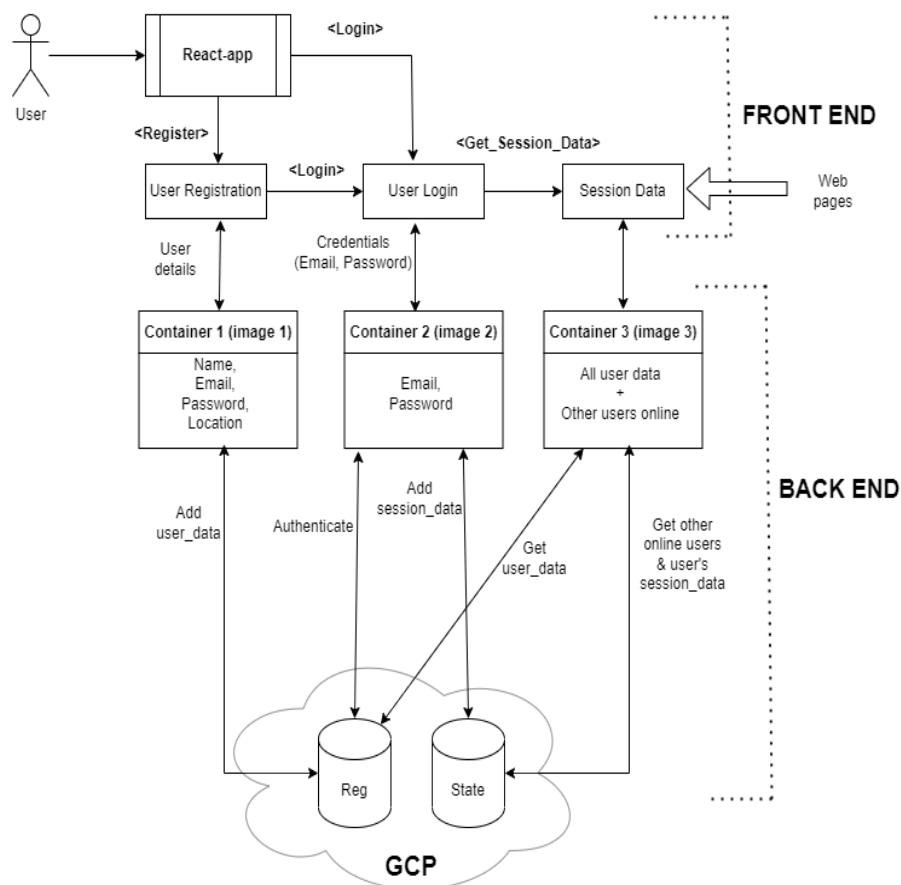


Figure 3.1.: *Flowchart describing the operations that I have performed.*

3.2. Steps

3.2.1. Configure environment

1. Create images for all the given code and front-end (i have created the front-end in React)

Note: Download the service account key from GCP IAM. In the docker file, copy the file to the container, and set it as an Environment variable as follows

```
1 COPY assignment-2-390620-60700558380c.json /app/assignment  
     -2-390620-60700558380c.json  
2  
3 ENV GOOGLE_APPLICATION_CREDENTIALS /app/assignment-2-390620-60700558380c.  
     json
```

Listing 3.1: Dockerfile Code snippet

This will allow the containers to access the collection in Firestore.

2. Create separate repositories for all containers (3 containers: back-end; 1 container: front-end) in **Artifact Registry**.
3. Push the images to the respective repositories
4. Using the images, create services for all the containers, in **Cloud run** (with proper **PORT** and other configuration)

3.2.2. Control and data flow of the deployments

1. User performs registration

2. Send a POST request to Container-1 (API: /A2/register), with all *user_data*.

3. Registration by **Container-1**:

- a) Create record in collection "Reg" with *user_data* (shown below)

```
1 {  
2     'Name': <name>,  
3     'Password': <password>,  
4     'Email': <email>,  
5     'Location': <location>  
6 }
```

- b) Return response (success/failure).

4. Front-end: re-direct to **login page** after successful registration (API: /A2/login).

5. User enters credentials.

6. Authentication by **Container-2**

- a) Fetch the respective document from the collection "Reg"

- b) Validate credentials
 - c) Return Success/Failure
7. After successful authentication,
- a) If the user is logging in for the 1st time, create a *session_data* document(shown below) in the collection "state", with the user_email as the document id.
- ```

1 {
2 'Online': True,
3 'Offline': False,
4 'Timestamp': <current_system_time>
5 }
```
- b) Else, just update the fields (toggle Online & Offline, update the current timestamp of system)
8. Redirect to **session\_data page** (API: /A2/sessionData/<email>)
9. Session data from **Container-3**
- a) Fetch *user\_data* from the collection "Reg", *session\_data* from the collection "state", by matching with the <email>
  - b) Display all the user details
  - c) To display the other users, who are online, filter out all documents in the collection "state" where the key "**Online**" is **True**. (API: /A2/sessionData/usersOnline/<email >)
  - d) Display the filtered document ids (except the current user who has logged in).
10. Logout by Container-3:
- a) On clicking logout, **end the current session**. (API: /A2/sessionData/<email >/logout)
  - b) Update the *session\_data* in the collection "state" (toggle Online & Offline fields)
  - c) Re-direct to the Login page.

# 4. Screenshots

## 4.1. Environment setup

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the command `docker build -t northamerica-north east1-docker.pkg.dev/assignment-2-390620/a2-cntr1/cntr1` followed by its execution log:

```
[+] Building 10.6s (11/11) FINISHED
-> [internal] Load build definition from dockerfile
-> [internal] Transfer file: 475B
-> [internal] Load dockerignore
-> [internal] Transfer context: 2B
-> [internal] Load metadata for docker.io/library/python:latest
-> [auth] library/python:pull token for registry-1.docker.io
-> [internal] Load build context
-> [internal] Load build context
-> [internal] Transfer context: 4.75kB
-> CACHED [2/5] WORKDIR /app
-> [3/5] COPY . .
-> [4/5] RUN python assignment-2-390620-60700558380c.json /app/assignment-2-390620-60700558380c.json
-> [5/5] RUN pip install --no-cache-dir -r requirements.txt
-> exporting to image
-> exporting layers
-> writing manifest sha256:fc56bf7a2772893d47398edfd183aa9242dc1f92a1a3f882754d2aa220bc3
-> + needs to set the env variable DOCKER_CLI_EXPERIMENTAL=enabled
PS C:\Users\vikra\Desktop\WAC5\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice1>
```

Figure 4.1.: *Build container 1*

```

File Edit Selection View Go Run Terminal Help LoginForm.js - Part B - Visual Studio Code
EXPLORER JS LoginForm.js JS RegistrationForm.js JS SessionData.js
PART B
 > .idea
 > public
 > src
 > Login
 # LoginForm.css
 JS LoginForm.js
 > Registration
 # RegistrationForm.css
 JS RegistrationForm.js
 > SessionData
 # SessionData.css
 JS SessionData.js
 # App.css
 JS App.js
 JS App.test.js
 JS AuthContext.js
 # index.css
 JS index.js
 logo.svg
 JS PrivateRoute.js
 JS reportWebVitals.js
 JS setupTests.js
 .gitignore
 Dockerfile
 package-lock.json
 package.json
 README.md
 > microservice1
 assignment-2-3906...
 > OUTLINE
 > TIMELINE
 0 0 GitLens Pro (trial)

TERMINAL
powershell - microservice1 + v □ □ ... ×
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice> docker push northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr1/cntr1:latest
sebc0cf6495b: Pushed
bef4f4038e05: Pushed
21be576476d1: Pushed
4599a0a0a04d: Pushed
d0e41de1f543: Pushed
b860f3828204: Pushed
5ef4dc47ff52: Pushed
037f20f86912: Pushed
a597a0a0a04d: Pushed
964529c819fb: Pushed
2f98fa2985b1: Pushed
332b199f36eb: Pushed
latest: digest: sha256:7f9fc66e6ad73094d5345c24f852df14652bdca8024ee8076330699712973ef size: 2842
○ PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice>

```

Figure 4.2.: *Push container 1*

```

File Edit Selection View Go Run Terminal Help LoginForm.js - Part B - Visual Studio Code
EXPLORER JS LoginForm.js JS RegistrationForm.js JS SessionData.js
PART B
 > .idea
 > public
 > src
 > Login
 # LoginForm.css
 JS LoginForm.js
 > Registration
 # RegistrationForm.css
 JS RegistrationForm.js
 > SessionData
 # SessionData.css
 JS SessionData.js
 # App.css
 JS App.js
 JS App.test.js
 JS AuthContext.js
 # index.css
 JS index.js
 logo.svg
 JS PrivateRoute.js
 JS reportWebVitals.js
 JS setupTests.js
 .gitignore
 Dockerfile
 package-lock.json
 package.json
 README.md
 > microservice1
 assignment-2-3906...
 > OUTLINE
 > TIMELINE
 0 0 GitLens Pro (trial)

TERMINAL
powershell - microservice2 + v □ □ ... ×
PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice> docker build -t northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr2/cntr2 .
[+] Building 0.3s (10/10) FINISHED
--> [internal] load .dockerignore
--> [internal] load Dockerfile
--> [internal] load build definition from Dockerfile
--> [internal] transfer Dockerfile: 658B
--> [internal] load metadata for docker.io/library/python:latest
--> [internal] load build context
--> [internal] load build context
--> [internal] transfer context: 1668B
--> CACHED [2/5] WORKDIR /app
--> CACHED [3/5] COPY requirements.txt .
--> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
--> CACHED [5/5] RUN pip install --no-cache-dir -r requirements.txt
--> exporting to image
--> exporting layers
--> writing manifest to /tmp/docker-manifests/7f34069907fafb3091399407f7f5604433c9a67dc2083945ab222700
--> writing image sha256:6180707f7f34069907fafb3091399407f7f5604433c9a67dc2083945ab222700
○ PS C:\Users\vikra\Desktop\MACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice>

```

Figure 4.3.: *Build container 2*

```

PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice3> docker push northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr2/cntr2:latest
f7c5d452a9b3: Pushed
51e882ca6699: Pushed
46c68c960bfd: Pushed
The push refers to repository [northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr2/cntr2]
f7c5d452a9b3: Pushed
806ff3828204: Mounted from assignment-2-390620/a2-cntr1/cntr1
5ef4dc47ef52: Mounted from assignment-2-390620/a2-cntr1/cntr1
037f26fb812: Mounted from assignment-2-390620/a2-cntr1/cntr1
a5959a08090: Mounted from assignment-2-390620/a2-cntr1/cntr1
964529c8104b: Mounted from assignment-2-390620/a2-cntr1/cntr1
2f98fa2985b1: Mounted from assignment-2-390620/a2-cntr1/cntr1
332b199f36eb: Mounted from assignment-2-390620/a2-cntr1/cntr1
latest: digest: sha256:48847af7c93b7810edbf7794b53f8929c4e9915e6377fd2d2c1c439chde69 size: 2842
○ PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice3>

```

Figure 4.4.: *Push container 2*

```

PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice3> docker build -t northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr3/cntr3 .
[+] Building 9.8s (11/11) FINISHED
 ➜ [1/1] FROM python:3.8-slim@sha256:980led4c2ff029f4d4135d2e7d771f520062ea5237130f9bc379f6e007a256400
 ➜ [2/5] WORKDIR /app
 ➜ [3/5] COPY assignment-2-390620-60700558380c.json /app/assignment-2-390620-60700558380c.json
 ➜ [4/5] RUN pip install --no-cache-dir --requirement requirements.txt
 ➜ [5/5] RUN pip install --no-cache-dir --requirement requirements.txt
 ➜ [internal] load metadata for docker.io/library/python:3.8-slim
 ➜ [internal] load manifest for registry 1.docker.io
 ➜ [internal] load build context
 ➜ [internal] transfer context 6.3kB
 ➜ [internal] COPY assignment-2-390620-60700558380c.json /app/assignment-2-390620-60700558380c.json
 ➜ [internal] RUN pip install --no-cache-dir --requirement requirements.txt
 ➜ [internal] export to image
 ➜ [internal] write manifest sha256:980led4c2ff029f4d4135d2e7d771f520062ea5237130f9bc379f6e007a256400
 ➜ [internal] naming to northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr3/cntr3
○ PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B\microservice3>

```

Figure 4.5.: *Build container 3*

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with project files like `index.js`, `App.js`, `AuthContext.js`, `Dockerfile`, and `README.md`. The main area has three tabs open: `LoginForm.js`, `RegistrationForm.js`, and `SessionData.js`. The `TERMINAL` tab is active, displaying a PowerShell session with the command `docker push northamerica-northeast1-docker.pkg.dev/assignment-2-390620/a2-cntr3/cntr3:latest`. The output shows several layers being pushed, such as `60e41de15f43: Mounted from assignment-2-390620/a2-cntr2/cntr2` and `980ded4c2cff: Mounted from assignment-2-390620/a2-cntr2/cntr2`. The status bar at the bottom indicates the file is saved in GitLens.

Figure 4.6.: *Push container 3*

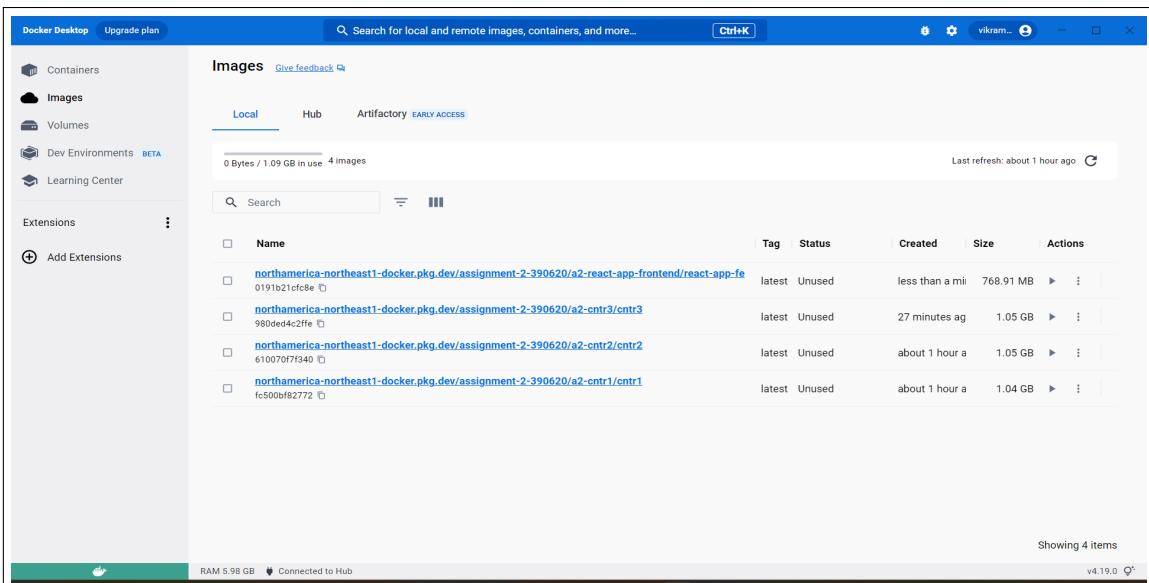


Figure 4.7.: *Built images listed in Docker Desktop*

#### 4.1.1. Collections & Repositories

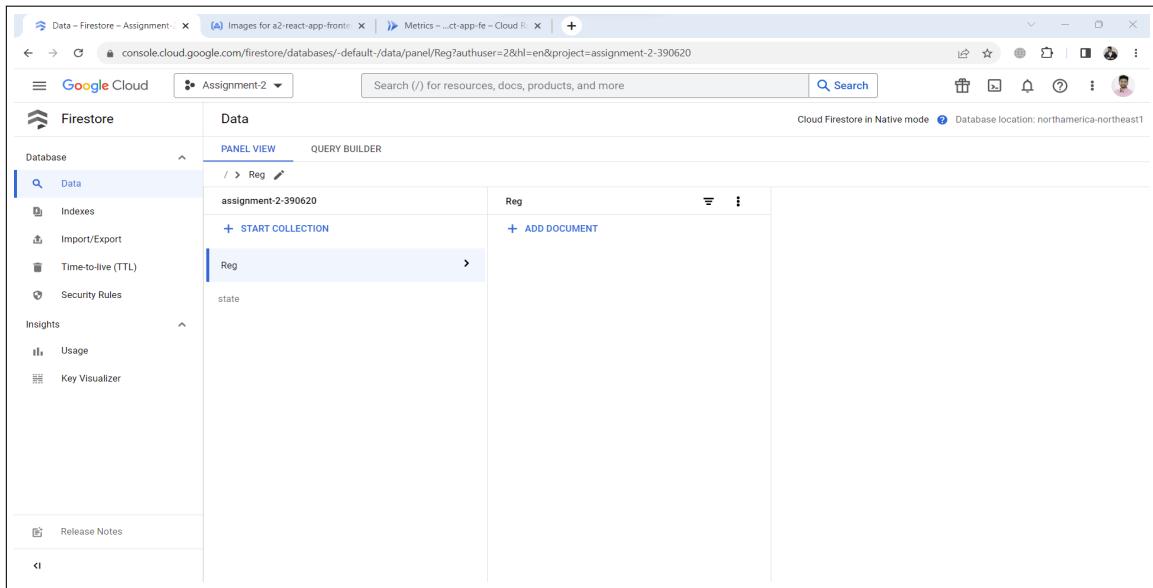


Figure 4.8.: *Empty collection "Reg"*

The screenshot shows the Google Cloud Firestore Data panel. On the left, the sidebar includes Database, Indexes, Import/Export, Time-to-live (TTL), Security Rules, Insights (Usage, Key Visualizer), and Release Notes. The main area displays a collection named 'state' under the database 'assignment-2-390620'. A sub-collection named 'Reg' is shown, which contains a document named 'state'. The interface includes a 'PANEL VIEW' tab, a 'QUERY BUILDER' section, and a search bar at the top.

Figure 4.9.: *Empty collection "state"*

The screenshot shows the Google Cloud Artifact Registry interface. The sidebar includes Repositories and Settings. The main area displays a table of repositories. The table has columns for Name, Format, Type, Location, Description, Labels, and Version. The data in the table is as follows:

| Name                  | Format | Type     | Location                           | Description | Labels | Version |
|-----------------------|--------|----------|------------------------------------|-------------|--------|---------|
| a2-cntr1              | Docker | Standard | northamerica-northeast1 (Montréal) |             |        |         |
| a2-cntr2              | Docker | Standard | northamerica-northeast1 (Montréal) |             |        |         |
| a2-cntr3              | Docker | Standard | northamerica-northeast1 (Montréal) |             |        |         |
| a2-react-app-frontend | Docker | Standard | northamerica-northeast1 (Montréal) |             |        |         |

Figure 4.10.: *Artifact registry repositories*

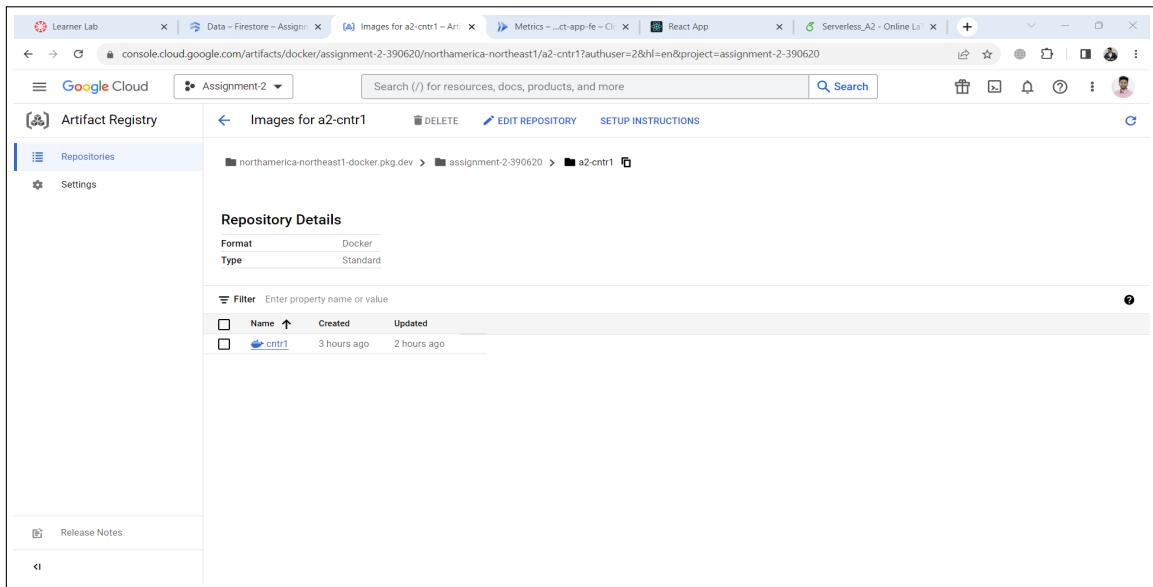


Figure 4.11.: *Artifact registry repository - container 1*

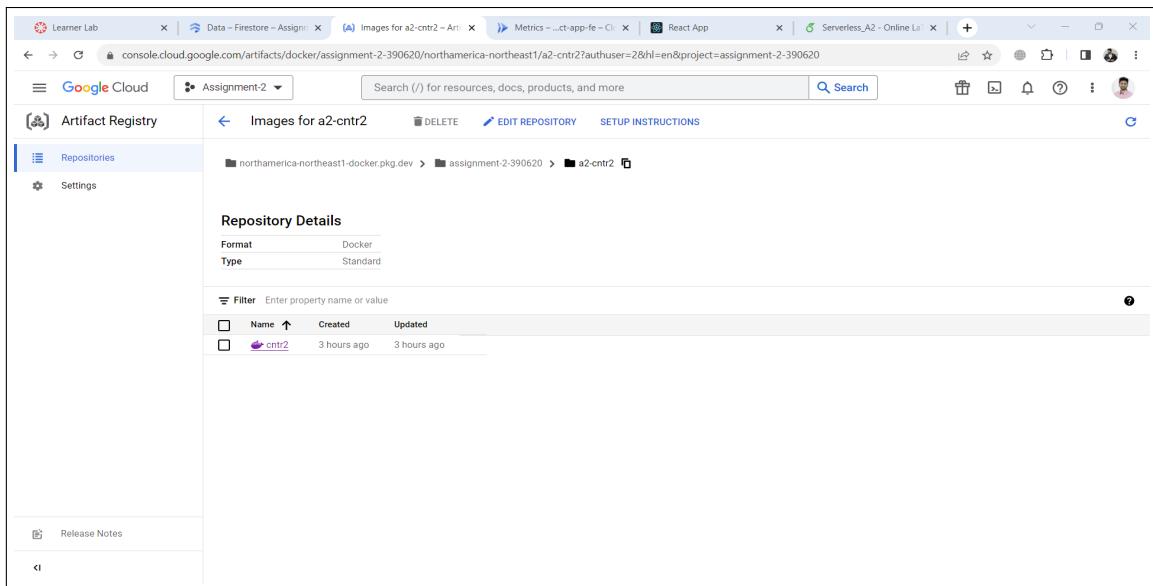


Figure 4.12.: *Artifact registry repository - container 2*

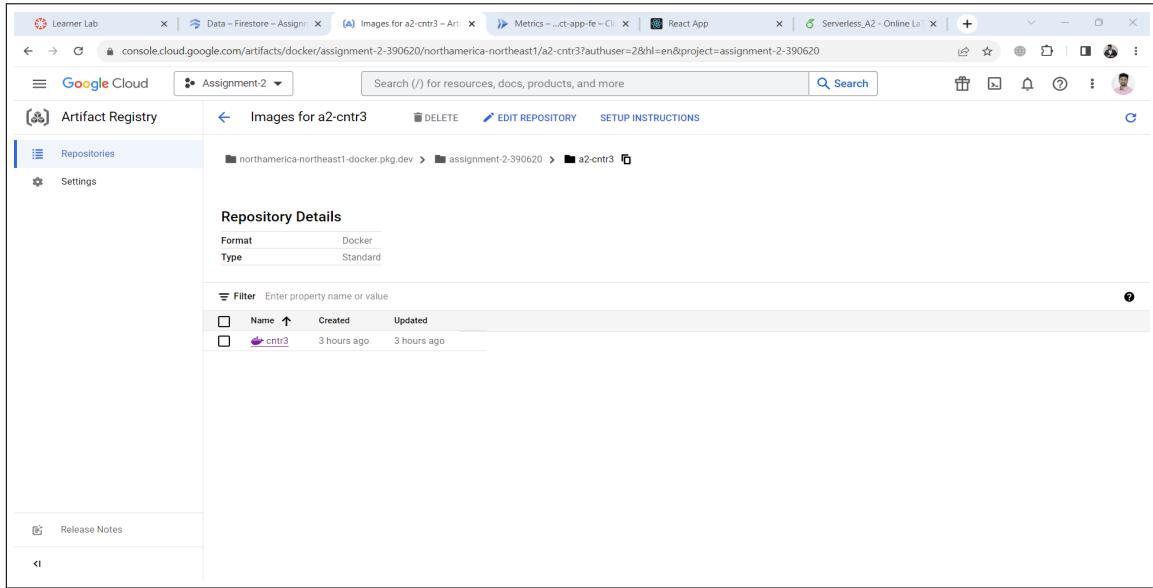


Figure 4.13.: *Artifact registry repository - container 3*

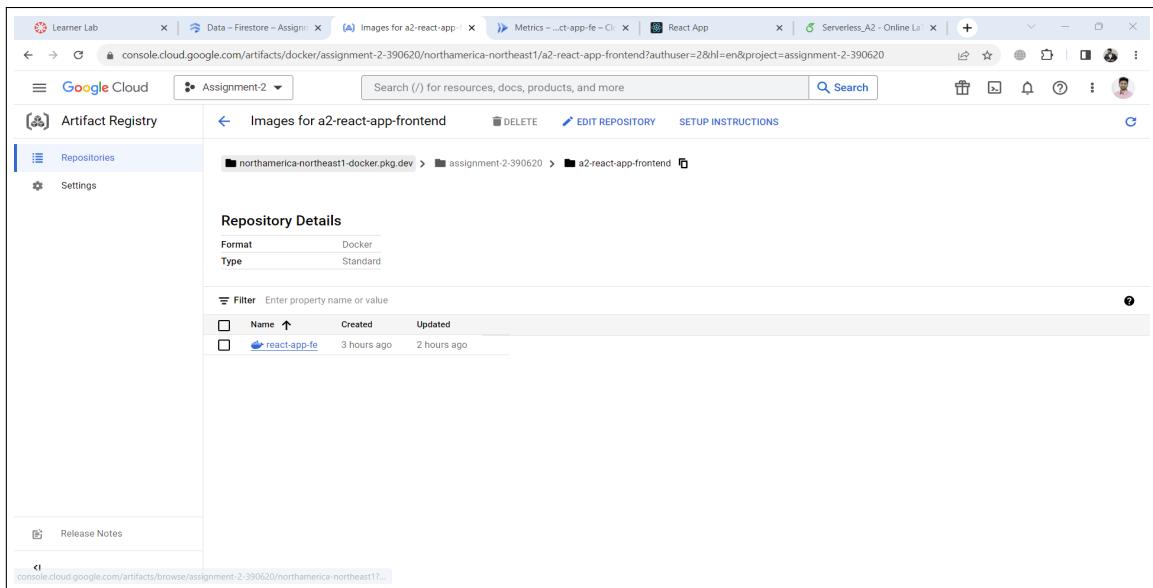


Figure 4.14.: *Artifact registry repository - react-app frontend container*

The screenshot shows the Google Cloud Platform Services - Cloud Run interface. The top navigation bar includes tabs for Data - Firestore - Assignment, Repositories - Artifact Registry, Services - Cloud Run - Assignment, and React App. The main header has a search bar and various navigation icons. Below the header, there are tabs for Cloud Run, Services, Create Service, Create Job, Manage Custom Domains, Copy, Delete, Tags, Release Notes, and Hide Info Panel. The Services tab is selected, and the sub-tab SERVICES is also selected. A table lists four services: cntr1, cntr2, cntr3, and react-app-fe. Each service row includes columns for Name, Req/sec, Region, Authentication, Ingress, Recommendation, Last deployed, and Deployed by. The 'react-app-fe' service was deployed 1 hour ago. To the right of the table, there are sections for PERMISSIONS and LABELS, both of which are currently empty. A message box indicates "Please select at least one resource.".

| Name         | Req/sec | Region                  | Authentication        | Ingress | Recommendation | Last deployed | Deployed by                  |
|--------------|---------|-------------------------|-----------------------|---------|----------------|---------------|------------------------------|
| cntr1        | 0       | northamerica-northeast1 | Allow unauthenticated | All     | SECURITY       | 2 hours ago   | vikramvenkatapathi@gmail.com |
| cntr2        | 0       | northamerica-northeast1 | Allow unauthenticated | All     | SECURITY       | 2 hours ago   | vikramvenkatapathi@gmail.com |
| cntr3        | 0       | northamerica-northeast1 | Allow unauthenticated | All     | SECURITY       | 2 hours ago   | vikramvenkatapathi@gmail.com |
| react-app-fe | 0       | northamerica-northeast1 | Allow unauthenticated | All     |                | 1 hour ago    | vikramvenkatapathi@gmail.com |

Figure 4.15.: *Services created in Cloud Run*

## 4.2. Demo

**NOTE:** In some of the screenshots, the URLs displayed on the front end may vary due to the process of deleting and recreating services on Cloud Run. These changes were made during the documentation process, which resulted in different URLs being assigned to the deployed containers. Please note that despite the variations in the URLs shown in the screenshots, the overall functionality and deployment process remains consistent.

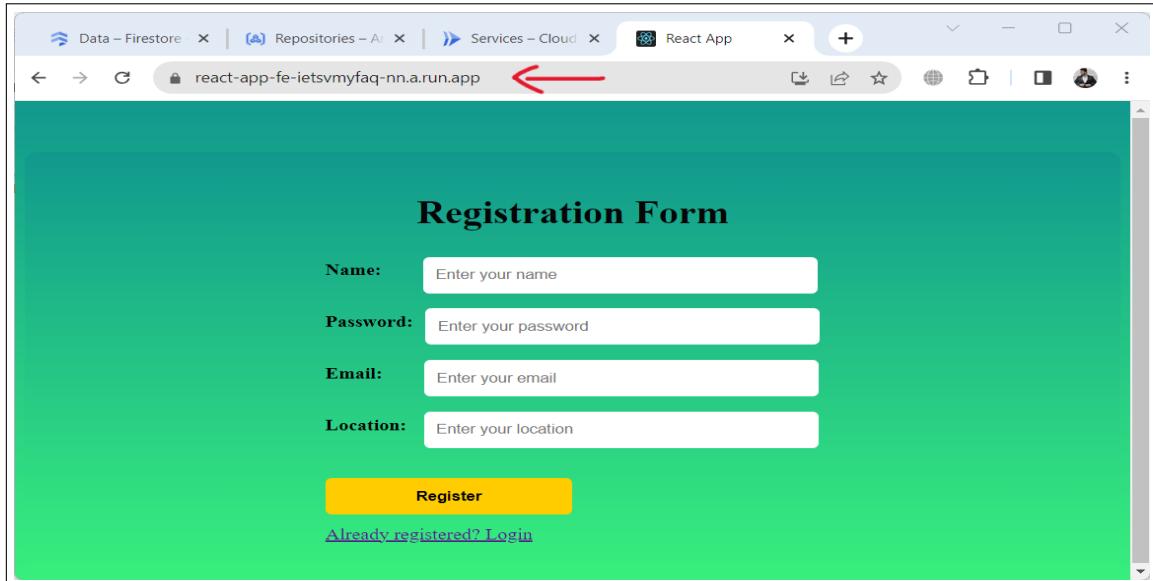


Figure 4.16.: *Registration page*

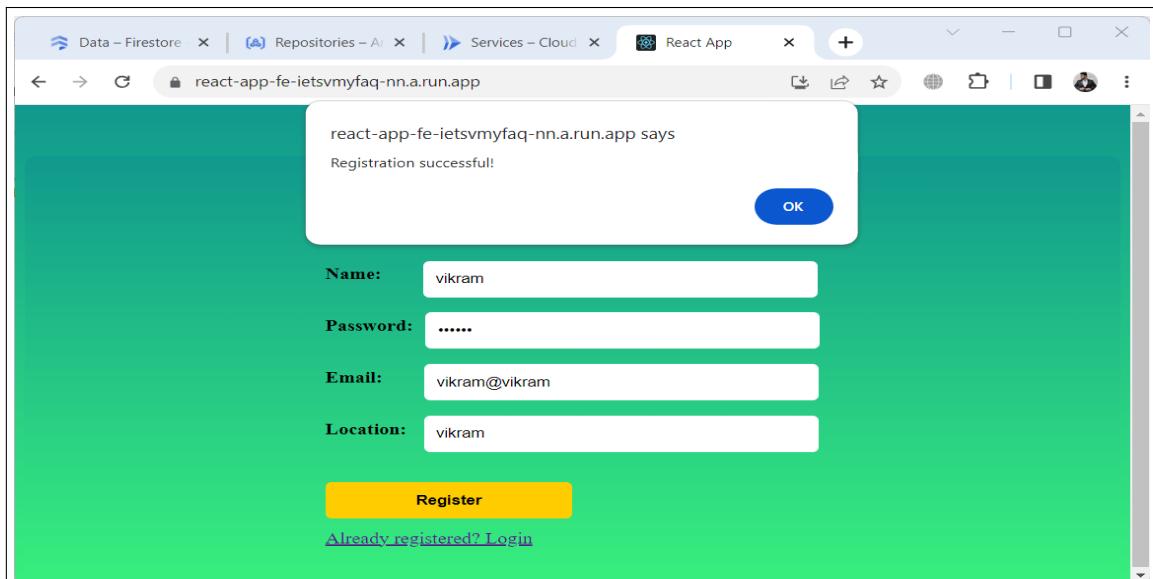


Figure 4.17.: *User - Vikram: Registration success*

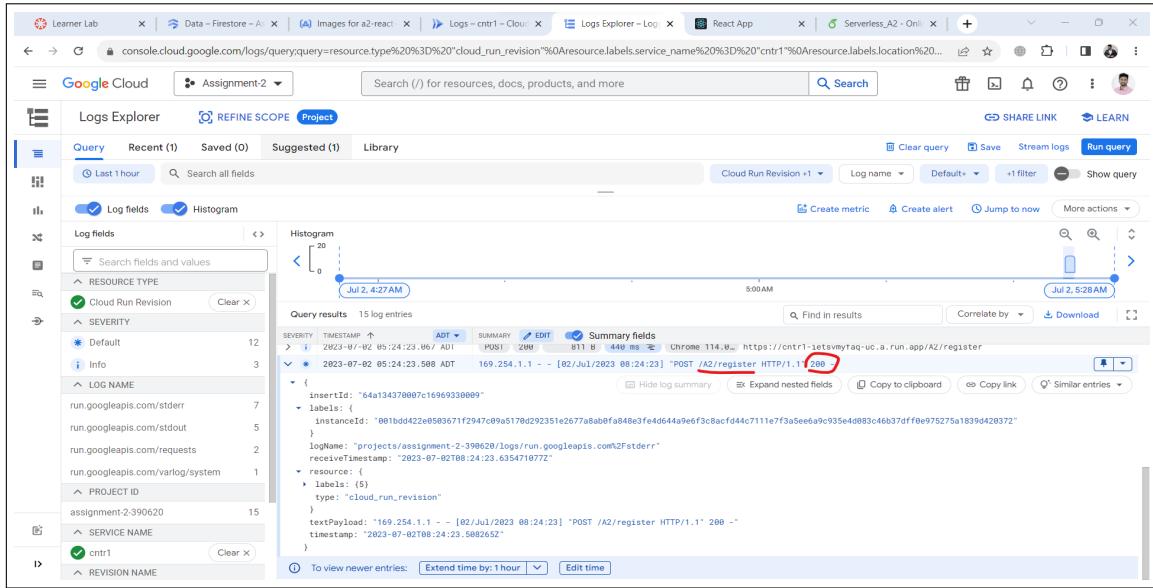


Figure 4.18.: *User - Vikram: Registration Logs*

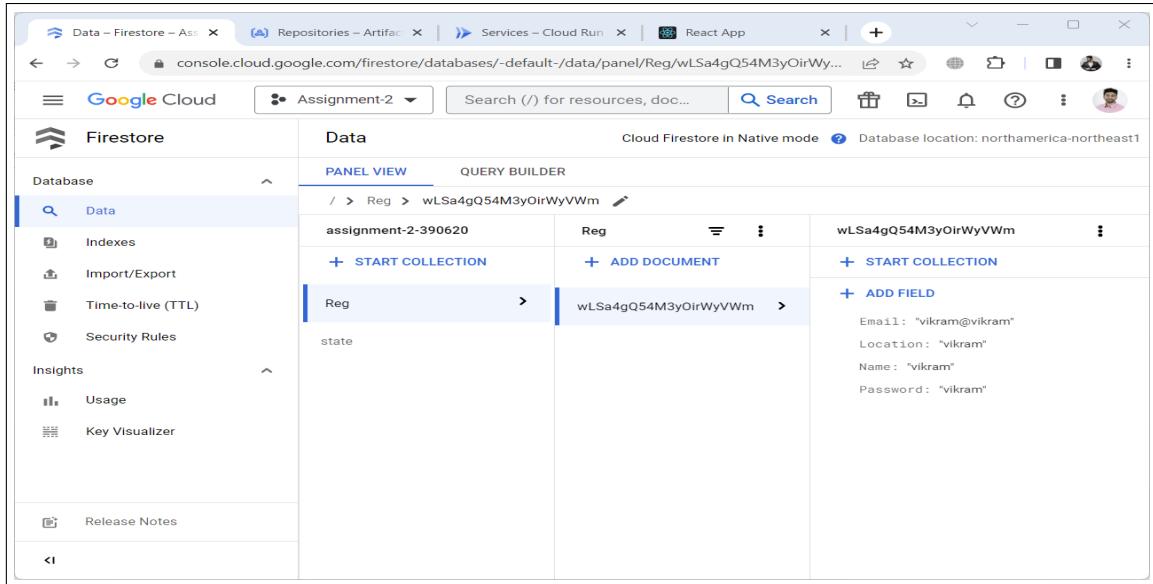


Figure 4.19.: *User - Vikram: Document created in "Reg"*

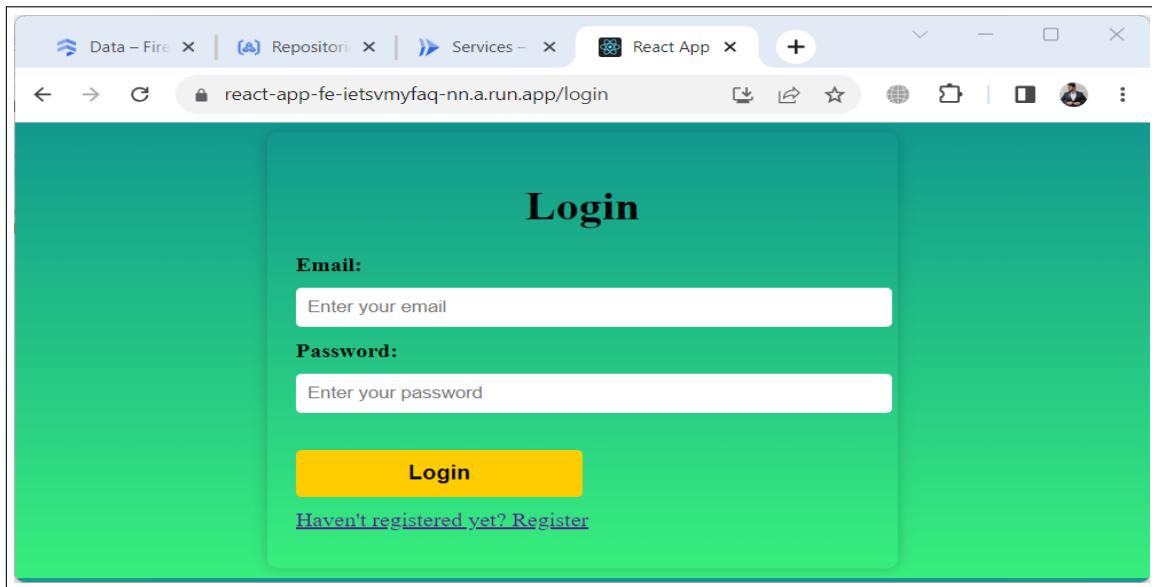


Figure 4.20.: *Login page*

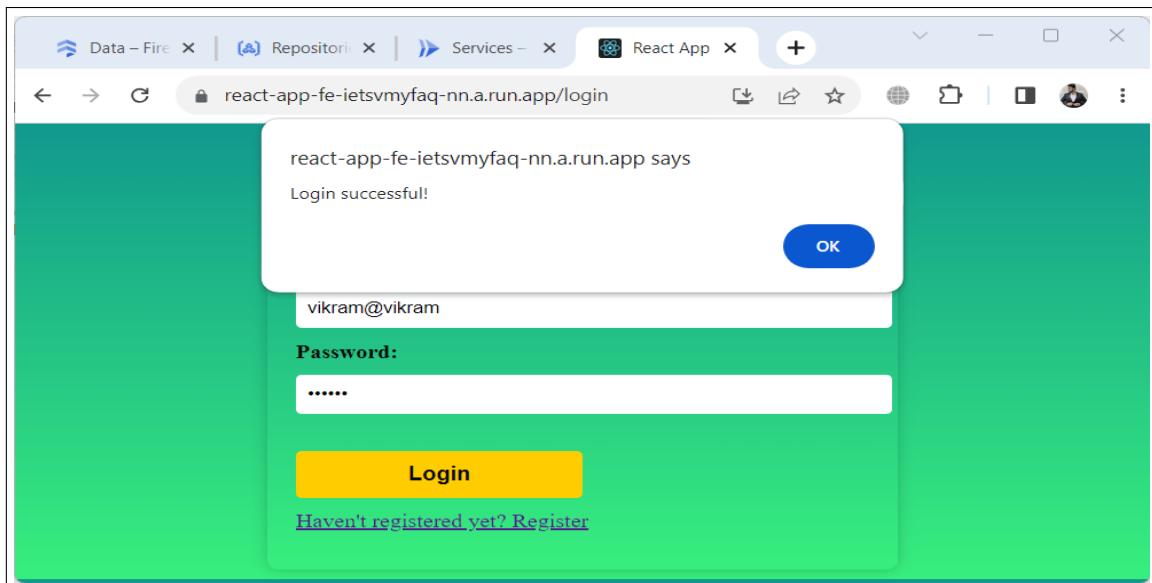


Figure 4.21.: *User - Vikram: Login Success*

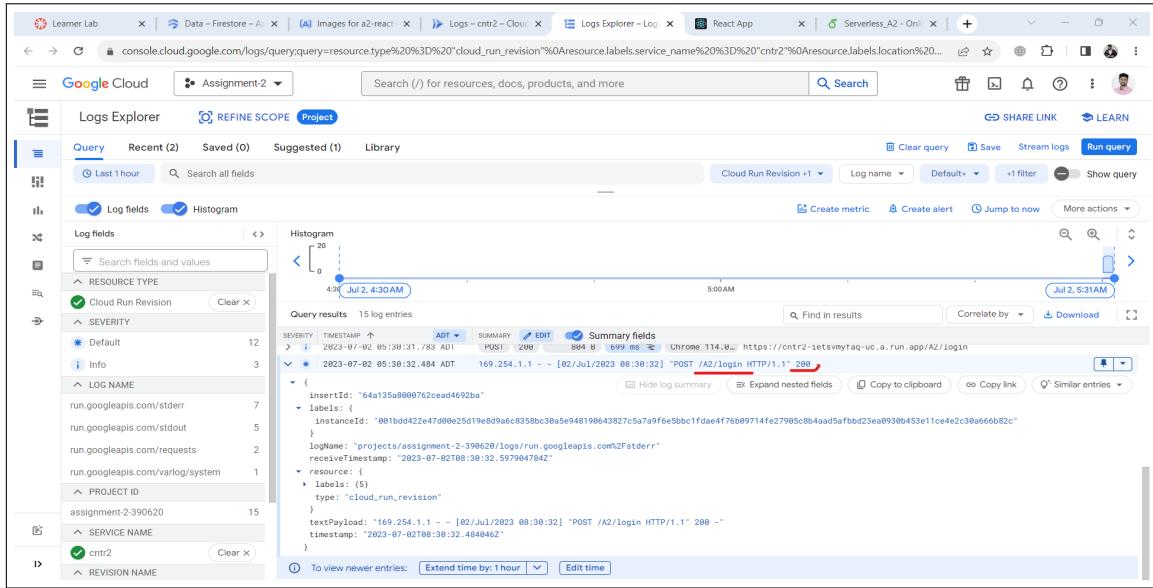


Figure 4.22.: *User - Vikram: Login logs*

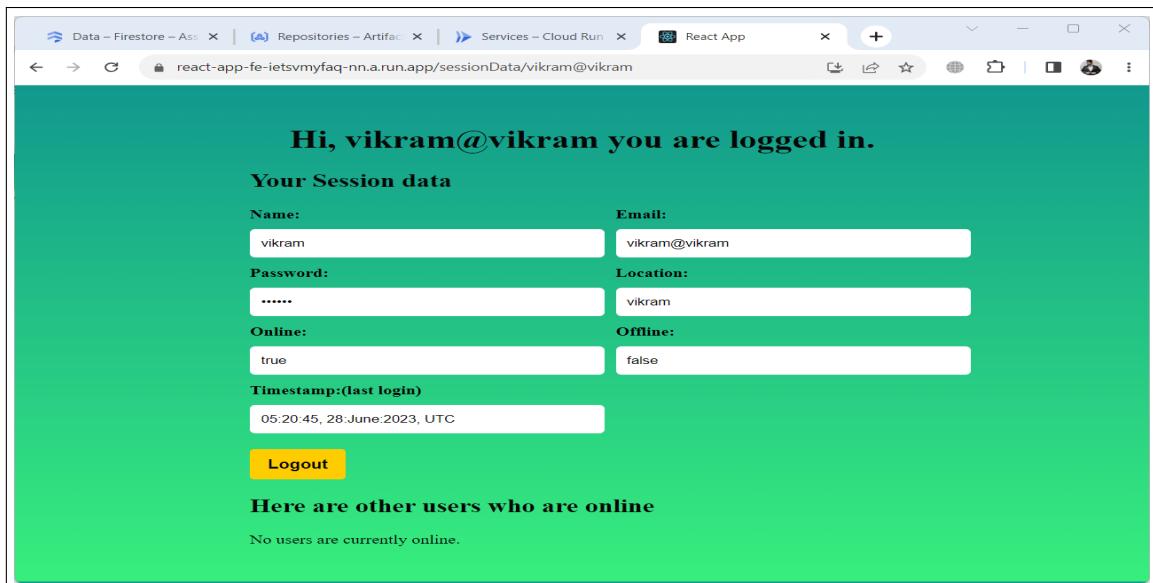


Figure 4.23.: *User - Vikram: Session data page (no other users online)*

The screenshot shows the Google Cloud Platform interface for a Cloud Run service named 'cntr3'. The 'Logs' tab is selected. The log entries are filtered by severity (Default) and timestamp. One specific log entry is highlighted with a red box:

```

SEVERITY TIMESTAMP SUMMARY
> 2023-07-02 05:30:37.352 ADT /usr/local/lib/python3.11/site-packages/google/cloud/firestore_v1/base_collection.py:290: UserWarning: Detected filter using positional arguments. Prefer using the 'filter' keyword.
> 2023-07-02 05:30:37.352 ADT return query.where(field_path, op_string, value)
> 2023-07-02 05:30:37.740 ADT 169.254.1.1 - - [02/Jul/2023 08:30:37] "GET /A/sessionData/usersOnline/vikram@vikram HTTP/1.1" 200 -
> 2023-07-02 05:30:37.948 ADT 169.254.1.1 - - [02/Jul/2023 08:30:37] "GET /A/sessionData/vikram@vikram HTTP/1.1" 200 -
 {
 insertId: "64a135ad080e0779df23854f"
 labels: {
 instanceId: "001bdd422e1822c4d1568d7054807e70ff6e15alc108852b28a752ac23df264c0a39f72e7ff1a464d4920dca36f62aaaf3f55f5676f18501692d39bc29fed18e94"
 }
 logName: "projects/assignment-2-398629/logs/run.googleapis.com%2Fstderr"
 receiveTimestamp: "2023-07-02T08:38:25Z999166Z"
 resource: {
 labels: {
 type: "cloud_run_revision"
 }
 textPayload: "169.254.1.1 - - [02/Jul/2023 08:30:37] "GET /A/sessionData/vikram@vikram HTTP/1.1" 200 -"
 timestamp: "2023-07-02T08:38:37.948115Z"
 }
 }

```

No newer entries found matching current filter.

Figure 4.24.: *User - Vikram: session logs*

The screenshot shows the Google Cloud Platform interface for a Firestore database. The 'Data' panel is open, showing a collection named 'state' under the path '/ > state > vikram@vikram'. A single document named 'vikram@vikram' is displayed with the following data:

| Field | Type   | Value                                                                                                     |
|-------|--------|-----------------------------------------------------------------------------------------------------------|
| state | Object | vikram@vikram                                                                                             |
| state | Object | Reg<br>state<br>+ ADD FIELD<br>Offline: false<br>Online: true<br>Timestamp: "05:20:45, 28:June:2023, UTC" |

Figure 4.25.: *User - Vikram: Session data document in "state"*

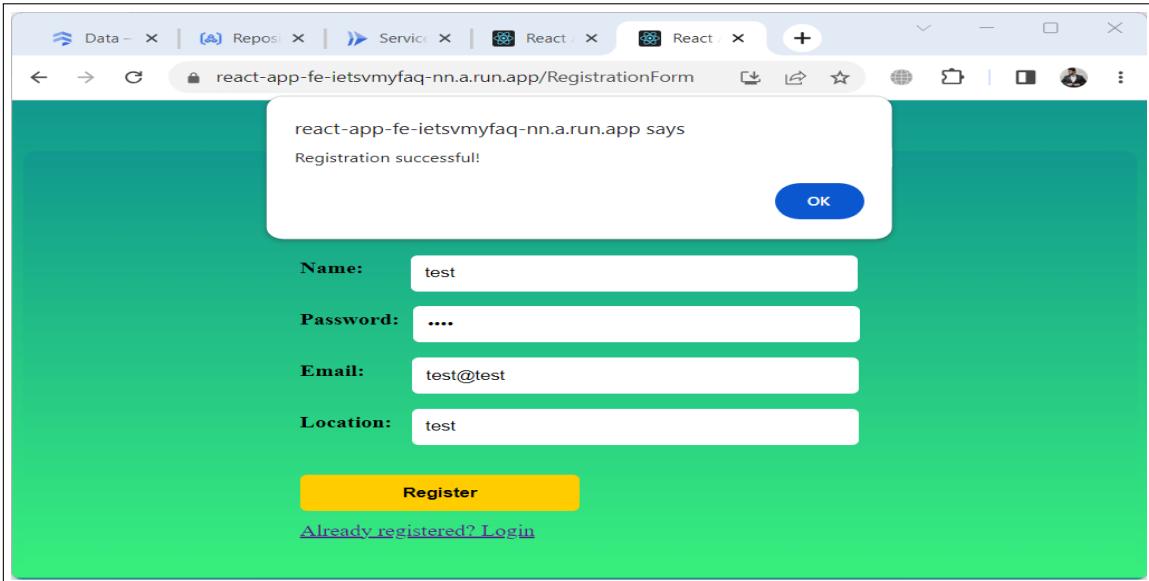


Figure 4.26.: *User - test: Registration success*

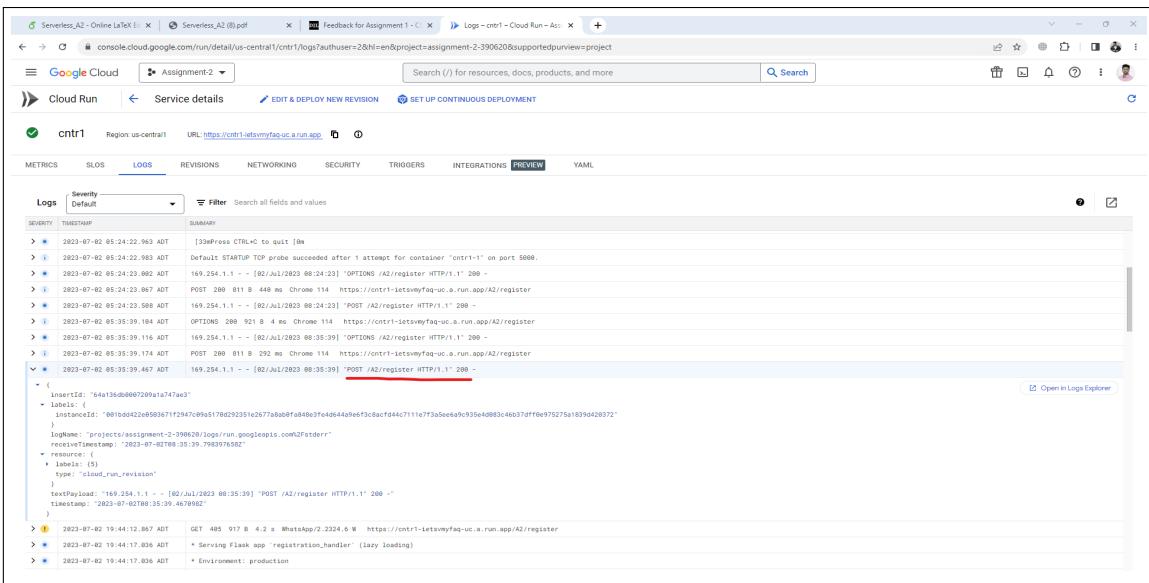


Figure 4.27.: *User - test: Registration success logs*

The screenshot shows the Google Cloud Firestore interface. On the left, there's a sidebar with 'Database' and 'Indexes' sections. The main area is titled 'Data' and shows a 'PANEL VIEW' of a document structure. The path is / > Reg > ytZszPgsEj4NjbGUeanm. The document 'Reg' has a field 'state' with value 'ytZszPgsEj4NjbGUeanm'. This document also contains four fields: 'Email' (value 'test@test'), 'Location' (value 'test'), 'Name' (value 'test'), and 'Password' (value 'test').

Figure 4.28.: *User - test: document created in "Reg"*

The screenshot shows a web browser window with the URL 'react-app-fe-ietsvmyfaq-nn.a.run.app/login'. A modal dialog box is open, displaying the message 'react-app-fe-ietsvmyfaq-nn.a.run.app says' followed by 'Login successful!' and an 'OK' button. Below the modal, there is a login form. It has two input fields: one for 'Email' containing 'test@test' and another for 'Password' containing '....'. There is a yellow 'Login' button. At the bottom of the form, there is a link 'Haven't registered yet? Register'.

Figure 4.29.: *User - test: Login success*

The screenshot shows the Google Cloud Logs interface for the 'cntr2' service. The 'Logs' tab is selected, displaying log entries from July 2, 2023. One entry highlights a successful login attempt:

```

2023-07-02 05:35:46.558 ADT OPTIONS 200 921 B 3 ms Chrome 114 https://cntr2-letsmyfaq-uc.a.run.app/A2/login
2023-07-02 05:35:46.569 ADT 169.254.1.1 - [02/Jul/2023 08:35:46] "OPTIONS /A2/login HTTP/1.1" 200 -
2023-07-02 05:35:46.626 ADT POST 200 884 B 494 ms Chrome 114 https://cntr2-letsmyfaq-uc.a.run.app/A2/login
2023-07-02 05:35:47.121 ADT 169.254.1.1 - [02/Jul/2023 08:35:47] "POST /A2/login HTTP/1.1" 200 -

```

Another entry shows a Python request to the login endpoint:

```

2023-07-03 00:56:08.649 ADT POST 200 765 B 4.1 s python-requests/2.28.2 https://cntr2-letsmyfaq-uc.a.run.app/A2/login

```

Figure 4.30.: *User - test: Login success logs*

The screenshot shows the Google Cloud Logs Explorer interface for the 'cntr3' project. The 'Logs' tab is selected, displaying log entries from July 2, 2023. A histogram on the right shows activity between 5:30 AM and 5:37 AM. One entry highlights a session data test:

```

2023-07-02 05:38:37.254 ADT * Running on http://127.0.0.1:5002
2023-07-02 05:38:37.254 ADT * Running on http://169.254.8.1:5002
2023-07-02 05:38:37.254 ADT [34mPress CTRL+C to quit [1m
2023-07-02 05:38:37.294 ADT Default STARTUP TCP probe succeeded after 1 attempt for container "cntr3-1" on port 5002.
2023-07-02 05:38:37.352 ADT [usr/local/lib/python3.11/site-packages/google/cloud/firestore_v1/base_collection.py:290: UserWarning: Detected filter using positio-
2023-07-02 05:38:37.352 ADT return query.where(field_path, op, string, value)
2023-07-02 05:38:37.749 ADT 169.254.1.1 - [02/Jul/2023 08:38:37] "GET /A2/sessionData/usersOnline/vikram@vikram HTTP/1.1" 200 -
2023-07-02 05:38:37.749 ADT 169.254.1.1 - [02/Jul/2023 08:38:37] "GET /A2/sessionData/vikram@vikram HTTP/1.1" 200 -
2023-07-02 05:35:48.695 ADT GET 200 919 B 242 ms Chrome 114.0 https://cntr3-letsmyfaq-uc.a.run.app/A2/sessionData/test@test
2023-07-02 05:35:48.697 ADT GET 200 792 B 202 ms Chrome 114.0 https://cntr3-letsmyfaq-uc.a.run.app/A2/sessionData/usersOnline/test@test
2023-07-02 05:35:48.697 ADT 169.254.1.1 - [02/Jul/2023 08:35:48] "GET /A2/sessionData/usersOnline/test@test HTTP/1.1" 200 -
2023-07-02 05:35:48.853 ADT 169.254.1.1 - [02/Jul/2023 08:35:48] "GET /A2/sessionData/test@test HTTP/1.1" 200 -

```

Figure 4.31.: *User - test: Session data logs*

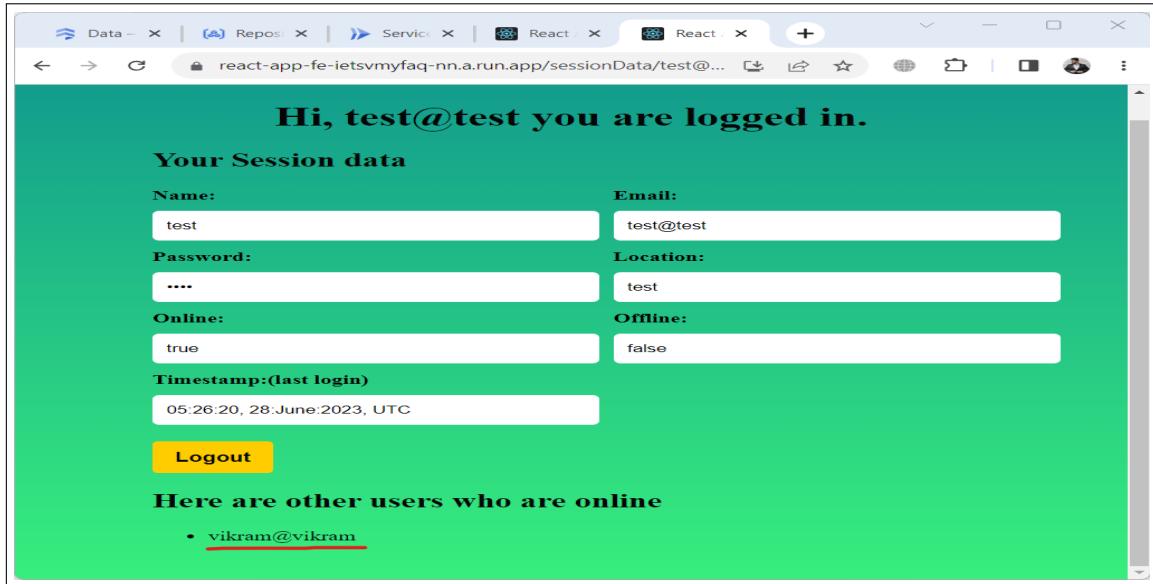


Figure 4.32.: User - test: session data (other online user(s) - "Vikram" is displaying)

Google Cloud

Cloud Firestore in Native mode Database location: northamerica-northeast1

PANEL VIEW QUERY BUILDER

/ > state > test@test

| assignment-2-390620 | state          | test@test                                                                  |
|---------------------|----------------|----------------------------------------------------------------------------|
| + START COLLECTION  | + ADD DOCUMENT | + START COLLECTION                                                         |
| Reg                 | test@test      | vikram@vikram                                                              |
| state               |                | Offline: false<br>Online: true<br>Timestamp: "05:26:20, 28:June:2023, UTC" |

Figure 4.33.: User - test: document created in "state"

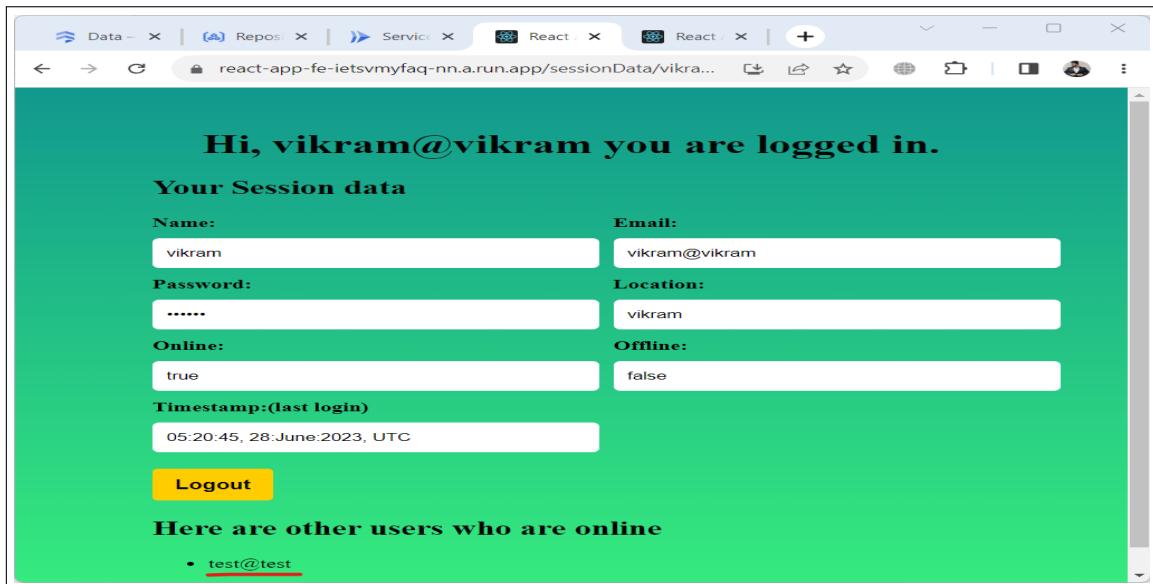


Figure 4.34.: *User - Vikram: session data (other online user(s) - "test" is displaying after refreshing the page)*

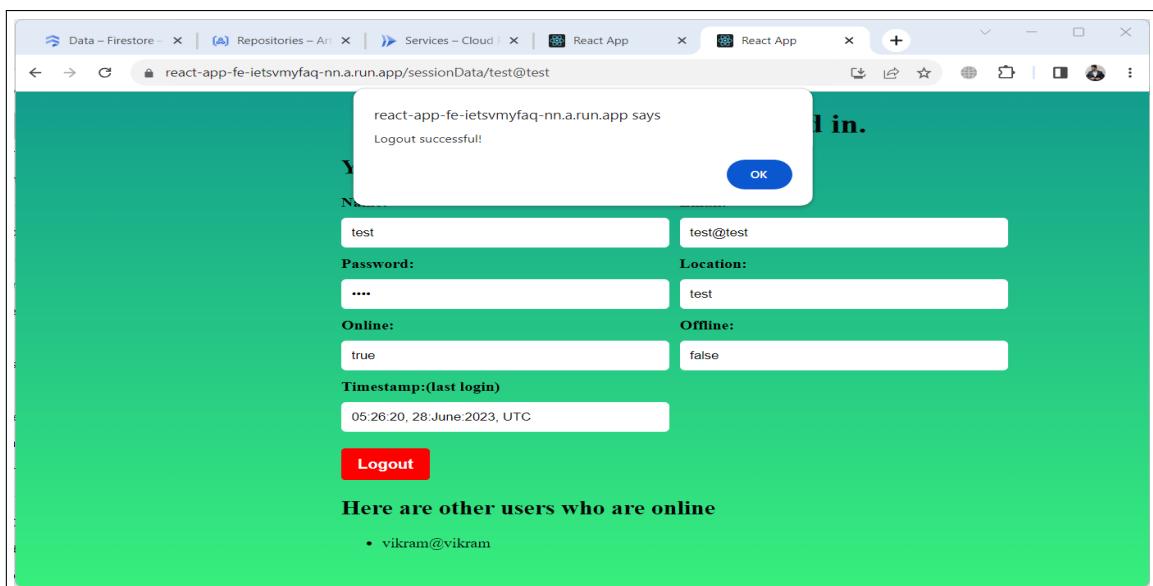


Figure 4.35.: *User - test: Logout success*

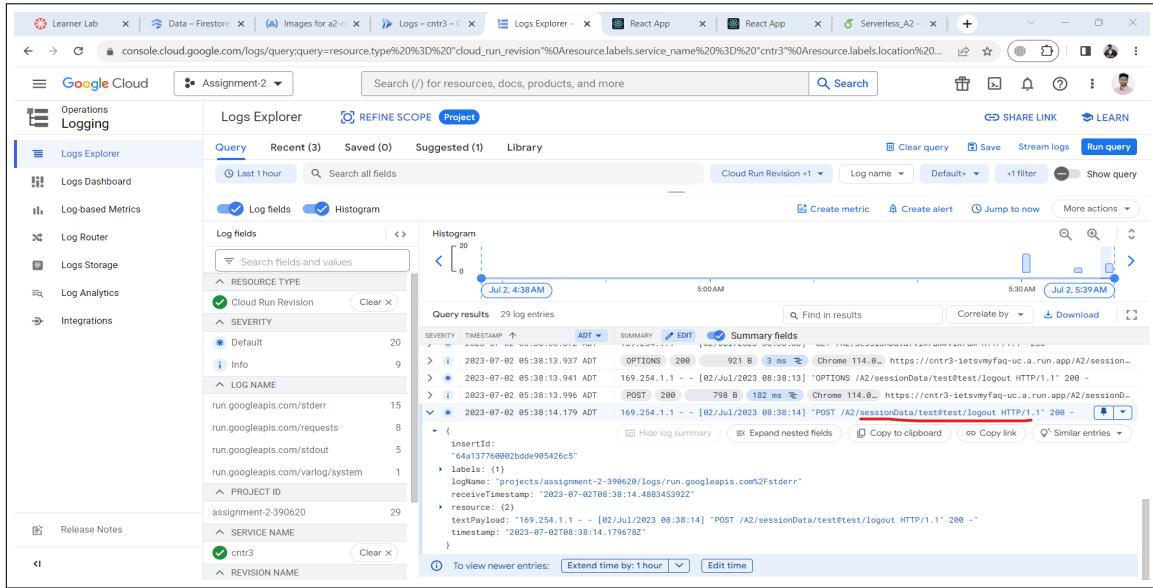


Figure 4.36.: *User - test: Logout logs*

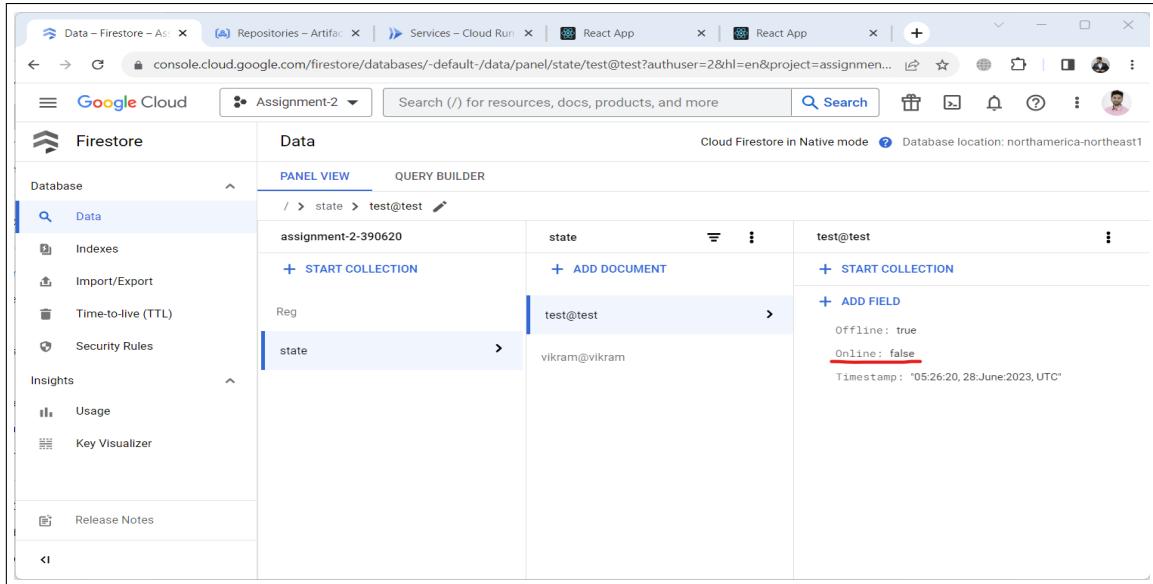


Figure 4.37.: *User - test: Session is Offline*

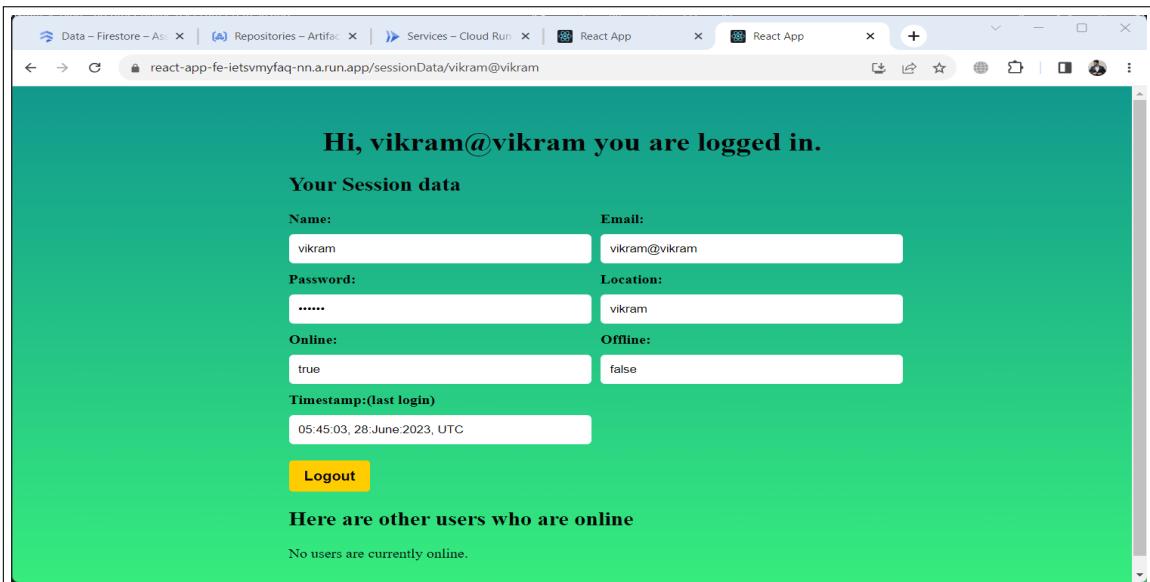


Figure 4.38.: *User - Vikram: no users are online after user: test logged out (after refreshing the page)*

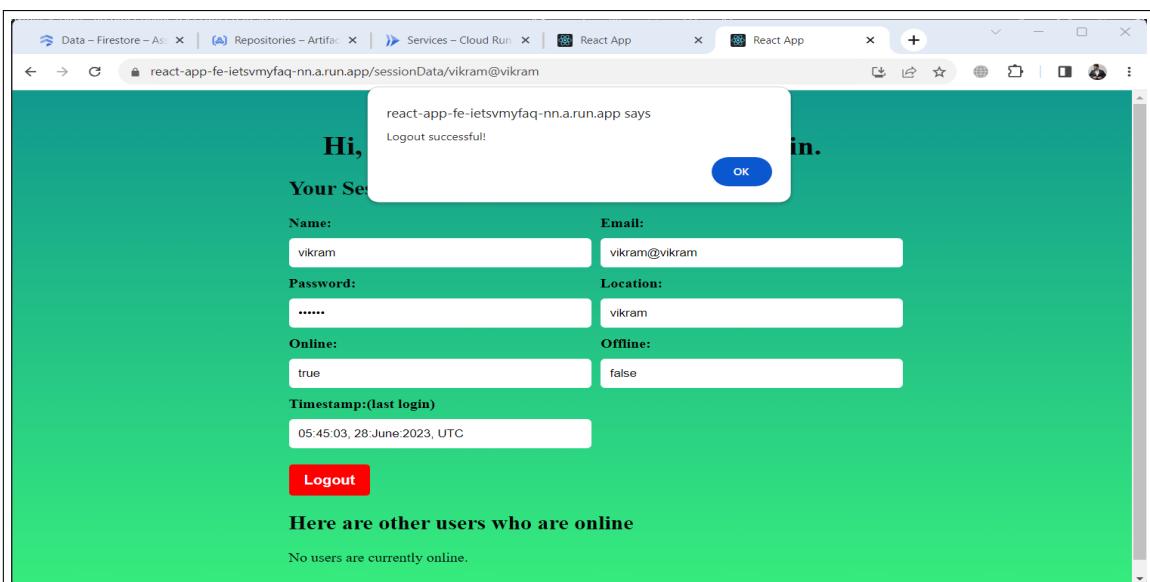


Figure 4.39.: *User - Vikram: Logout success*

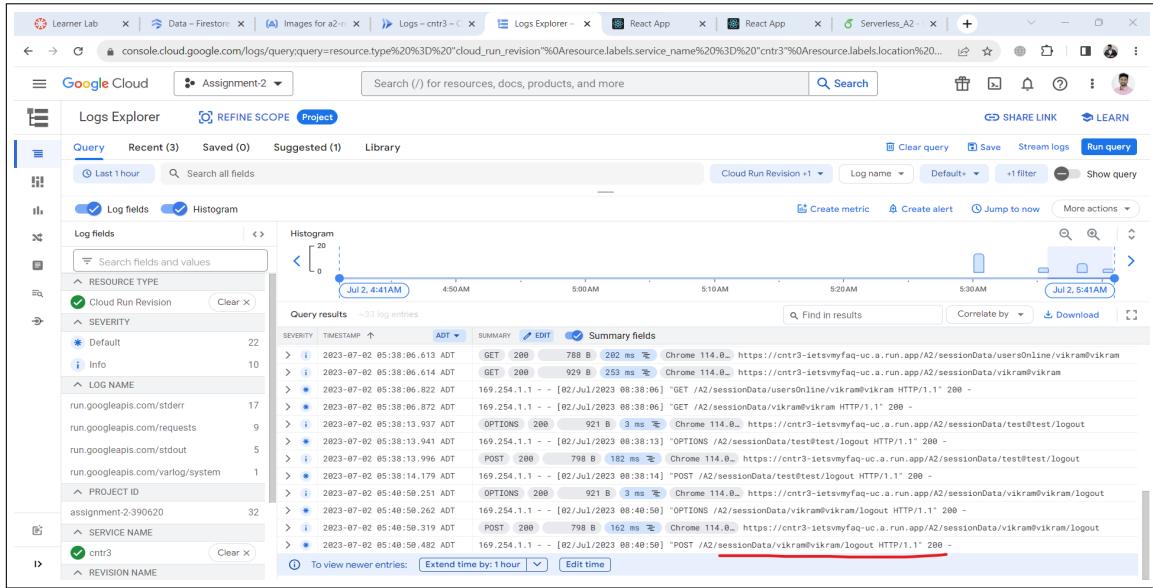


Figure 4.40.: *User - Vikram: Logout logs*

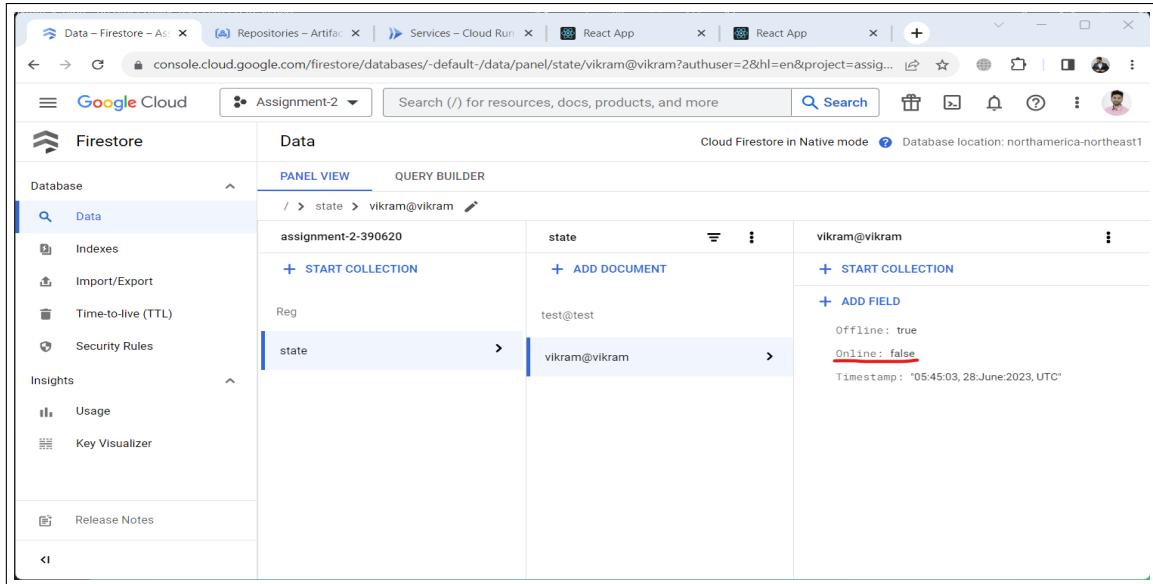


Figure 4.41.: *User - Vikram: Session is offline*

### 4.3. Test cases

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "react-app-fe-ietsvmyfaq-uca.run.app/RegistrationForm". The page itself has a green gradient background and displays a "Registration Form". It contains three input fields: "Name" (with value "test"), "Password" (with value "...."), and "Email" (with value "test"). Below the "Email" field, there is an error message: "L Please include an '@' in the email address. 'test' is missing an '@'." A yellow button labeled "Register" is visible, along with a link "Already registered? Login".

Figure 4.42.: *Email validation - must have '@' symbol*

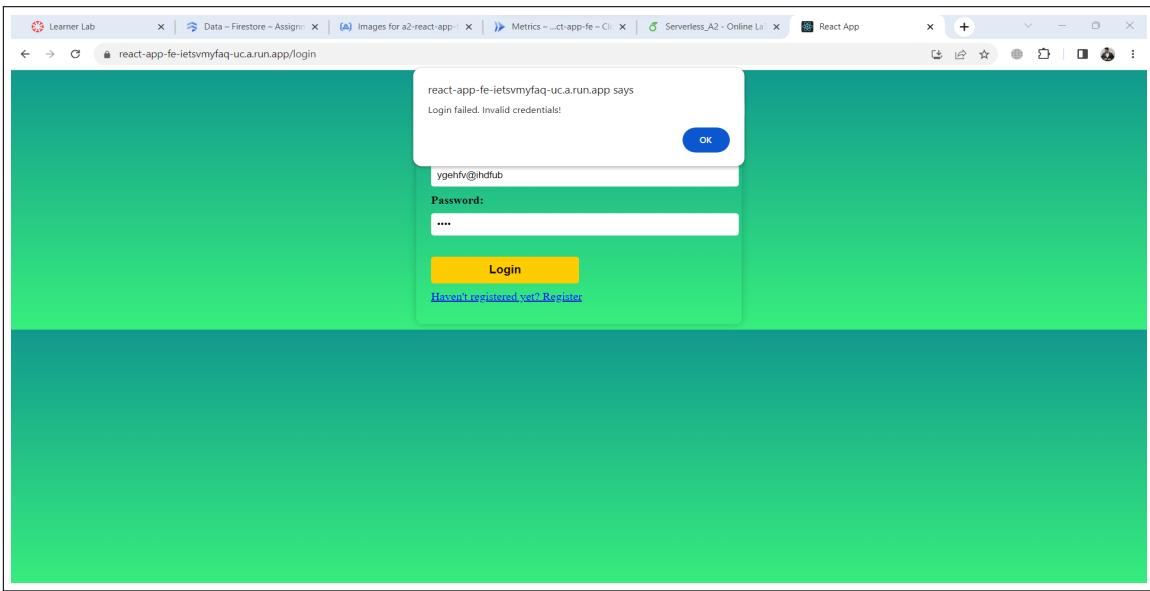


Figure 4.43.: *Input: random credentials, Error: "Invalid credentials" - Since the record is not present in the collection "Reg"*

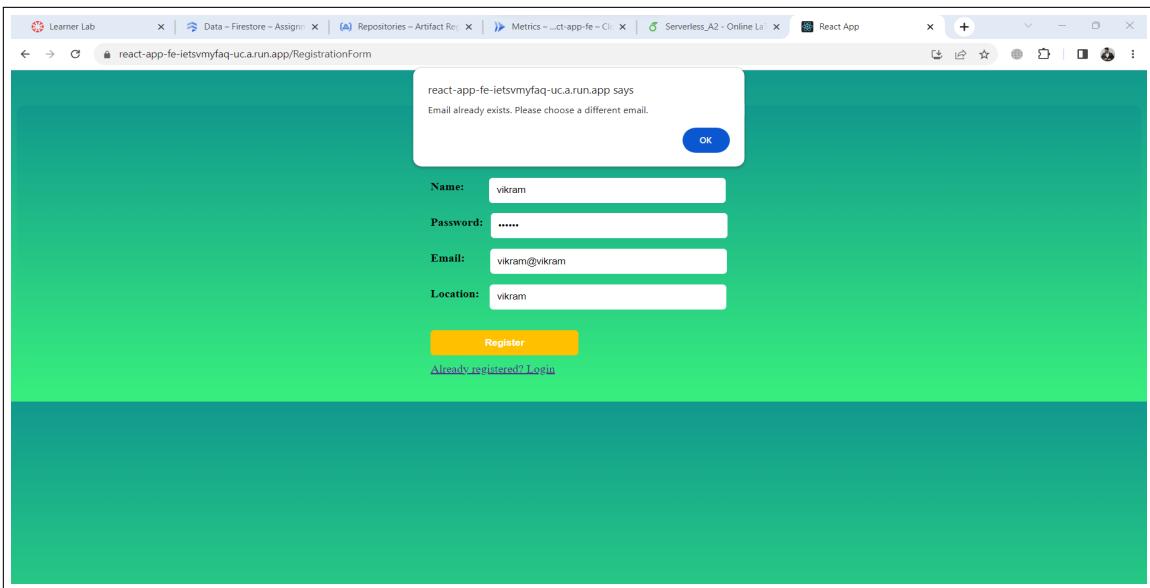


Figure 4.44.: *Input: registration with an already existing email. Error: "Email already exists" - indicating that the entered email is already present in the collection "Reg"*

```

registration_handler.py - Part B - Visual Studio Code

File Edit Selection View Go Run Terminal Help registration_handler.py - Part B - Visual Studio Code

EXPLORER
PART B
> .idea
> cntr1
> _pycache_
microservice1
assignment-2-3906...
cntr1_test.py
Dockerfile
registration_handler...
requirements.txt
sample.py
microservice2
microservice3
Screenshots
Testing
sample_test.py
assignment-2-3906...
docker commands.txt
docker-compose.yml
package-lock.json
package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
FS E:\Nikra\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B & C:/Users/vkra/AppData/Local/Microsoft/Windows/Python/python3.10.exe "C:/Users/vkra/Desktop/WACS/TERM-2/CSCI 5410 Serverless Data Processing/Assignment/A2-v2/csci5410-summer-23-b00936916/Part B/microservice1/test.py"
Test Case 1: Registration Success ; Response code 200 - PASSED ✅
Test Case 2: Registration Failed, Exception : 'Email already exists' ; Response code 409 - PASSED ✅
PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B>

```

Figure 4.45.: *Test cases for container 1*

```

cntr2_test.py - Part B - Visual Studio Code

File Edit Selection View Go Run Terminal Help cntr2_test.py - Part B - Visual Studio Code

EXPLORER
PART B
> .idea
> cntr1
> _pycache_
microservice1
assignment-2-3906...
cntr1_test.py
Dockerfile
login_handler.py
requirements.txt
sample.py
microservice2
Screenshots
Testing
cntr1_test.py
cntr2_test.py
sample_test.py
assignment-2-3906...
docker commands.txt
docker-compose.yml
package-lock.json
package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B & C:/Users/vkra/AppData/Local/Microsoft/Windows/Python/python3.10.exe "C:/Users/vkra/Desktop/WACS/TERM-2/CSCI 5410 Serverless Data Processing/Assignment/A2-v2/csci5410-summer-23-b00936916/Part B/test1/cntr2/test.py"
Test Case 1: Login Success ; Response code 200 - PASSED ✅
Test Case 2: Login Failed, Exception : 'Invalid credentials / Credentials not found' ; Response code 404 - PASSED ✅
PS C:\Users\vkra\Desktop\WACS\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci5410-summer-23-b00936916\Part B>

```

Figure 4.46.: *Test cases for container 2*

```

File Edit Selection View Go Run Terminal Help cnt3_test.py - Part B - Visual Studio Code
EXPLORER PART B
> .idea
> cntr1
> microservice1
 () assignment-2-3906...
 Dockerfile
 login_handler.py
 requirements.txt
 microservice2
 () assignment-2-3906...
 Dockerfile
 requirements.txt
 session_handler.py
 Screenshot
 Testing
 cnt1_test.py
 cnt2_test.py
 cnt3_test.py
 sample_test.py
 () assignment-2-390620...
 docker_commands.txt
 docker-compose.yml
 package-lock.json
 package.json
 OUTLINE TIMELINE
0 0 GitLens Pro (trial)

cnt3_test.py - Part B - Visual Studio Code
cntr3_test.py - Part B - Visual Studio Code
Testing > cnt3_test.py > session_handler.py > make_user_online
43 assert len(response_data) == 0
44 print("test Case 2: Get other online users ; No users - PASSED",
45 green_tick)
46 except AssertionError as e:
47 print("test Case 2: Get other online users ; No users - FAILED:",
48 red_cross, e)
49
50 def make_user_online():
51 cntr2_url = "https://cntr2-ietsvmyfaq-uc.a.run.app"
52 login_api = "/az/login"
53 login_data = {
54 "email": "vikram@iitk.ac.in",
55 "password": "vikram123"
56 }
57
58 response = requests.post(cntr2_url + login_api, data=login_data)
59 response_data = response.json()
60
61 assert response.status_code == 200
62 assert response_data["status"] == "Success"
63 assert response_data["message"] == "User registered successfully"
64
65 user_id = response_data["user_id"]
66
67 # Test Case 3: Get other online users ; users_count > 0 - PASSED
68 response = requests.get(cntr2_url + login_api)
69 response_data = response.json()
70
71 assert response.status_code == 200
72 assert response_data["status"] == "Success"
73 assert response_data["message"] == "User registered successfully"
74
75 user_id = response_data["user_id"]
76
77 # Test Case 4: Get other online users ; users_count > 0 - FAILED
78 response = requests.get(cntr2_url + login_api)
79 response_data = response.json()
80
81 assert response.status_code == 200
82 assert response_data["status"] == "Success"
83 assert response_data["message"] == "User registered successfully"
84
85 user_id = response_data["user_id"]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\vikra\Desktop\IITK\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci15410-summer-23-b00936916\Part B & C:\Users\vikra\appData\local\Microsoft\Windows\Start Menu\Programs\Python3.10\Python.exe "C:/users/vikra/Desktop/IITK/TERM-2/CSCI 5410 Serverless Data Processing/Assignment/A2-v2/csci15410-summer-23-b00936916/Part B/test\log\cnt3.log" test
Test Case 1: Get session data Success ; Response code 200 - PASSED ✓
Test Case 2: Get other online users ; No users - PASSED ✓
User - Vikram : Login success
Test Case 3: Get other online users ; users_count > 0 - PASSED ✓
User - Vikram : Login success
Test Case 4: Get other online users ; users_count > 0 - FAILED
User - Vikram : Login failed
PS C:\Users\vikra\Desktop\IITK\TERM-2\CSCI 5410 Serverless Data Processing\Assignment\A2-v2\csci15410-summer-23-b00936916\Part B>

You, now Lin 57, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store) ⚡

```

Figure 4.47.: *Test cases for container 3*

## 4.4. My view on leveraging Google Cloud Run, GCR/Artifact Registry, and Docker Containers for Efficient Application Deployment

My application relies on the power of Google Cloud technologies to achieve seamless deployment and management. By utilizing Google Cloud Run[3], I can deploy my application as stateless containers, benefitting from automatic scaling and secure execution. This allows me to focus on developing my application code while leaving the infrastructure complexities to Cloud Run.

To store and manage my container images, I utilized Google Artifact Registry[2]. This central repository enabled me to version control my images, control access to containers, and efficiently distribute them across different environments. With easy push and pull capabilities[8], I can ensure smooth deployment and updates[7] of my container images.

Docker containers[6] play a vital role in the deployment process. They encapsulate the application code, dependencies, and configurations, making deployments consistent and portable. Docker's containerization technology provides isolation, scalability, and optimized resource utilization, contributing to improved performance and maintainability of the application.

### My personal observation:

During the process of pushing Docker images to Artifact Registry, I observed an interesting behavior. When pushing the "cnt2" images, I noticed that instead of uploading all the layers from scratch, the system fetched some layers from the previously pushed "cntr1" images (Refer figures 4.2,4.4,4.6). Since these layers were identical, reusing them saved valuable time and resources. This optimization in the image pushing process demonstrates the efficiency of Google Artifact Registry in handling and

managing container images, further enhancing the overall deployment experience.

In conclusion, my application benefits from Google Cloud Run's scalability, Artifact Registry's image management capabilities, and Docker containers' consistency and portability. These technologies work harmoniously to provide me with an efficient, secure, and easily manageable application infrastructure. Additionally, Docker and Google Cloud Platform offer extensive documentation and resources, which enabled me to maximize the benefits of these tools in my application development and deployment process.

**Part III.**

**PART C**

---

---

## Building a Chatbot: Using AWS Lex

---

---

## 5. Bot RideRequest

### 5.1. Actions taken to perform the bot creation operation

To create the ”**TaxiRequest**” and ”**SelfDriveRequest**” intents in AWS Lex, log in to your AWS account and access the AWS Management Console. From there, open the Amazon Lex service and either create a new bot. Once inside the bot, navigate to the ”**Intents**” section and click on ”**Create Intent.**” For the ”TaxiRequest” intent, name it accordingly and add sample utterances that users might say when requesting a taxi ride. Create the necessary slots to collect information like **pickup address**, **pickup date**, **pickup time**, and **vehicle type**. Configure prompts for each slot to prompt the user for the required details. Similarly, for the ”SelfDriveRequest” intent, name it accordingly and add sample utterances related to self-drive requests. Create slots to collect information such as **arrival time**, **vehicle type**, and **number of vehicles**. Configure prompts for each slot to obtain the necessary details from the user. Define **confirmation prompts** for both intents using placeholders for variable values to confirm the user’s request. Set up the **fulfillment logic** for each intent, including appropriate success and failure messages. Save the intent configurations and **build the bot** to apply the changes. These steps allow us to create the ”TaxiRequest” and ”SelfDriveRequest” intents in AWS Lex and we can customize them according to our requirements. Finally, **test the bot**, to check if it satisfies all the requirements.

### 5.2. Intents

#### 5.2.1. TaxiRequest

The ”TaxiRequest” intent allows users to request a taxi ride. It collects information such as the customer’s address for pickup, pickup date, and pickup time.

##### Sample utterances:

Can you call me a taxi?  
I need a taxi to the airport.  
I need a ride to the train station.  
I’m looking for a taxi service.

##### Create Custom Slot type Vehicle

1. Create a new, Custom slot type in the bot
2. Slot name: Vehicle

3. Slot type values(restrict to these values only) :

- a) Key = suv ; Values = SUV, suv, Suv
- b) Key = minivan ; Values = Minivan, MiniVan, minivan

### **Slots & Prompts**

| <b>Slots</b>            | <b>Prompts</b>                                                                       | <b>Slot Type</b>             |
|-------------------------|--------------------------------------------------------------------------------------|------------------------------|
| <b>CustomerAddress</b>  | Sure, you've selected a taxi ride. Could you please provide your address for pickup? | AMAZON.FreeFormInput         |
| <b>PickupDate</b>       | Thank you! When do you need the taxi? Please provide the pickup date.                | AMAZON.Date                  |
| <b>PickupTime</b>       | Great! What time would you like the taxi to arrive? Please provide the pickup time.  | AMAZON.Time                  |
| <b>VehicleType</b>      | What do you want today (SUV, Sedan, Mini-van)?                                       | AMAZON.Vehicle(custom built) |
| <b>NumberOfVehicles</b> | How many?                                                                            | AMAZON.Number                |

### **Confirmation prompt:**

Use this prompt with placeholders for variable values (indicated using {} ), to get confirmation from the user(YES/NO).

- ```
1 You have requested for {NumberOfVehicles} {VehicleType} for pickup address  
2 {CustomerAddress}. Your pickup date is {PickupDate}, and your pickup time is {  
  PickupTime}. Please confirm.
```

Listing 5.1: Code snippet

Fulfillment :

On success	Your request has been placed successfully.
In case of failure	Oops!! Sorry, your request could not be placed at this moment. Please try again later.

5.2.2. SelfDriveRequest

The SelfDriveRequest intent is used to request a self-drive ride or car rental. It collects information about the vehicle type, number of vehicles, and pickup time.

Sample utterances:

I want to request a self-drive ride.
I'm looking to rent a car for the day.
I need a car to pick up my kids from school.
I'm going to the airport and need a car to get there.

Slots & Prompts

Slots	Prompts	Slot Type
ArrivalTime	Sure, you've selected a self-drive ride. When are you coming TODAY to get your vehicle?	AMAZON.Time
VehicleType	What do you want today (SUV, Sedan, Mini-van)?	AMAZON.Vehicle(custom built)
NumberOfVehicles	How many?	AMAZON.Number

Confirmation prompt:

Use this prompt with placeholders for variable values (indicated using {}), to get confirmation from the user (YES/NO).

```
1 You have requested for {NumberOfVehicles} {VehicleType}, and you will be
   arriving at {ArrivalTime}. Please confirm.
```

Listing 5.2: Code snippet

Fulfillment :

On success	Your request has been placed successfully.
In case of failure	Oops!! Sorry, your request could not be placed at this moment. Please try again later.

5.3. Screenshots

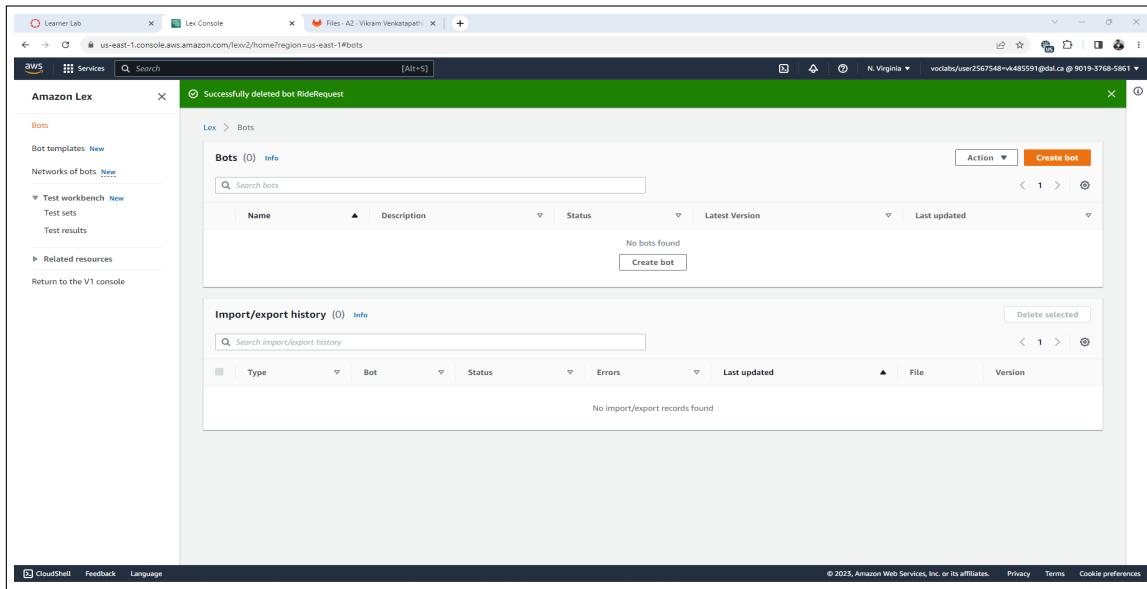


Figure 5.1.: *Empty bots section*

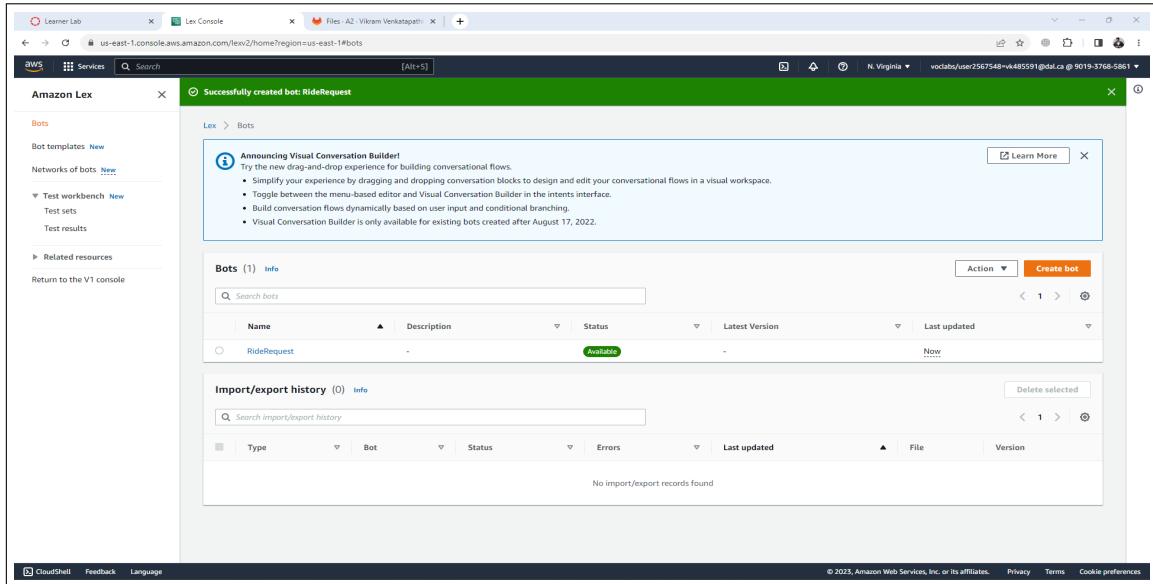


Figure 5.2.: *Bot created*

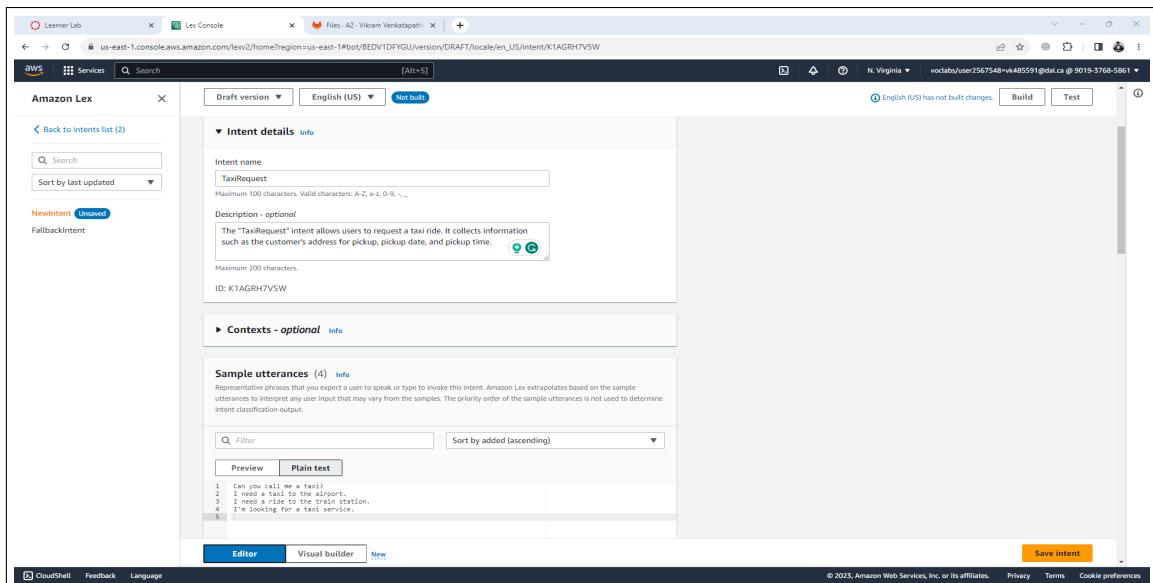


Figure 5.3.: *TaxiRequest intent creation*

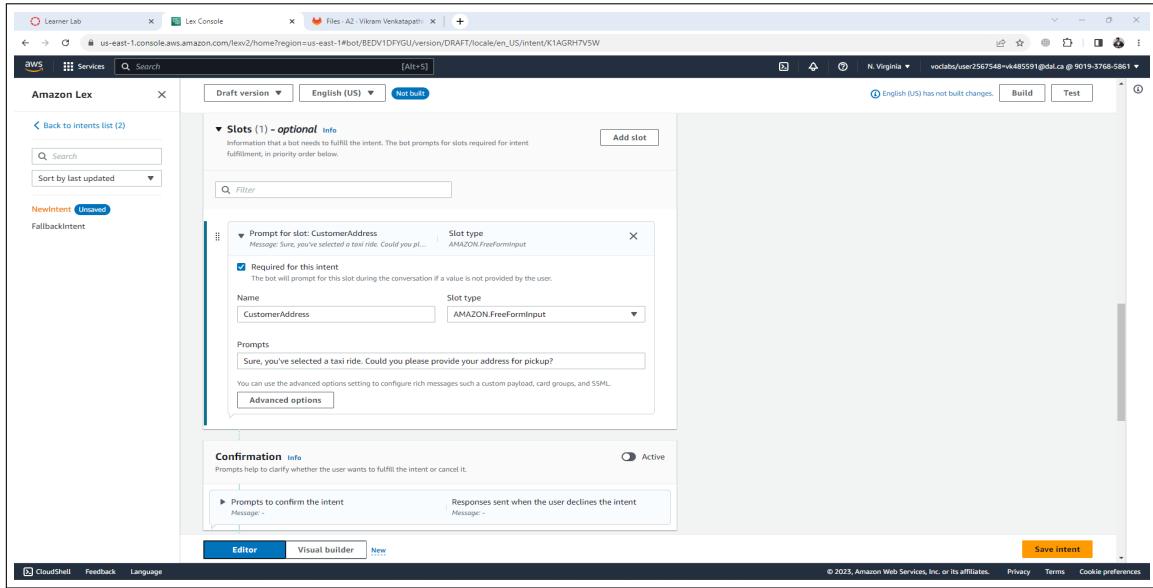


Figure 5.4.: *TaxiRequest intent creation - CustomerAddress slot*

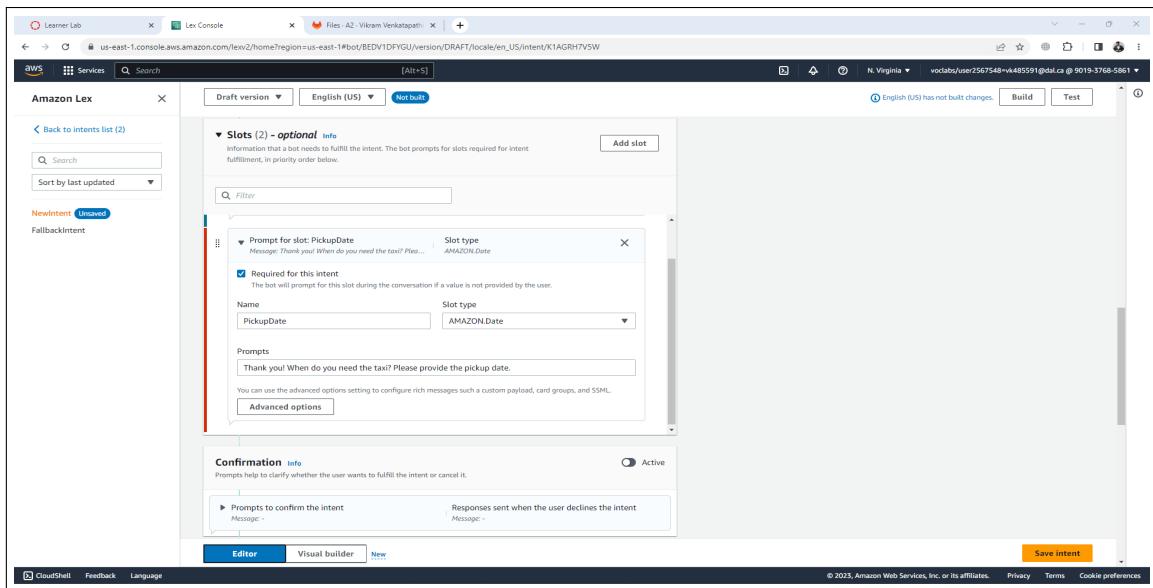


Figure 5.5.: *TaxiRequest intent creation - PickupDate slot*

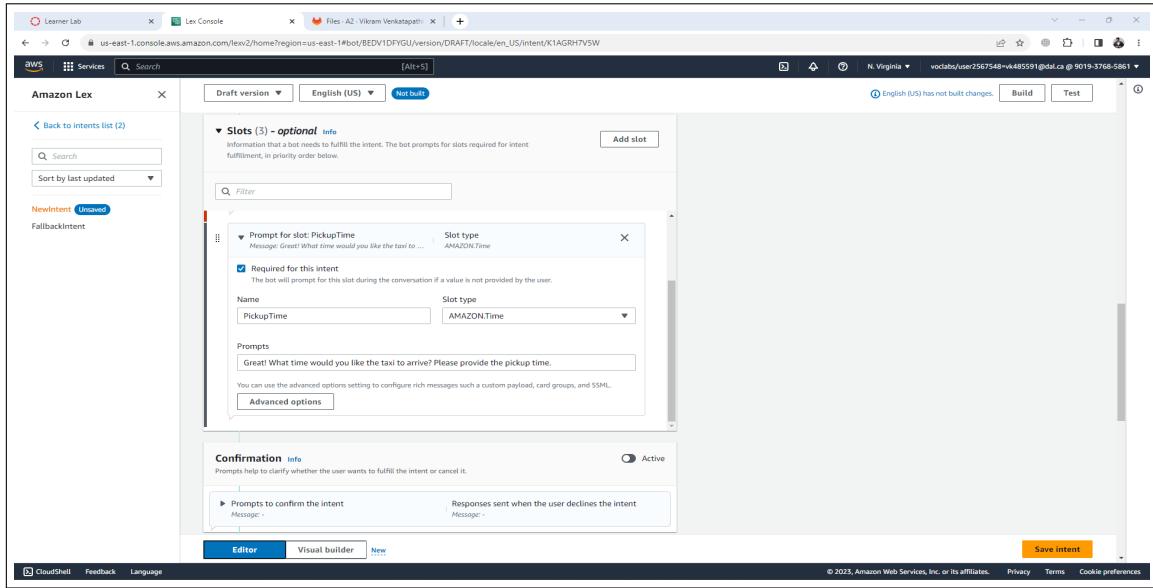


Figure 5.6.: *TaxiRequest intent creation - PickupTime slot*

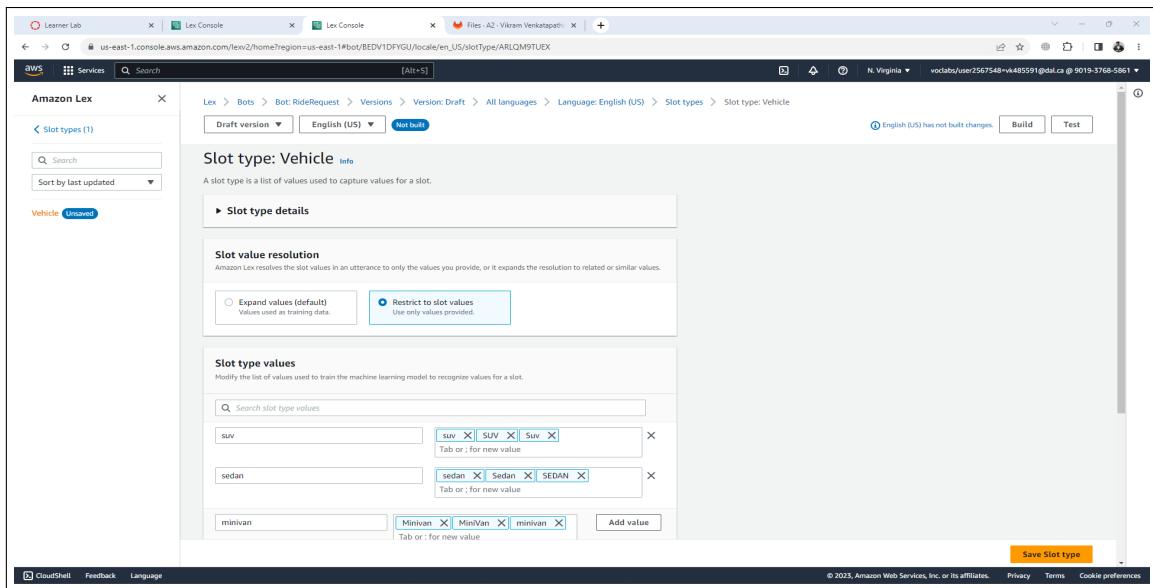


Figure 5.7.: *TaxiRequest intent creation - Custom slot type 'Vehicle' creation*

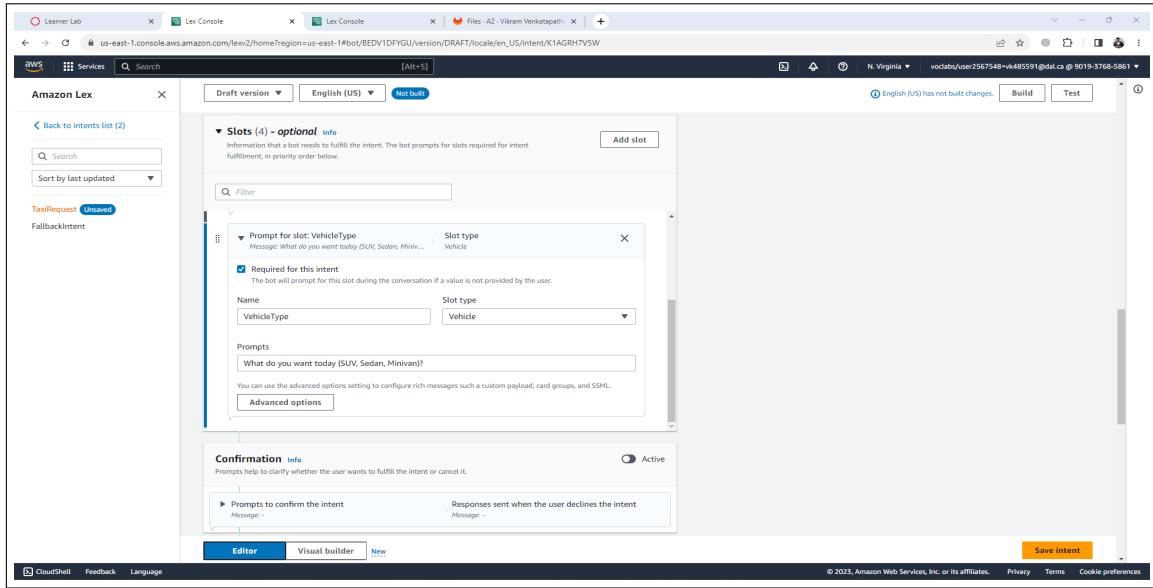


Figure 5.8.: *TaxiRequest intent creation - VechicleType slot*

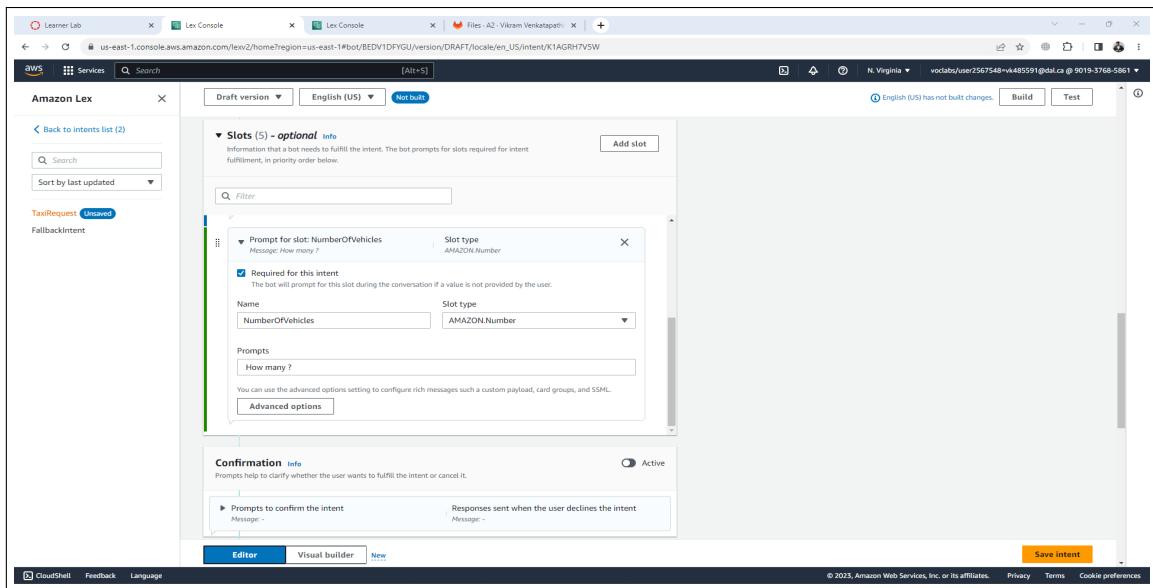


Figure 5.9.: *TaxiRequest intent creation - NumberOfVehicles slot*

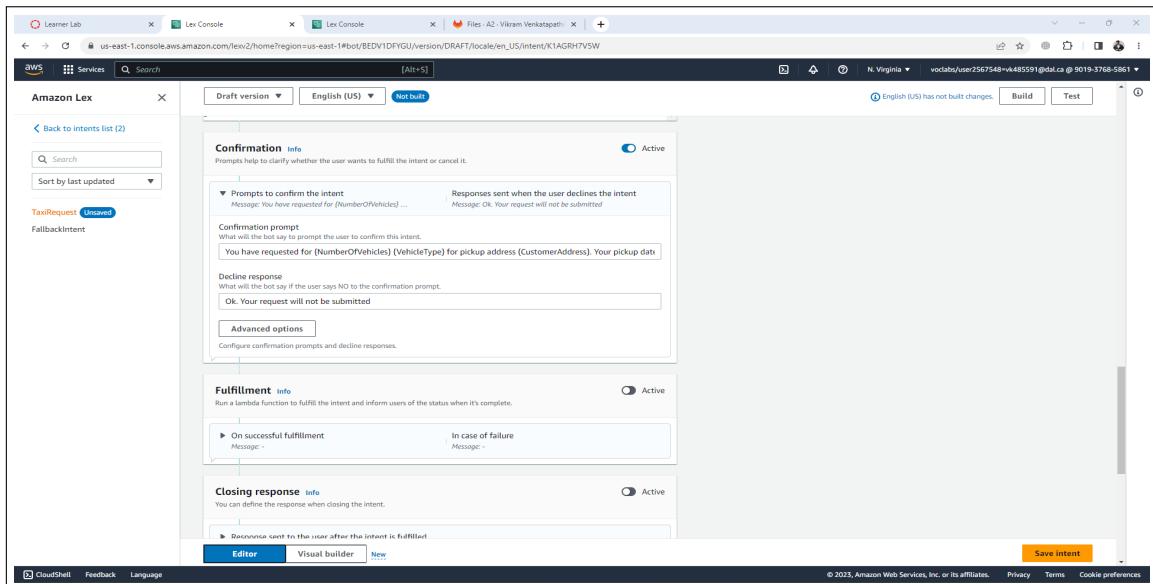


Figure 5.10.: *TaxiRequest intent creation - Add Confirmation prompt with placeholder values*

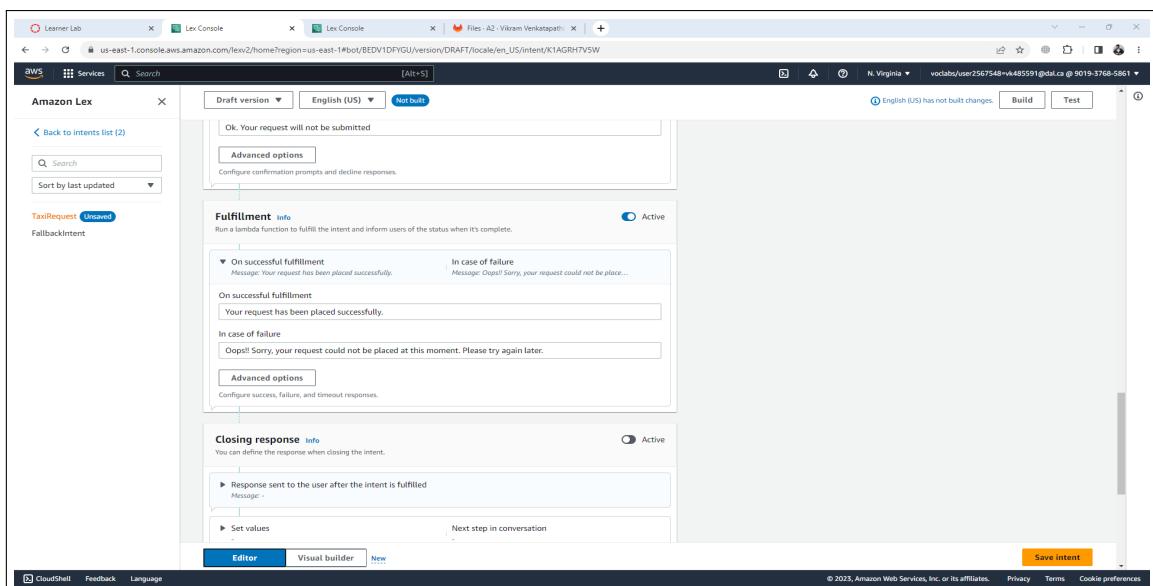


Figure 5.11.: *TaxiRequest intent creation - Add fulfillment prompt*

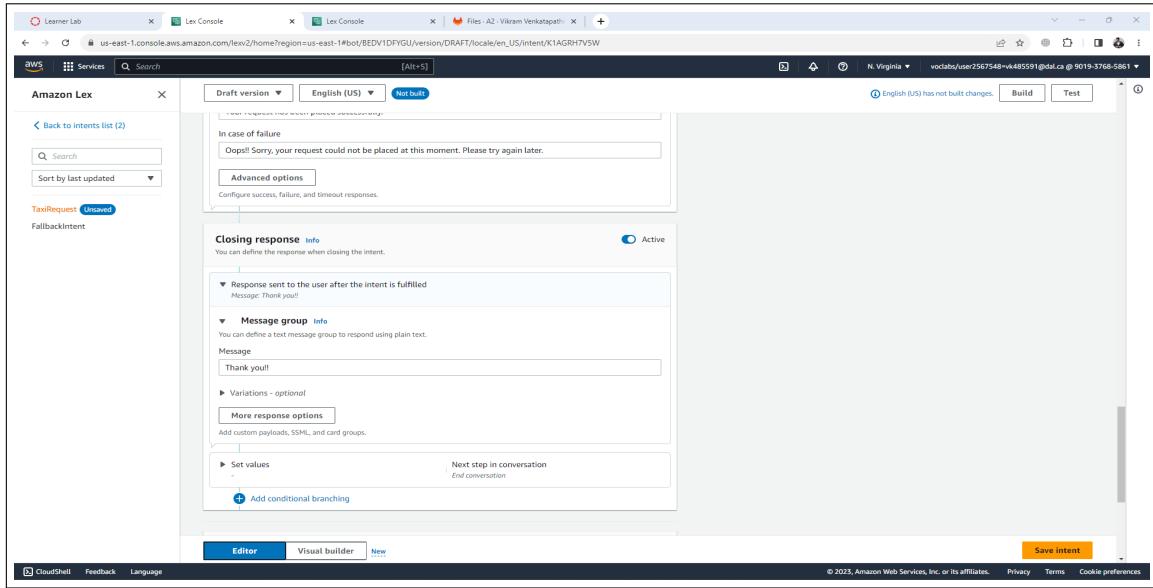


Figure 5.12.: *TaxiRequest intent creation - Add closing response*

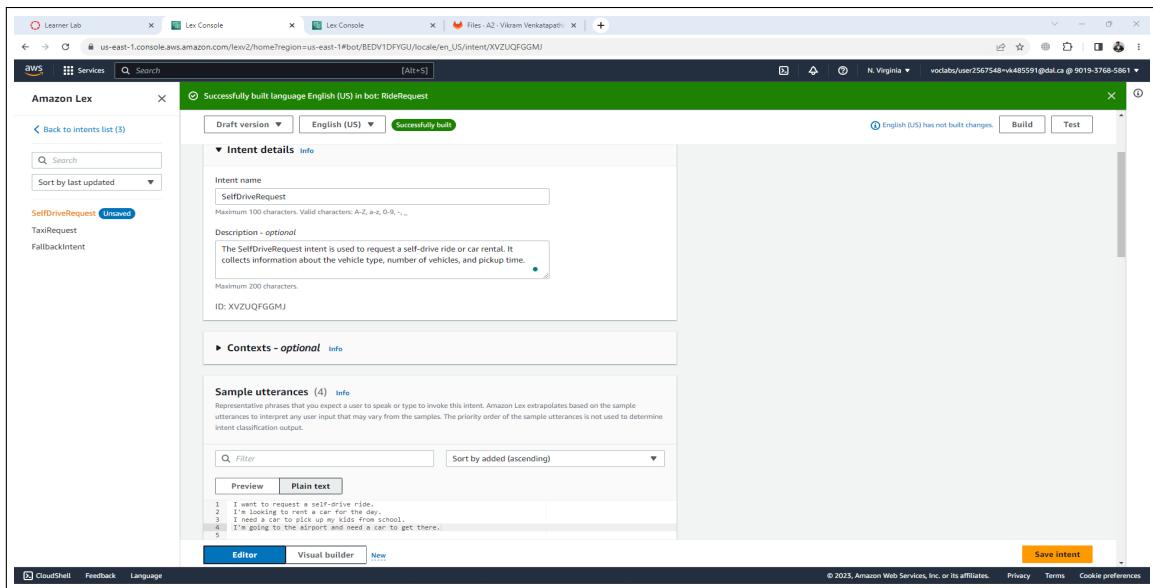


Figure 5.13.: *SelfDriveRequest intent creation*

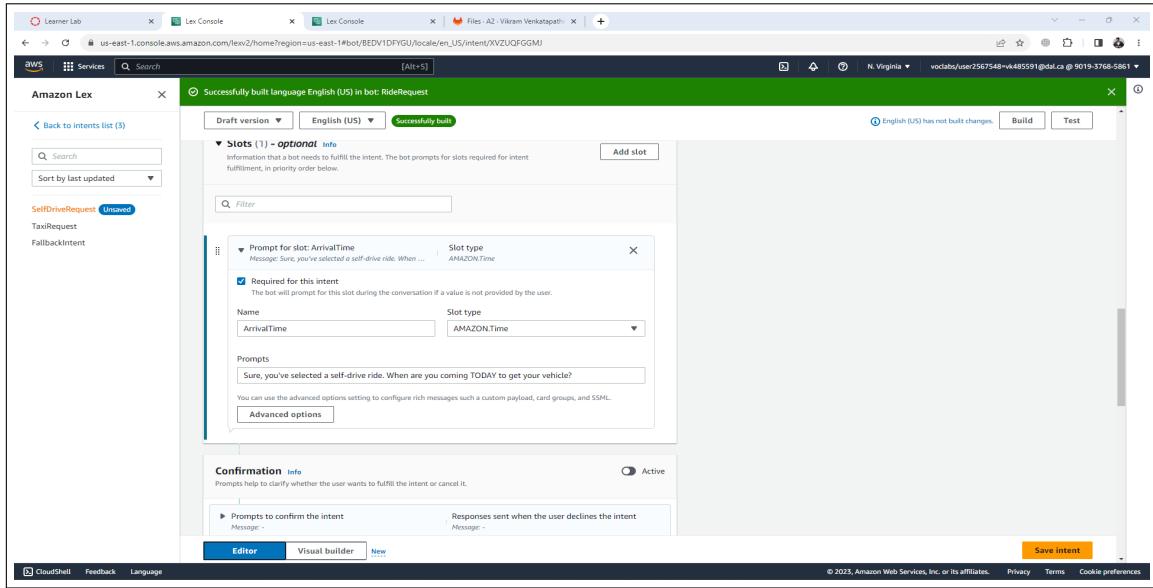


Figure 5.14.: *SelfDriveRequest intent creation - ArrivalTime slot*

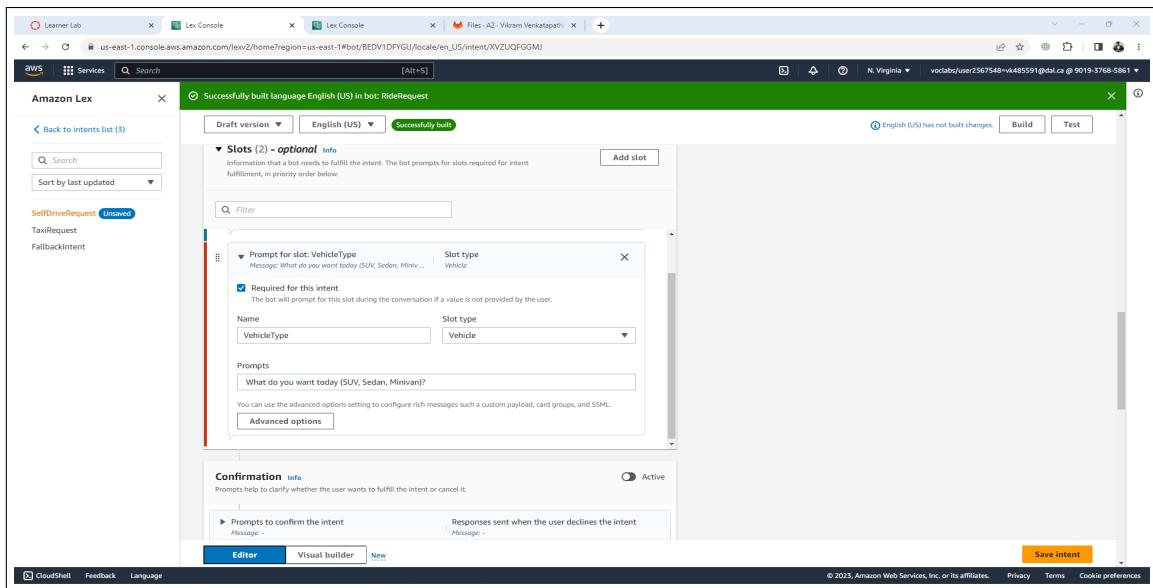


Figure 5.15.: *SelfDriveRequest intent creation - VehicleType slot*

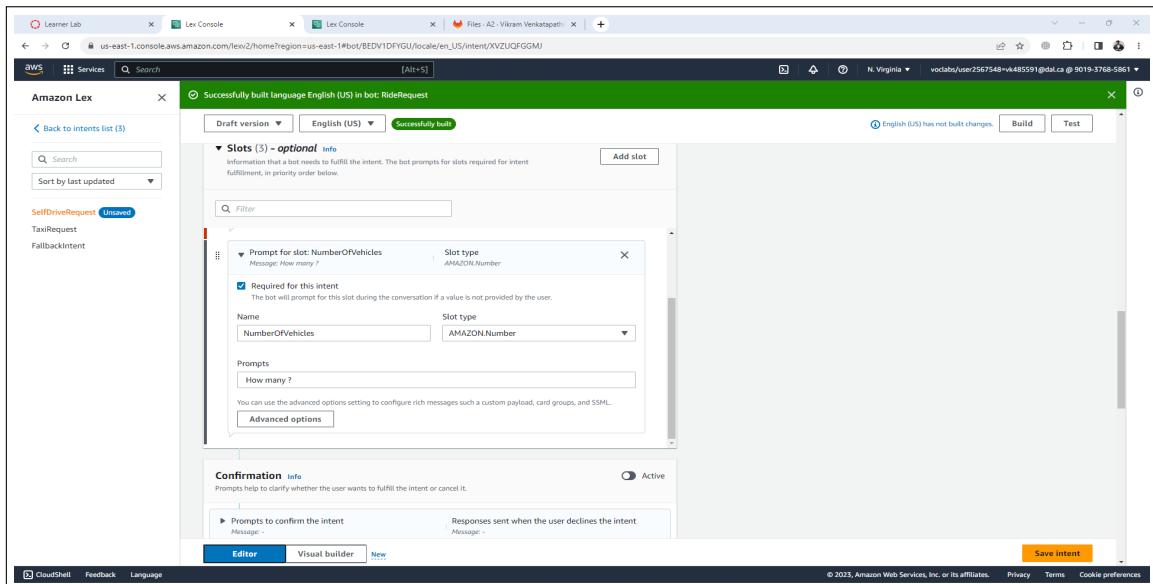


Figure 5.16.: *SelfDriveRequest intent creation - NumberOfVehicles slot*

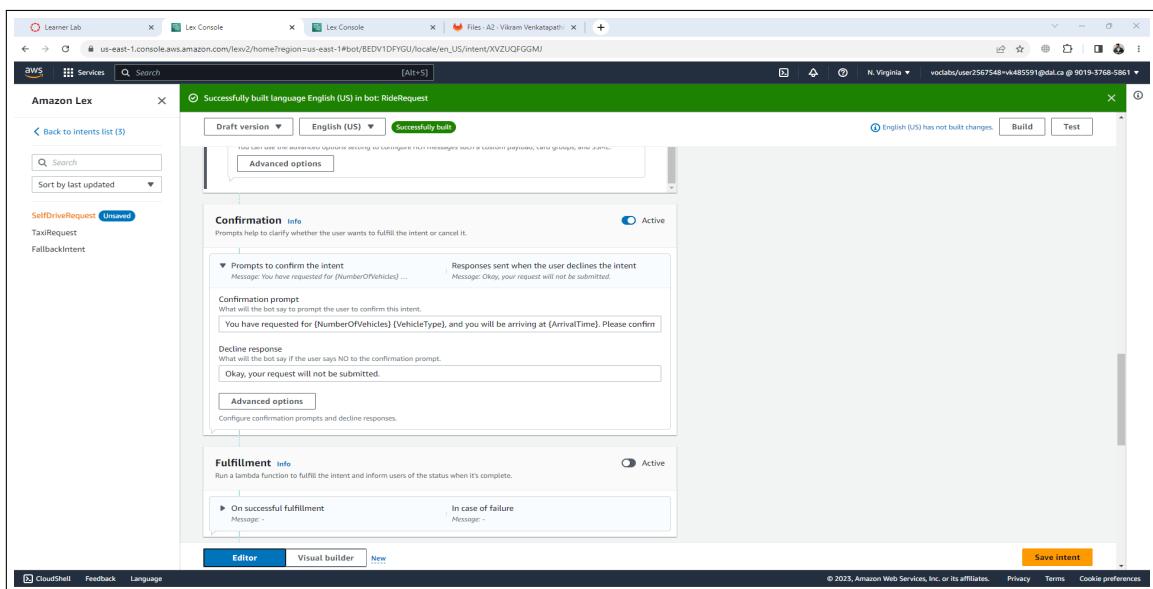


Figure 5.17.: *SelfDriveRequest intent creation - Add Confirmation prompt with placeholder values*

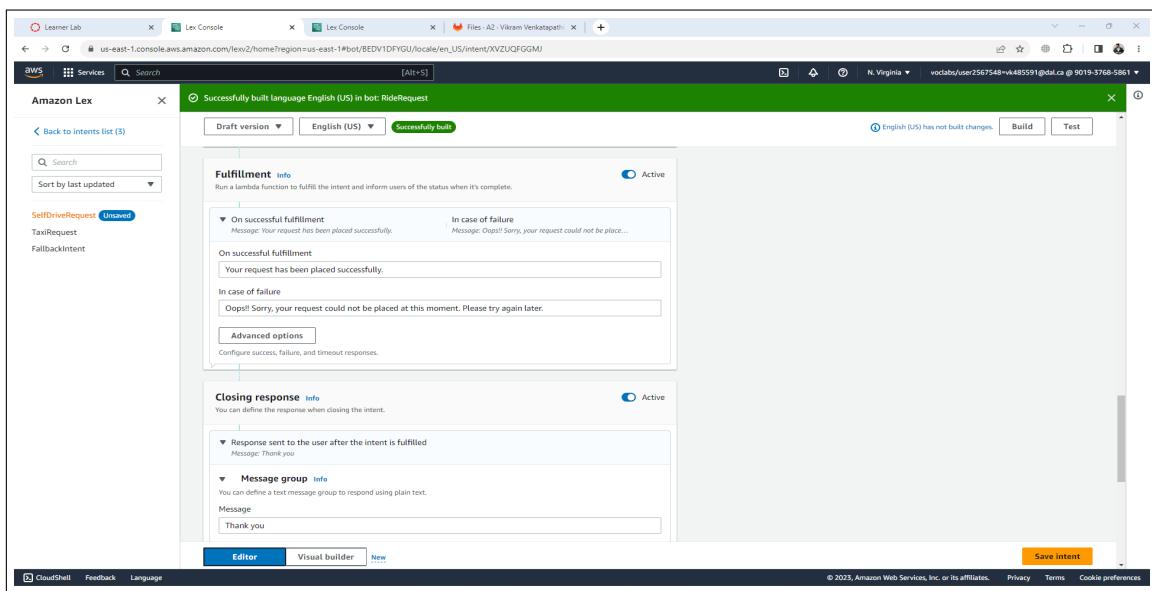


Figure 5.18.: *SelfDriveRequest intent creation - Add Fulfilment prompt and Closing response*

5.4. Output

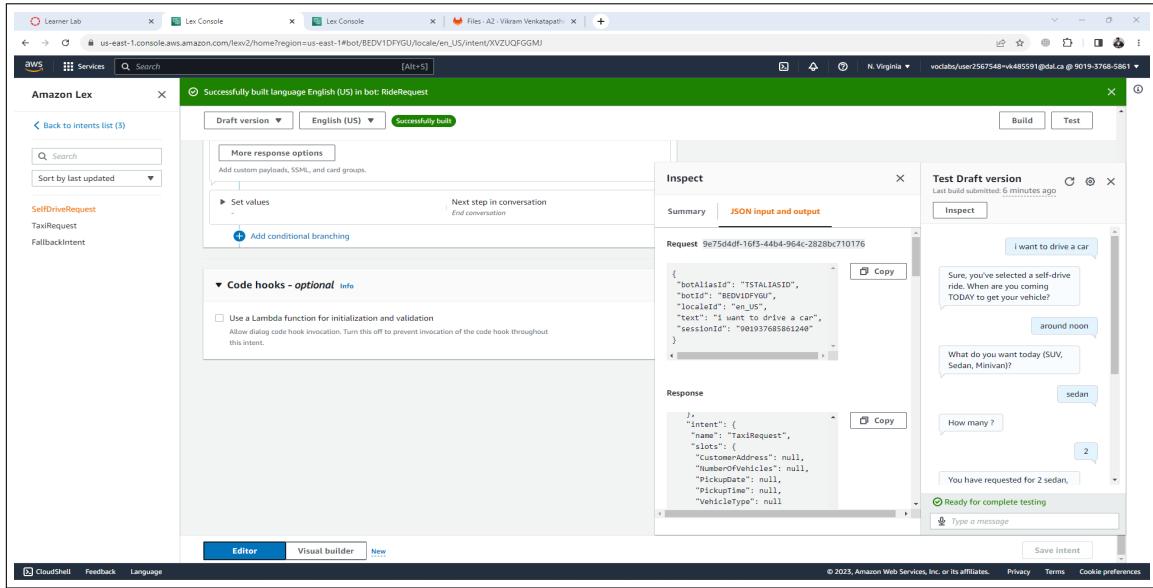


Figure 5.19.: *SelfDrive - Confirm request 1.1*

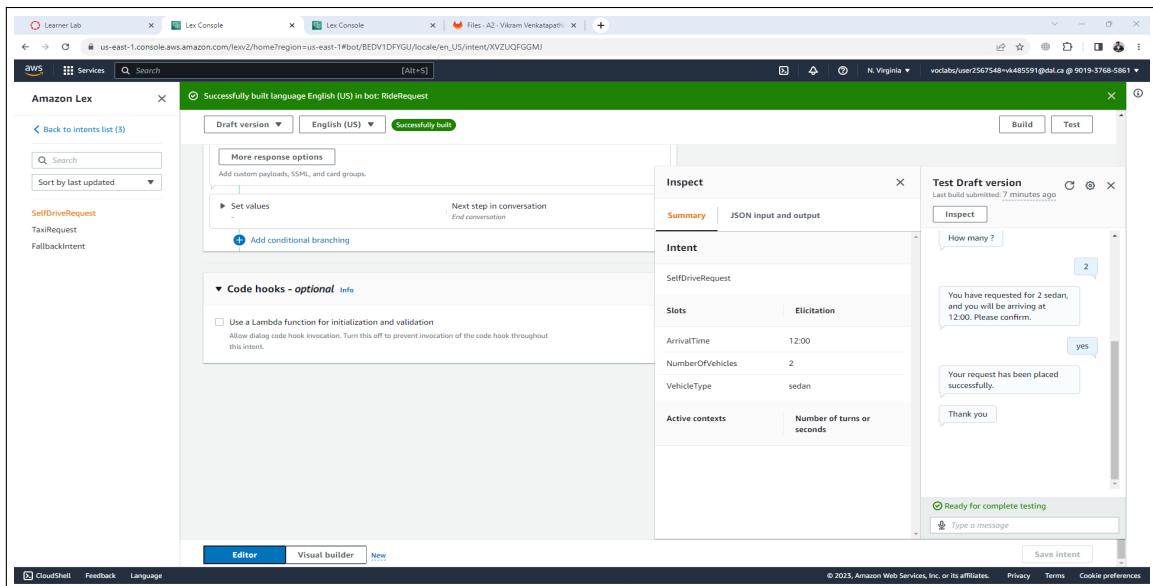


Figure 5.20.: *SelfDrive - Confirm request 1.2*

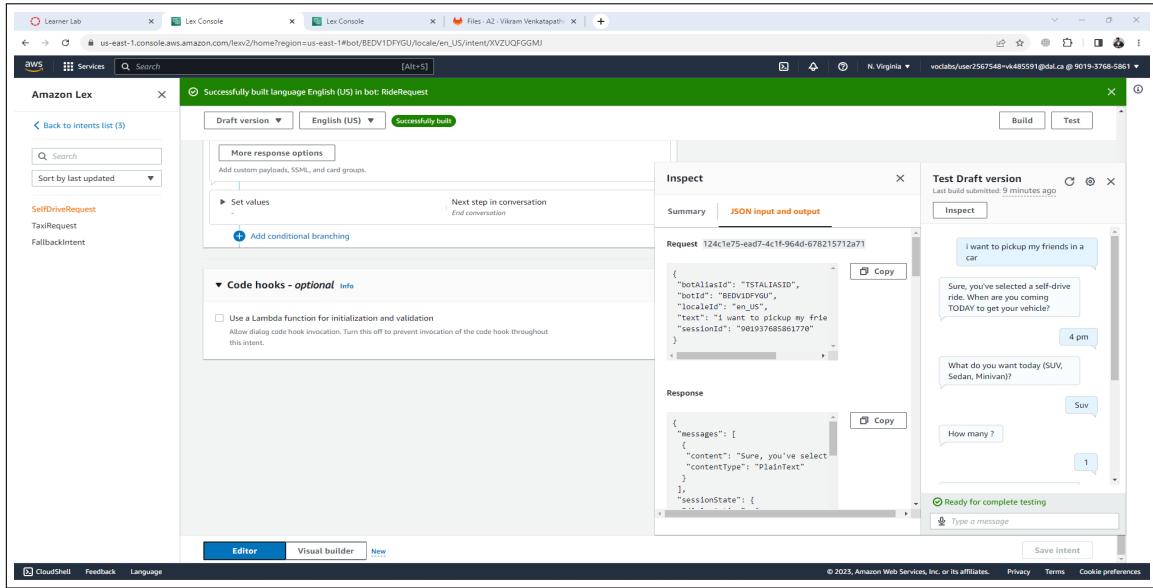


Figure 5.21.: *SelfDrive - Cancel request 1.1*

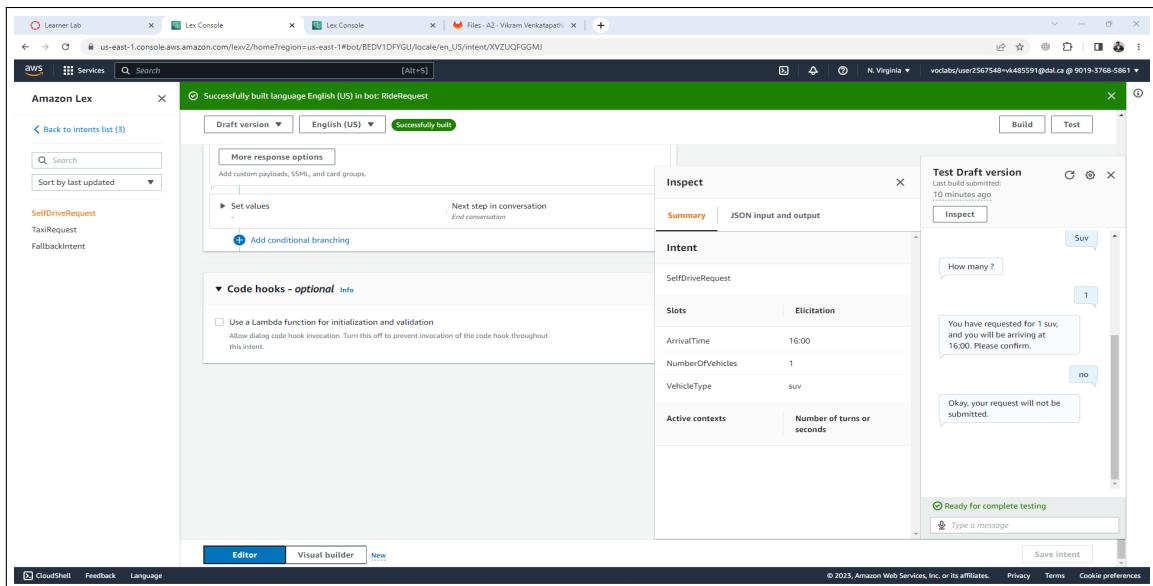


Figure 5.22.: *SelfDrive - Cancel request 1.2*

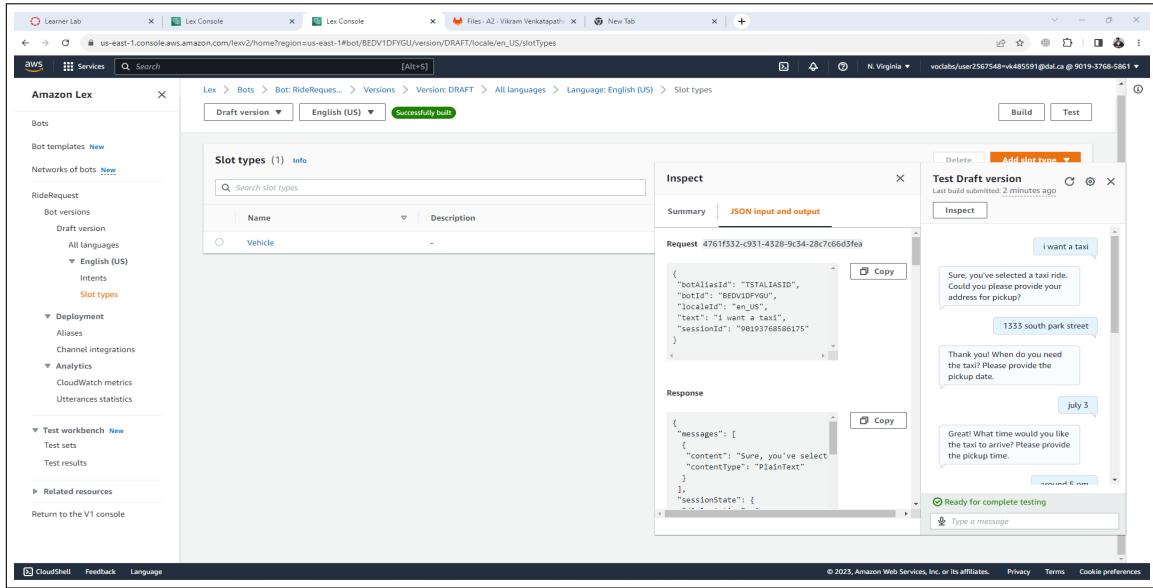


Figure 5.23.: *Taxi - Confirm request 1.1*

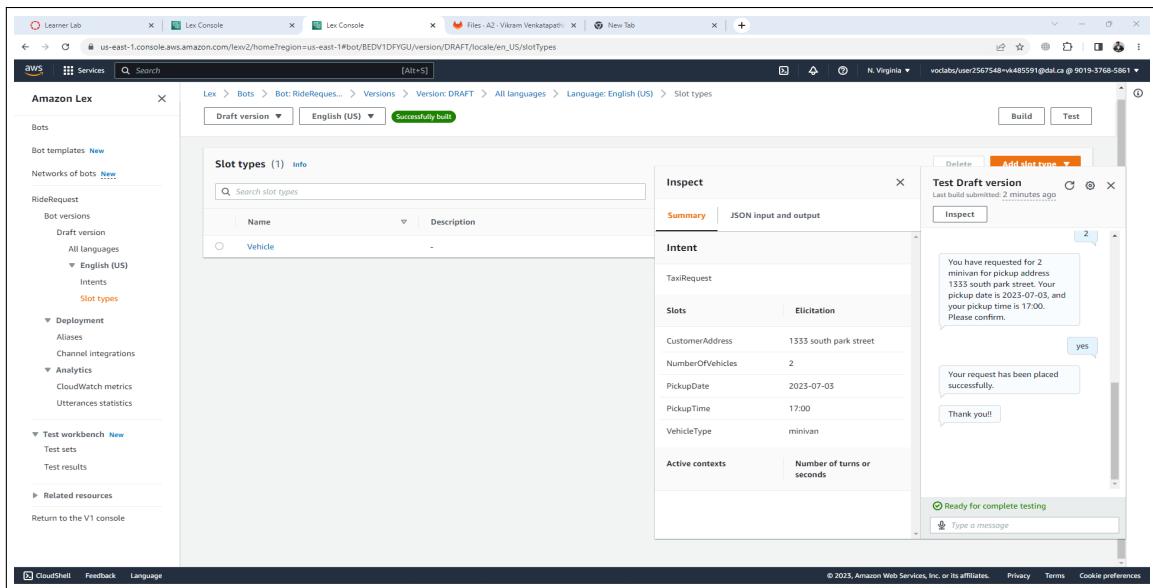


Figure 5.24.: *Taxi - Confirm request 1.2*

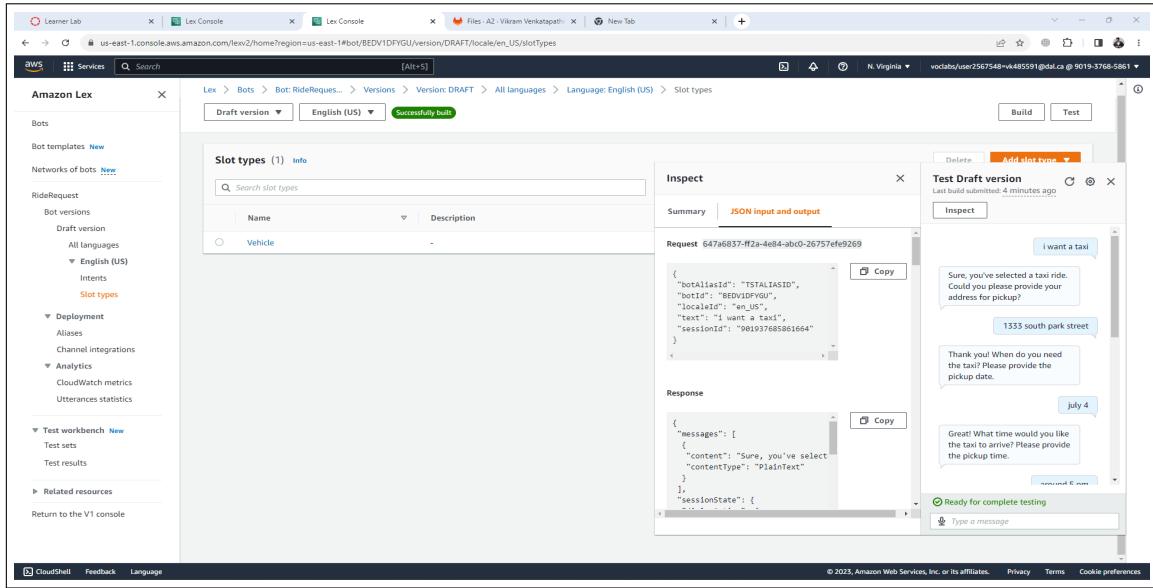


Figure 5.25.: *Taxi - Cancel request 1.1*

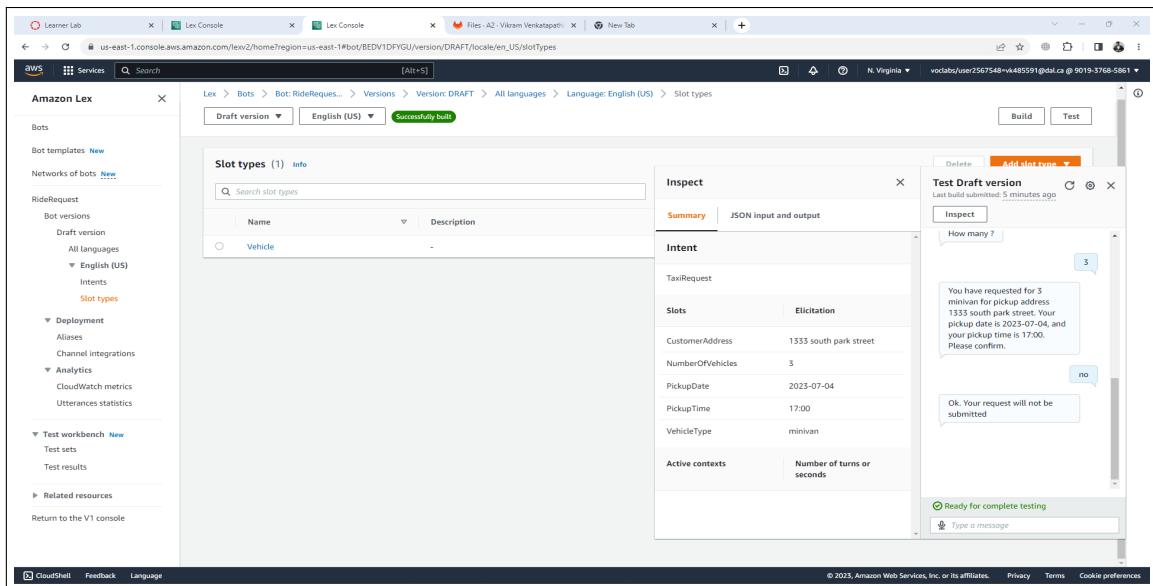


Figure 5.26.: *Taxi - Cancel request 1.2*

Part IV.

REFERENCES

References

- [1] N. Naik, "Performance Evaluation of Distributed Systems in Multiple Clouds using Docker Swarm," 2021 *IEEE International Systems Conference (SysCon)*, Vancouver, BC, Canada, 2021, pp. 1-6, doi: 10.1109/SysCon48628.2021.9447123
- [2] "Create standard repositories," *Google Cloud*. [Online]. Available: <https://cloud.google.com/artifact-registry/docs/repositories/create-repos>. [Accessed: 29 June 2023].
- [3] "Deploying to cloud run," *Google Cloud*. [Online]. Available: <https://cloud.google.com/run/docs/deploying>. [Accessed: 29 June 2023].
- [4] "Add data to cloud firestore," *Firebase*. [Online]. Available: <https://firebase.google.com/docs/firestore/manage-data/add-data>. [Accessed: 29 June 2023].
- [5] "Get data with cloud firestore," *Firebase*. [Online]. Available: <https://firebase.google.com/docs/firestore/query-data/get-data>. [Accessed: 29 June 2023].
- [6] "Containerize an application," *Docker Documentation*, 28-Jun-2023. [Online]. Available: https://docs.docker.com/get-started/02_our_app/. [Accessed: 29 June 2023].
- [7] "Update the application," *Docker Documentation*, 28-Jun-2023. [Online]. Available: https://docs.docker.com/get-started/03_updating_app/. [Accessed: 29 June 2023].
- [8] "Push and pull images," *Google Cloud*. [Online]. Available: <https://cloud.google.com/artifact-registry/docs/docker/pushing-and-pulling>. [Accessed: 29 June 2023].
- [9] "Create a new react app," *Reactjs.org*. [Online]. Available: <https://legacy.reactjs.org/docs/create-a-new-react-app.html>. [Accessed: 29 June 2023].
- [10] S.Gandotra, "ReactJS router," *GeeksforGeeks*, 13-Dec-2019. [Online]. Available: <https://www.geeksforgeeks.org/reactjs-router/>. [Accessed: 29 June 2023].
- [11] "LaTeX Listings package JSON formatting," *TeX Stack Exchange*, Available: <https://tex.stackexchange.com/questions/560830/latex-listings-package-json-formatting>. [Accessed 29 June 2023].
- [12] V. Venkatapathi, "B00936916_VikramVenkatapathi_A1_Report.pdf," *Dalhousie University*, Jan. 2023. [Online]. Available: https://git.cs.dal.ca/vikramv/csci5408_w23_b00936916_vikram_venkatapathi/-/blob/main/Assignment_1/B00936916_VikramVenkatapathi_A1_Report.pdf. [Accessed 29 June 2023].
- [13] Shanmuganathan, Vishakan. "OOAD-Project." *GitHub*, 2021, <https://github.com/svishakan/OOAD-Project>. [Accessed 29 June 2023].

- [14] AWS. "Amazon Lex V1", *Amazon.com* [Online]. Available: <https://docs.aws.amazon.com/lex/latest/dg/gs-bp-create-bot.html>. [Accessed: 29 June 2023].
- [15] "Swarm mode overview," *Docker Documentation*, 30-Jun-2023. [Online]. Available: <https://docs.docker.com/engine/swarm/>. [Accessed: 03 July 2023].
- [16] R. Powell, "Docker Swarm vs Kubernetes: how to choose a container orchestration tool," *CircleCI*, 12-Oct-2021. [Online]. Available: <https://circleci.com/blog/docker-swarm-vs-kubernetes/>. [Accessed: 03 July 2023].
- [17] "Production-Grade Container Orchestration," *Kubernetes*. [Online]. Available: <https://kubernetes.io/>. [Accessed: 03 July 2023].
- [18] "Lightweight kubernetes," *K3s.io*. [Online]. Available: <https://k3s.io/>. [Accessed: 03 July 2023].