

Contents

08.1a: API Gateway	4
8.1.1 MoTD function	4
8.1.2 API integration	4
8.1.3 Lambda code	4
8.1.4 Test code	4
8.1.5 Clean up	5
8.1.6 gettime API	5
8.1.7 Implement code	5
8.1.8 Test code	5
8.1.9 Clean up	5
8.2a: Lambda, API Gateway Guestbook	6
8.2.1 Overview	6
8.2.2 Obtain AWS account ID	6
8.2.3 REST API Code	6
8.2.4 Deploy the Lambda for viewing entries	6
8.2.5 Create API in API Gateway	6
8.2.6 Enable API to invoke Lambda function	6
8.2.7 API endpoint for viewing entries (1)	6
8.2.8 API endpoint for viewing entries (2)	6
8.2.9 CORS setup for viewing entries	6
8.2.10 Deploy API to production and view entries	6
8.2.11 API endpoint for signing (1)	7
8.2.12 API endpoint for signing (2)	7
8.2.13 CORS setup for signing	7
8.2.14 Deploy API to production and sign	7
8.2.15 Frontend Code	8
8.2.16 Configure and Deploy the Frontend	8
8.2.17 Clean up	9

8.2g: Cloud Functions, API Gateway Guestbook	9
8.2.1 Overview	9
8.2.2 Cloud Function backend (GET)	9
8.2.3 Cloud Function backend (POST)	9
8.2.4 Deploy the Cloud Functions	9
8.2.5 Test the Cloud Functions deployment	9
8.2.6 API Gateway	9
8.2.7 Create OpenAPI specification	9
8.2.8 –	9
8.2.9 Service account setup	9
8.2.10 Create the API Gateway	9
8.2.11 Test the API via Python Requests (GET)	9
8.2.12 Test the API via Python Requests (POST)	10
8.2.13 Client-side Guestbook application	10
8.2.14 guestbook.js	10
8.2.15 Version #1: Local file system	10
8.2.16 Version #2: Google Cloud Storage bucket	11
8.2.17 Clean up	12
8.3g OAuth2 Guestbook	12
8.3.1 OAuth2 Guestbook	13
8.3.2 OAuth2 Authorization Code flow	13
8.3.3 Checkout code	13
8.3.4 OAuth initiation code	13
8.3.5 Identity provider interaction	13
8.3.6 Callback code	13
8.3.7 –	13
8.3.8 Signing page	13
8.3.9 Model code	13
8.3.10 Build and deploy the code	13

8.3.11 Set up Identity Provider	13
8.3.12 –	13
8.3.13 Update deployment	13
8.3.14 Visit the application	13
8.3.15 Secrets manager deployment.....	17
8.3.16 Removing access	18
8.3.17 Clean up	18
8.4g: Firebase	18
8.4.1 Firebase web application	18
8.4.2 Project setup.....	18
8.4.3 Application setup	18
8.4.4 Authentication setup	18
8.4.5 Database setup.....	19
8.4.6 Storage setup.....	19
8.4.7 CLI setup	19
8.4.8 Bundling with Webpack	19
8.4.9 Configure Firebase within application	19
8.4.10 Initialize Firebase within application	19
8.4.11 Test application	19
8.4.12 Add authentication	20
8.4.13 Update UI	20
8.4.14 Test application with authentication	20
8.4.15 Add text messaging.....	20
8.4.16 Test application with text messaging	20
8.4.17 Manual message insertion	21
8.4.18 Add image messaging	22
8.4.19 Test application with image messaging.....	22
8.4.20 Deploy application	23

08.1a: API Gateway

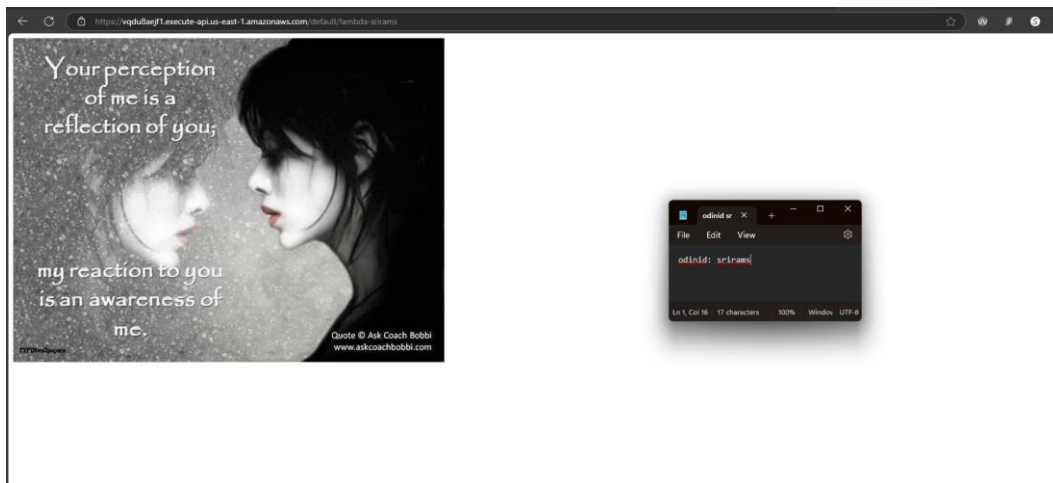
8.1.1 MoTD function

8.1.2 API integration

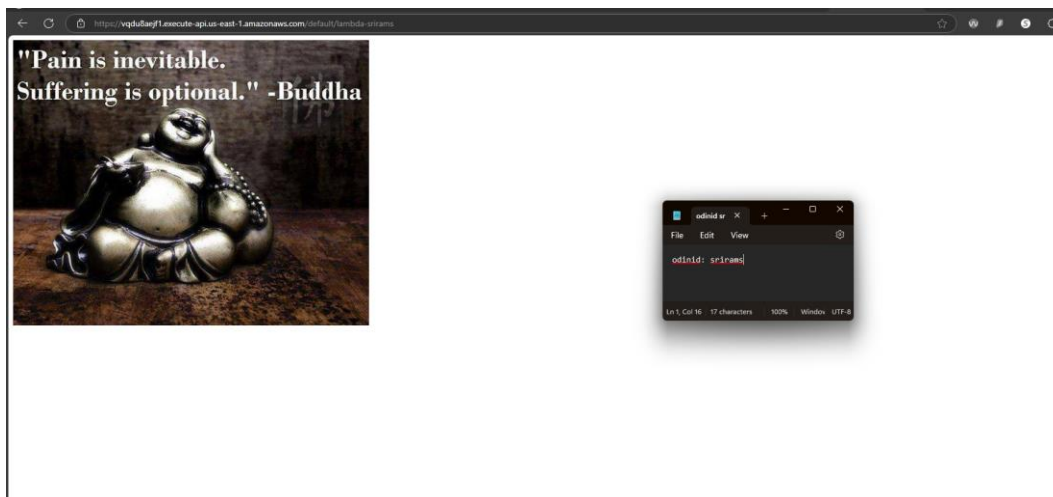
8.1.3 Lambda code

8.1.4 Test code

- Take a screenshot of the resulting page including the URL bar.



- Click "Reload" in the browser and take another screenshot showing the image has changed:



8.1.5 Clean up

8.1.6 gettime API

8.1.7 Implement code

8.1.8 Test code

- **Use curl on your Linux VM to access the API endpoint and show the results. Take a screenshot for your lab notebook.**

```
Last login: Thu Nov  7 08:17:58 2024 from 35.235.241.66
srirams@course-vm:~$ curl https://xcxdqxqkc4.execute-api.us-east-1.amazonaws.com/default/gettime-srirams
{"currentTime": "2024-11-24 01:46:21.030657"}srirams@course-vm:~$
```

8.1.9 Clean up

8.2a: Lambda, API Gateway Guestbook

8.2.1 Overview

8.2.2 Obtain AWS account ID

8.2.3 REST API Code

8.2.4 Deploy the Lambda for viewing entries

8.2.5 Create API in API Gateway

8.2.6 Enable API to invoke Lambda function

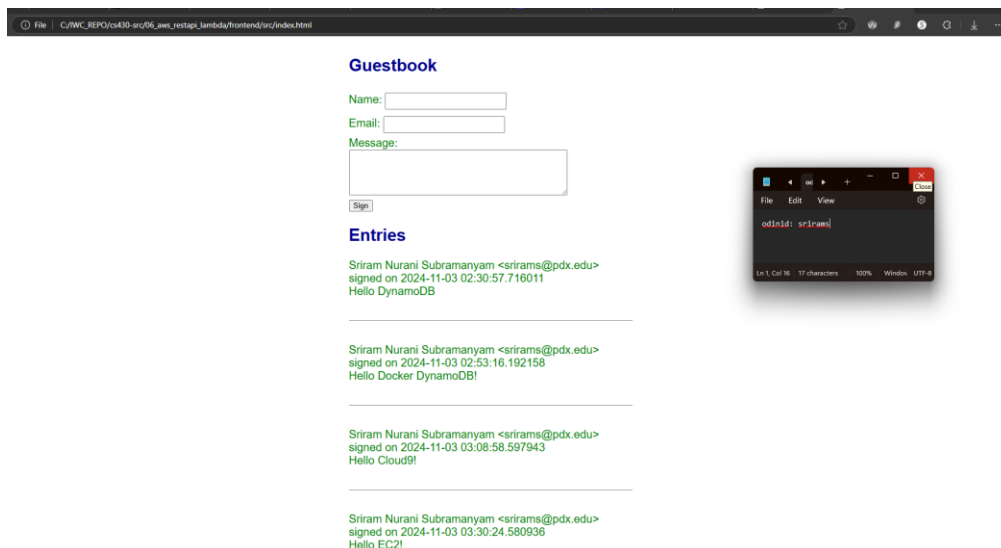
8.2.7 API endpoint for viewing entries (1)

8.2.8 API endpoint for viewing entries (2)

8.2.9 CORS setup for viewing entries

8.2.10 Deploy API to production and view entries

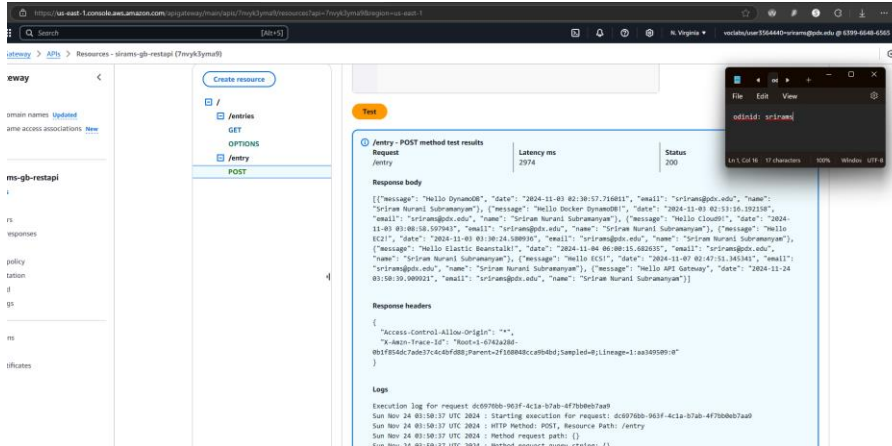
- **Take a screenshot that shows that you can view the entries in the backend database.**



8.2.11 API endpoint for signing (1)

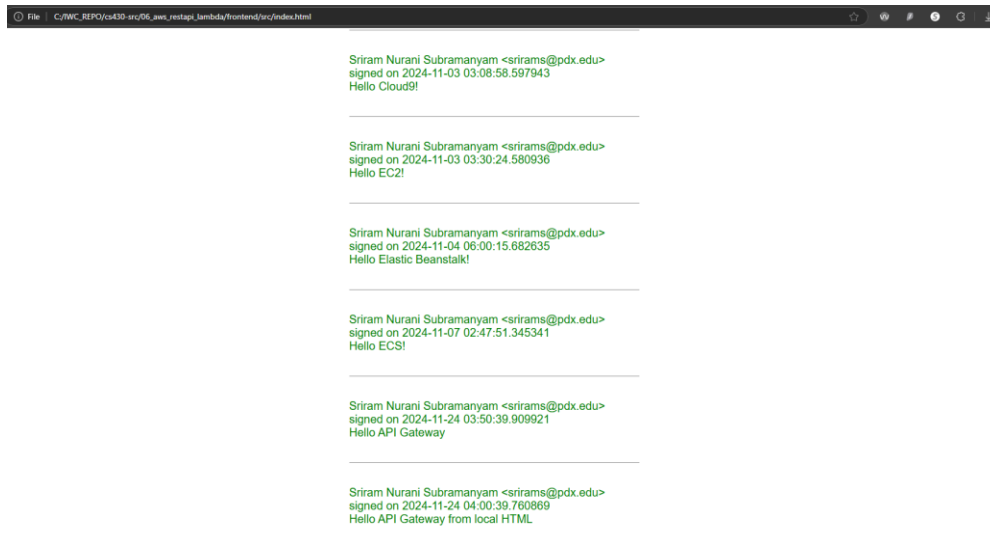
8.2.12 API endpoint for signing (2)

- Take a screenshot showing that the submission worked.



8.2.13 CORS setup for signing

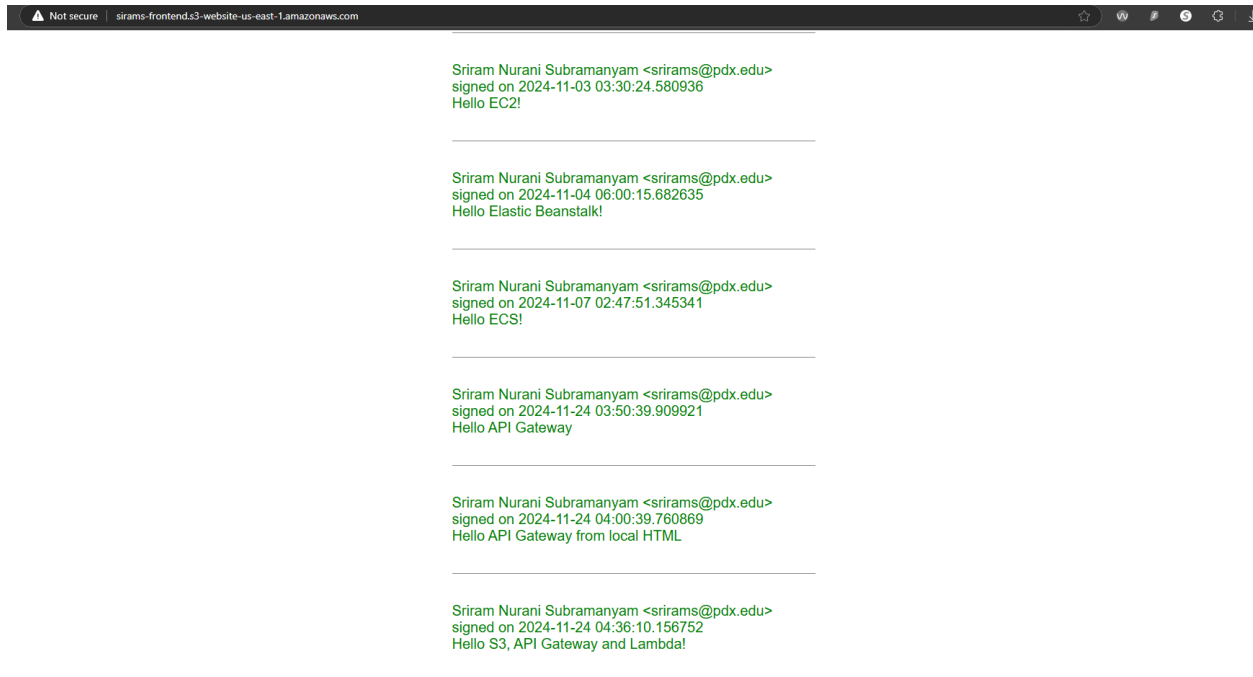
8.2.14 Deploy API to production and sign



8.2.15 Frontend Code

8.2.16 Configure and Deploy the Frontend

- **Take a screenshot as before that shows your entry and the static website hosting URL.**



8.2.17 Clean up

8.2g: Cloud Functions, API Gateway Guestbook

8.2.1 Overview

8.2.2 Cloud Function backend (GET)

8.2.3 Cloud Function backend (POST)

8.2.4 Deploy the Cloud Functions

8.2.5 Test the Cloud Functions deployment

8.2.6 API Gateway

8.2.7 Create OpenAPI specification

8.2.8 –

8.2.9 Service account setup

8.2.10 Create the API Gateway

- **Include the hostname for the API gateway in your lab notebook.**

gbapigw-ii2g3yo.uc.gateway.dev

```
srirama@cloudshell:~/cs430-src/06_gcp_restapi_cloudfunctions (cloud-murani-srirama)$ gcloud api-gateway gateways describe gbapigw --location=us-central1
apiConfig: projects/40274584080/locations/global/apis/gbapi/configs/gbapiconfig
createTime: '2024-11-25T00:13:20.844650252Z'
defaultHostname: gbapigw-ii2g3yo.uc.gateway.dev
displayName: gbapigw
name: projects/cloud-murani-srirama/locations/us-central1/gateways/gbapigw
state: ACTIVE
updateTime: '2024-11-25T00:16:23.890476216Z'
srirama@cloudshell:~/cs430-src/06_gcp_restapi_cloudfunctions (cloud-murani-srirama)$
```

8.2.11 Test the API via Python Requests (GET)

- **Take a screenshot of its output**

```

sriram@cloudshell:~/cs430-src/06_gcp_restapi/cloudfunctions (cloud-murari-sriram)$ python3
Python 3.12.3 (main, Nov 6 2024, 18:32:19) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>> resp = requests.get("https://ghapiw-l2g3yo.uc.gateway.dev/entries")
>>> print("Response HTTP status_code ", resp.status_code)
Response HTTP status_code: 200
>>> print("Response Headers ", resp.headers)
Response Headers: {'content-type': 'application/json', 'access-control-allow-origin': '*', 'x-cloud-trace-context': 'f069cae0134ebdf52d33a3b0994b041e0=1', 'alt-svc': 'h3="443"; ma=2592000,h3-29="443"; ma=2592000', 'Date': 'Mon, 25 Nov 2024 00:42:02 GMT', 'Server': 'Google Frontend', 'Content-Length': '372'}
>>> print("Response text ", resp.text)
Response text: [{"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 07:00:19.313303+00:00", "message": "Hello App Engine!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:13:45.944729+00:00", "message": "Hello Kubernetes!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 09:03:05.539926+00:00", "message": "Hello Cloud Run!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:48:22.616911+00:00", "message": "Hello Cloud Build!"}]
>>> print("Response represented as a JSON object ", resp.json())
Response represented as a JSON object: [{"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 07:00:19.313303+00:00", "message": "Hello App Engine!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:13:45.944729+00:00", "message": "Hello Kubernetes!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 09:03:05.539926+00:00", "message": "Hello Cloud Run!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:48:22.616911+00:00", "message": "Hello Cloud Build!"}]
>>> data = resp.json()
>>> if isinstance(data, list) and len(data) > 0:
...     first_entry = data[0]
...     print("Name:", first_entry.get("name", "N/A"))
...     print("Email:", first_entry.get("email", "N/A"))
...     print("Date:", first_entry.get("date", "N/A"))
...     print("Message:", first_entry.get("message", "N/A"))
... else:
...     print("No entries found in the response.")
>>>
Name: Sriram Murari Subramanyam
Email: sriram@pdx.edu
Date: 2024-11-04 07:00:19.313303+00:00
Message: Hello App Engine!

```

8.2.12 Test the API via Python Requests (POST)

- Take a screenshot of the output for your lab notebook

```

sriram@cloudshell:~/cs430-src/06_gcp_restapi/cloudfunctions (cloud-murari-sriram)$ python3
Python 3.12.3 (main, Nov 6 2024, 18:32:19) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import json
>>> mydict = {
...     'name': 'Sriram Murari Subramanyam',
...     'email': 'sriram@pdx.edu',
...     'message': 'Hello Cloud Functions from Python Requests!'
... }
>>> mydict
{'name': 'Sriram Murari Subramanyam', 'email': 'sriram@pdx.edu', 'message': 'Hello Cloud Functions from Python Requests!'}
>>> resp = requests.post("https://ghapiw-l2g3yo.uc.gateway.dev/entry", json=mydict)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'requests' is not defined
>>> import requests
>>> resp = requests.post("https://ghapiw-l2g3yo.uc.gateway.dev/entry", json=mydict)
>>> print("Response HTTP status_code ", resp.status_code)
Response HTTP status_code: 200
>>> print("Response Headers ", resp.headers)
Response Headers: {'content-type': 'application/json', 'access-control-allow-origin': '*', 'x-cloud-trace-context': 'addded66f3c20bd18b7750c3048c7505d1e0=1', 'alt-svc': 'h3="443"; ma=2592000,h3-29="443"; ma=2592000', 'Date': 'Mon, 25 Nov 2024 00:52:13 GMT', 'Server': 'Google Frontend', 'Content-Length': '741'}
>>> print("Response text ", resp.text)
Response text: [{"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 07:00:19.313303+00:00", "message": "Hello App Engine!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:13:45.944729+00:00", "message": "Hello Kubernetes!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 09:03:05.539926+00:00", "message": "Hello Cloud Run!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:48:22.616911+00:00", "message": "Hello Cloud Build!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-25 00:52:13.403207+00:00", "message": "Hello Cloud Functions from Python Requests!"}]
>>> print("Response represented as a JSON object ", resp.json())
Response represented as a JSON object: [{"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 07:00:19.313303+00:00", "message": "Hello App Engine!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:13:45.944729+00:00", "message": "Hello Kubernetes!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-04 09:03:05.539926+00:00", "message": "Hello Cloud Run!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-18 10:48:22.616911+00:00", "message": "Hello Cloud Build!"}, {"name": "Sriram Murari Subramanyam", "email": "sriram@pdx.edu", "date": "2024-11-25 00:52:13.403207+00:00", "message": "Hello Cloud Functions from Python Requests!"}]
>>>

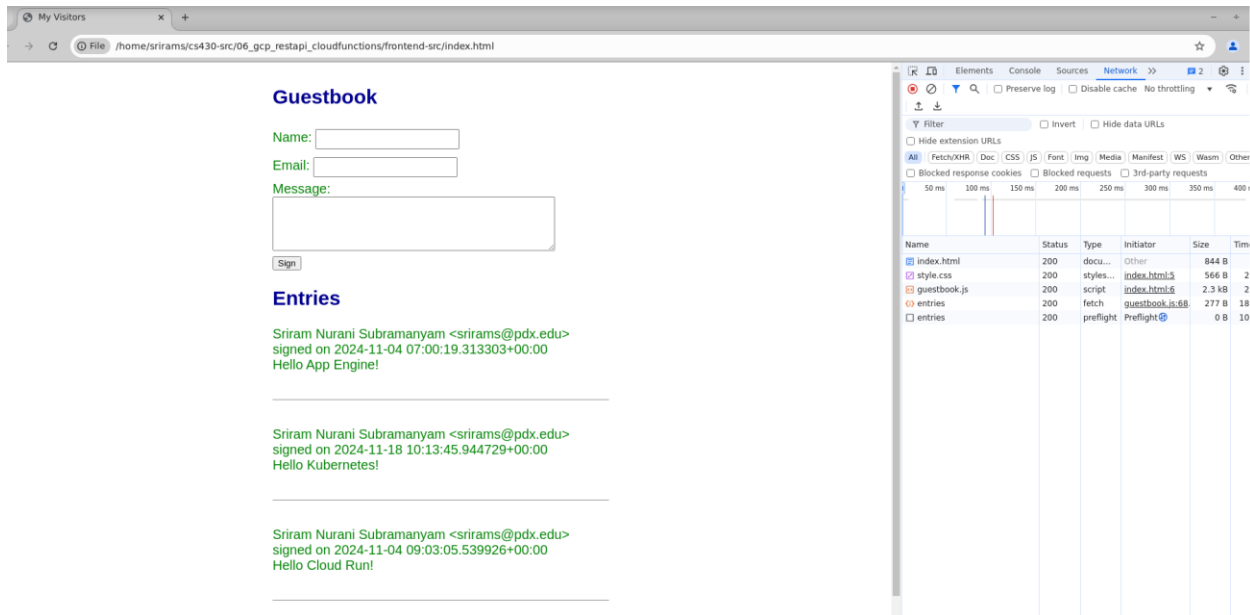
```

8.2.13 Client-side Guestbook application

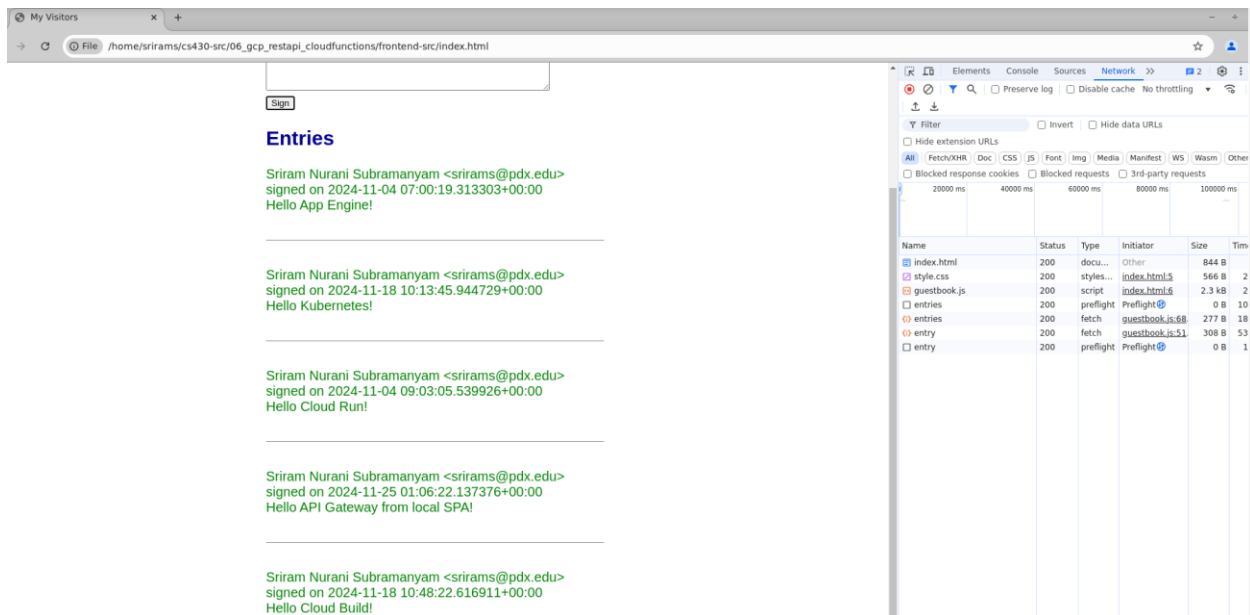
8.2.14 guestbook.js

8.2.15 Version #1: Local file system

- Take a screenshot of the Guestbook including the URL.



- Take a screenshot of the Guestbook including the URL.



8.2.16 Version #2: Google Cloud Storage bucket

Take a screenshot of the Guestbook including the URL



Guestbook

Name:

Email: 

Message:

Entries

Sriram Nurani Subramanyam <srirams@pdx.edu>
signed on 2024-11-25 01:13:33.042853+00:00
Hello API Gateway from SPA in GCS!

Sriram Nurani Subramanyam <srirams@pdx.edu>
signed on 2024-11-04 07:00:19.313303+00:00
Hello App Engine!

Sriram Nurani Subramanyam <srirams@pdx.edu>
signed on 2024-11-18 10:13:45.944729+00:00
Hello Kubernetes!

8.2.17 Clean up

8.3g OAuth2 Guestbook

8.3.1 OAuth2 Guestbook

8.3.2 OAuth2 Authorization Code flow

8.3.3 Checkout code

8.3.4 OAuth initiation code

8.3.5 Identity provider interaction

8.3.6 Callback code

8.3.7 –

8.3.8 Signing page

8.3.9 Model code

8.3.10 Build and deploy the code

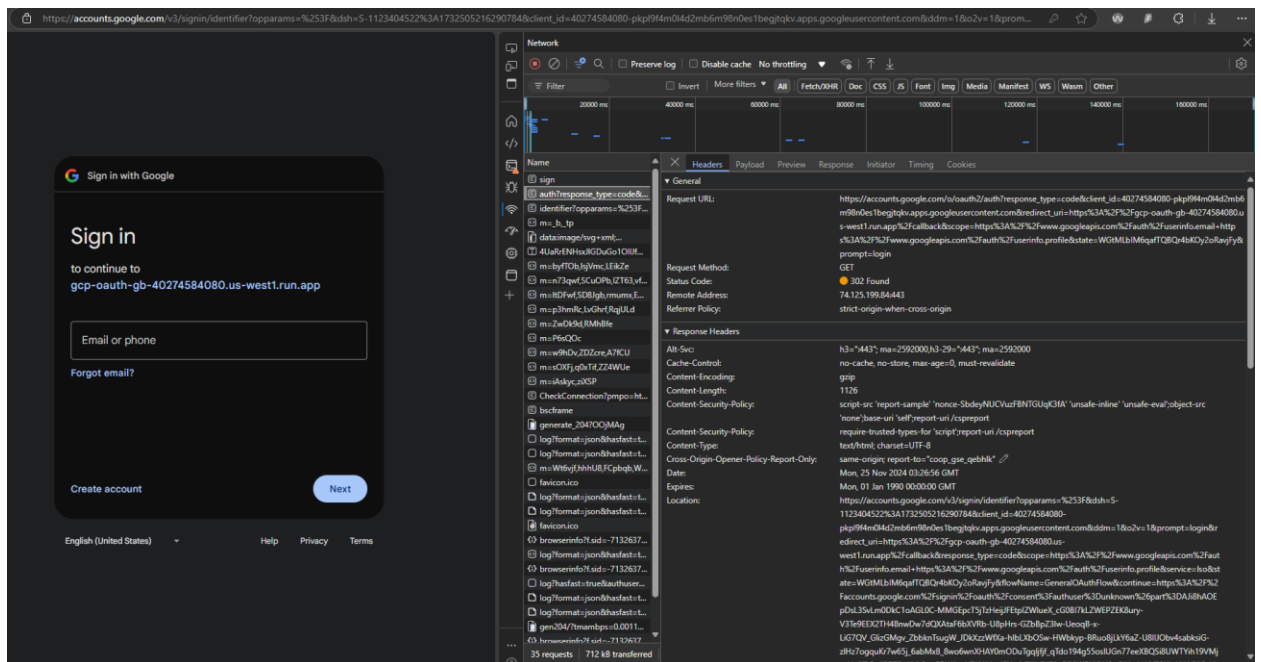
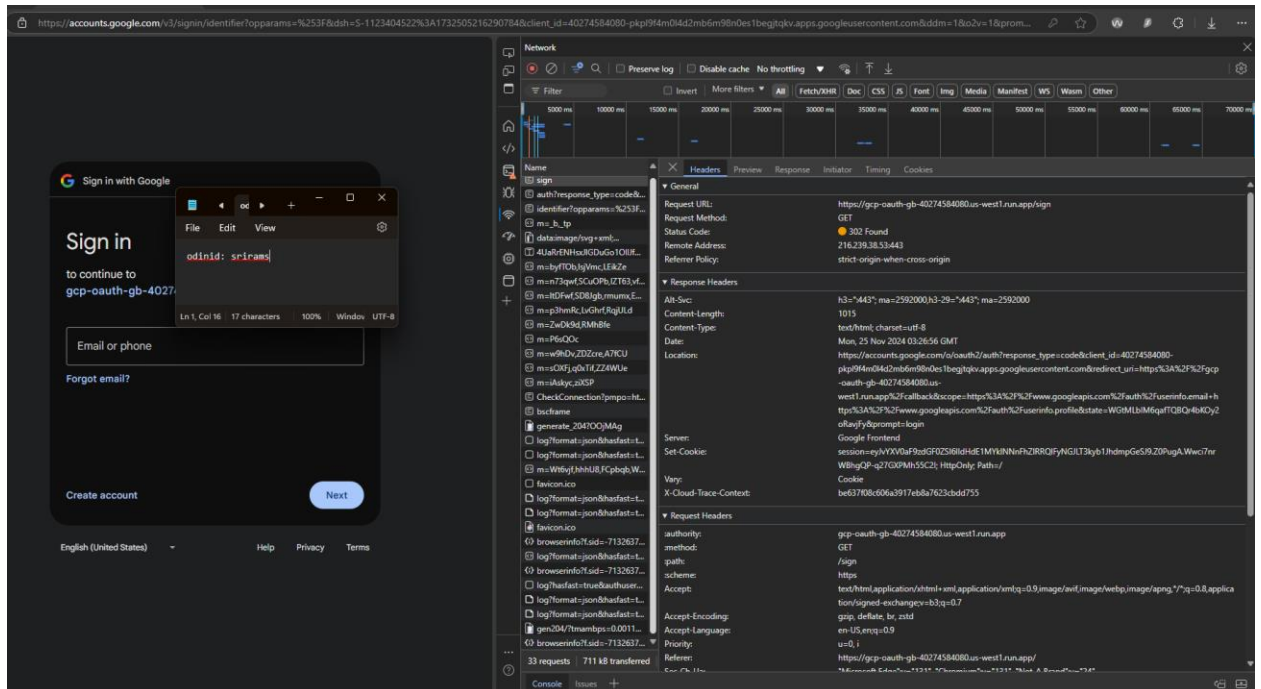
8.3.11 Set up Identity Provider

8.3.12 –

8.3.13 Update deployment

8.3.14 Visit the application

- **Take a screenshot of the Headers that includes the URL and the returned HTTP status code for the first two requests for your lab notebook.**



- Based on the description of the source code, what lines of code in our application are responsible for the second request shown?

```

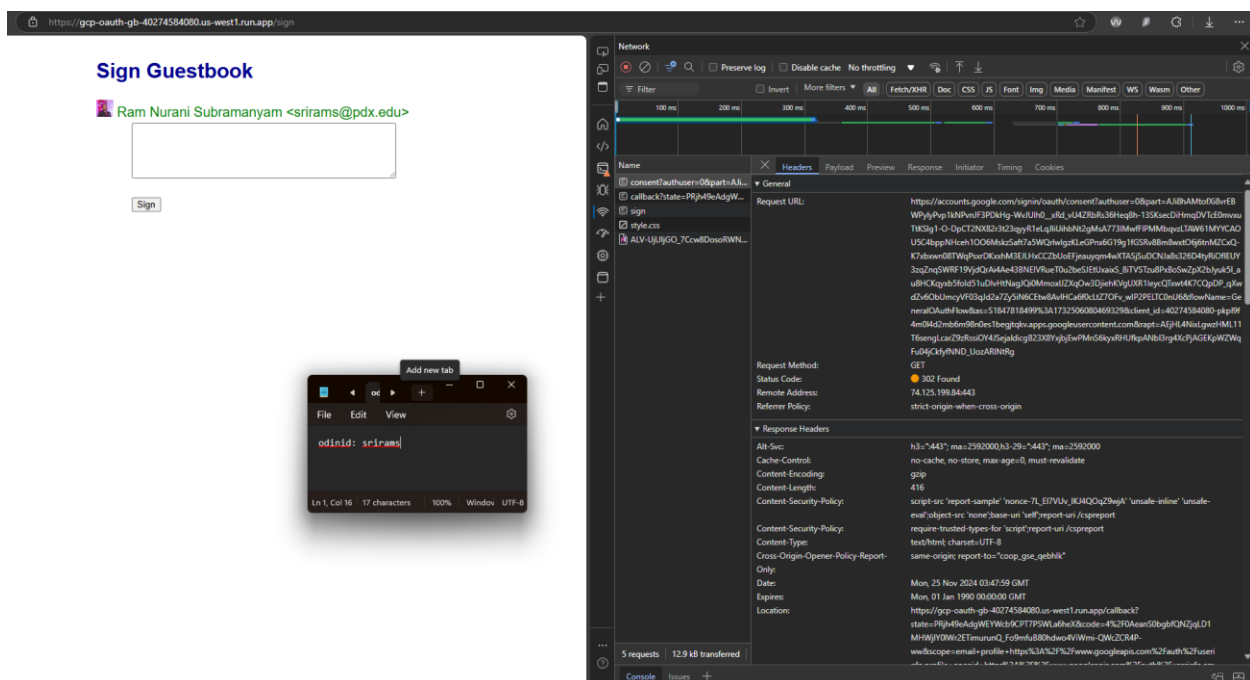
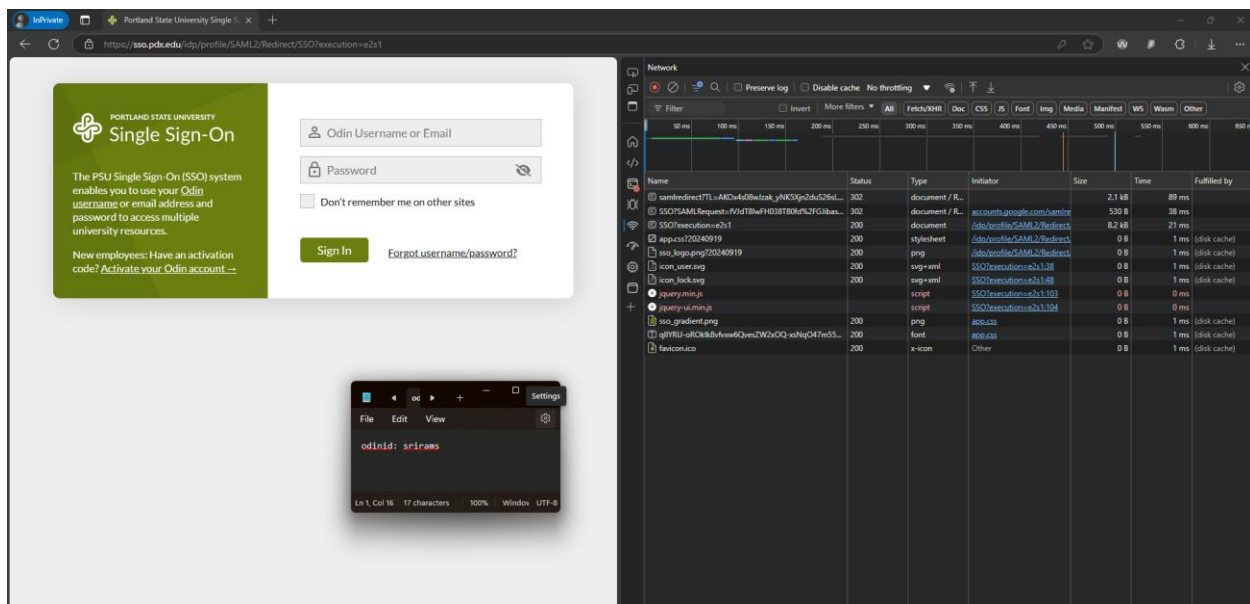
return render_template('sign.html', name=userinfo['name'], email=userinfo['email'], picture=userinfo['picture'])
else:
    # Redirect to the identity provider and ask the identity provider to return the client
    # back to /callback route with the code
    google = OAuth2Session(client_id,
        redirect_uri = redirect_callback,
        scope = 'https://www.googleapis.com/auth/userinfo.email ' +
            'https://www.googleapis.com/auth/userinfo.profile'
    )
    authorization_url, state = google.authorization_url(authorization_base_url, prompt='login')

    # Identity provider returns URL and random "state" that must be echoed later
    # to prevent CSRF.
    session['oauth_state'] = state
    return redirect(authorization_url)

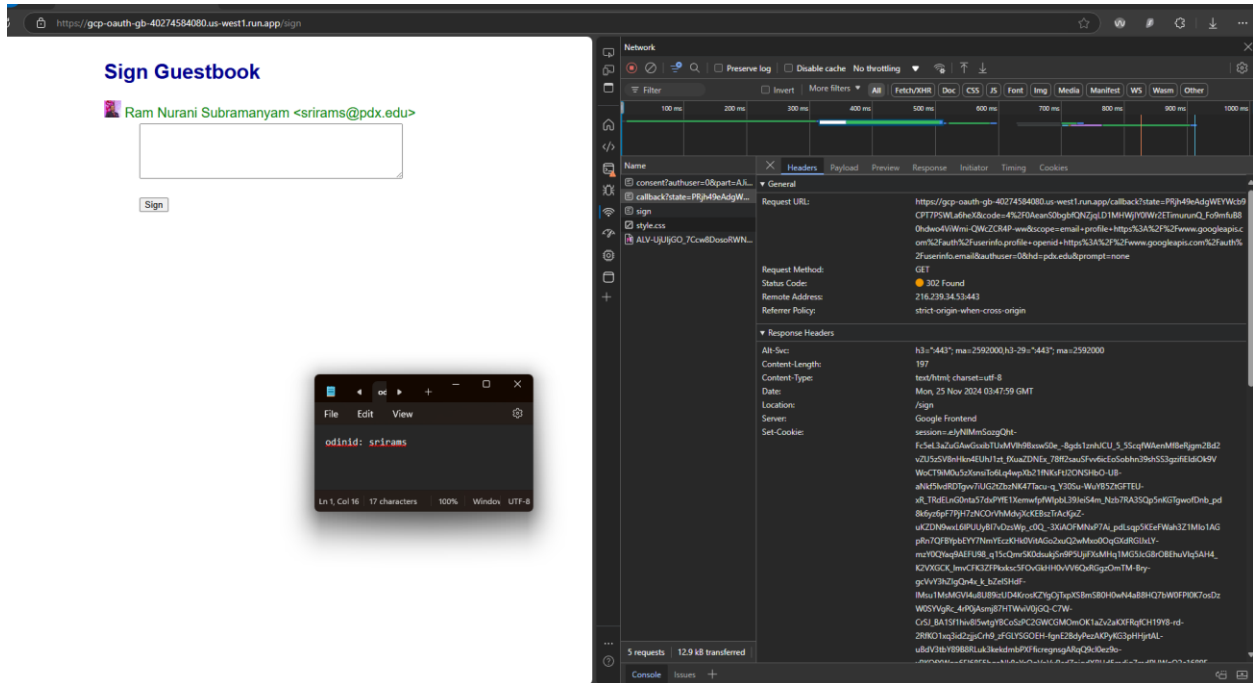
def post(self):
    """
    Accepts POST requests and processes the form
    """

```

- Take a screenshot of the permissions you (as a user) are granting the Guestbook access to.

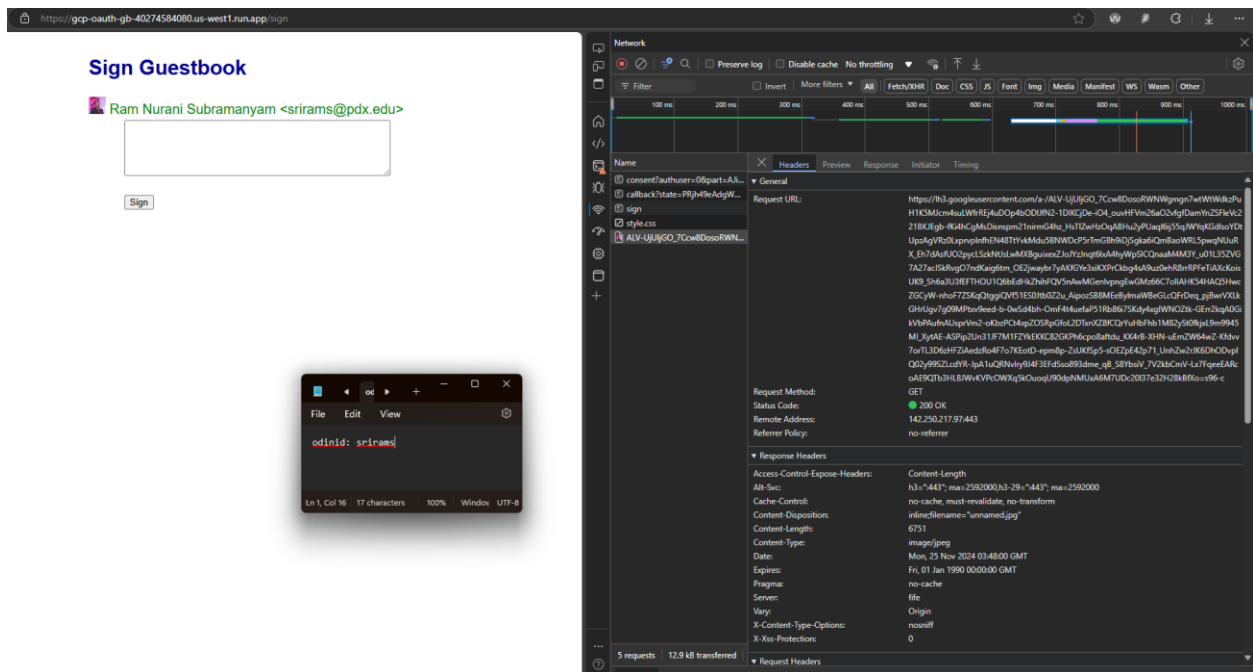


- Take a screenshot of the Headers that includes the entire Callback URL and its returned HTTP status code. What is the URI for the Location that the User sent to by the Callback?



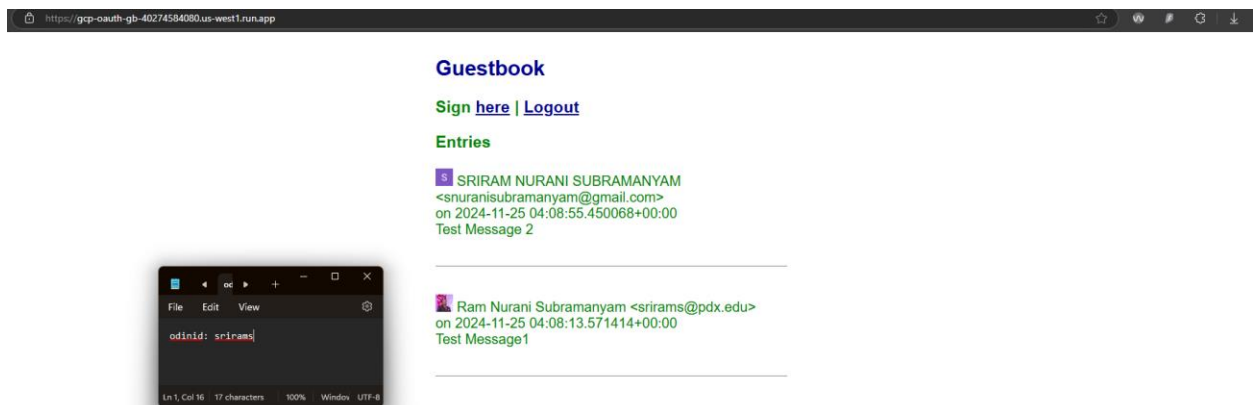
Location: /sign

- Find the request within Developer Tools that fetches the embedded image and take a screenshot of its URL.



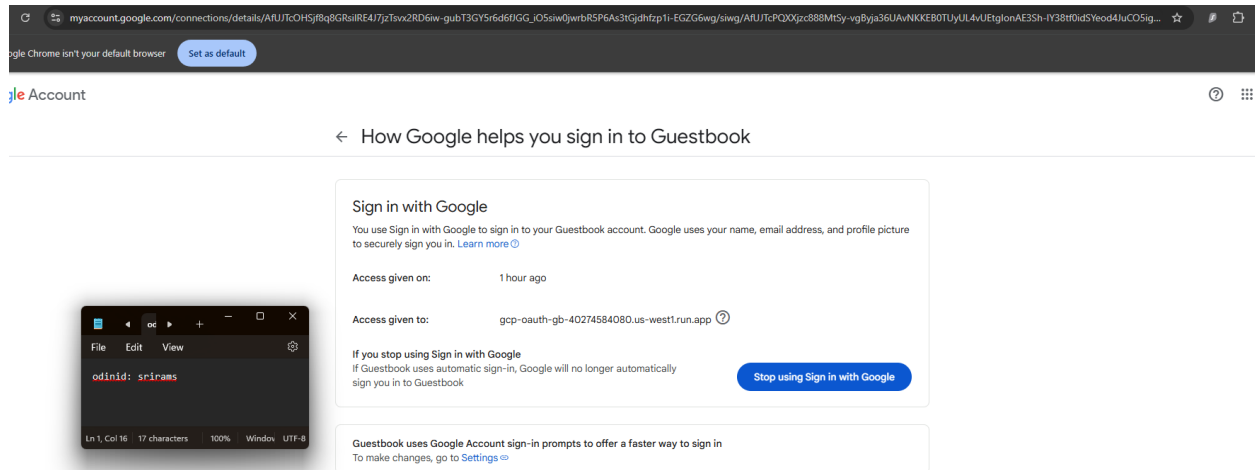
8.3.15 Secrets manager deployment

- Take a screenshot showing multiple authenticated accounts have been able to sign the Guestbook.



8.3.16 Removing access

- **Take a screenshot of the expanded information that includes your OdindId for your lab notebook.**



8.3.17 Clean up

8.4g: Firebase

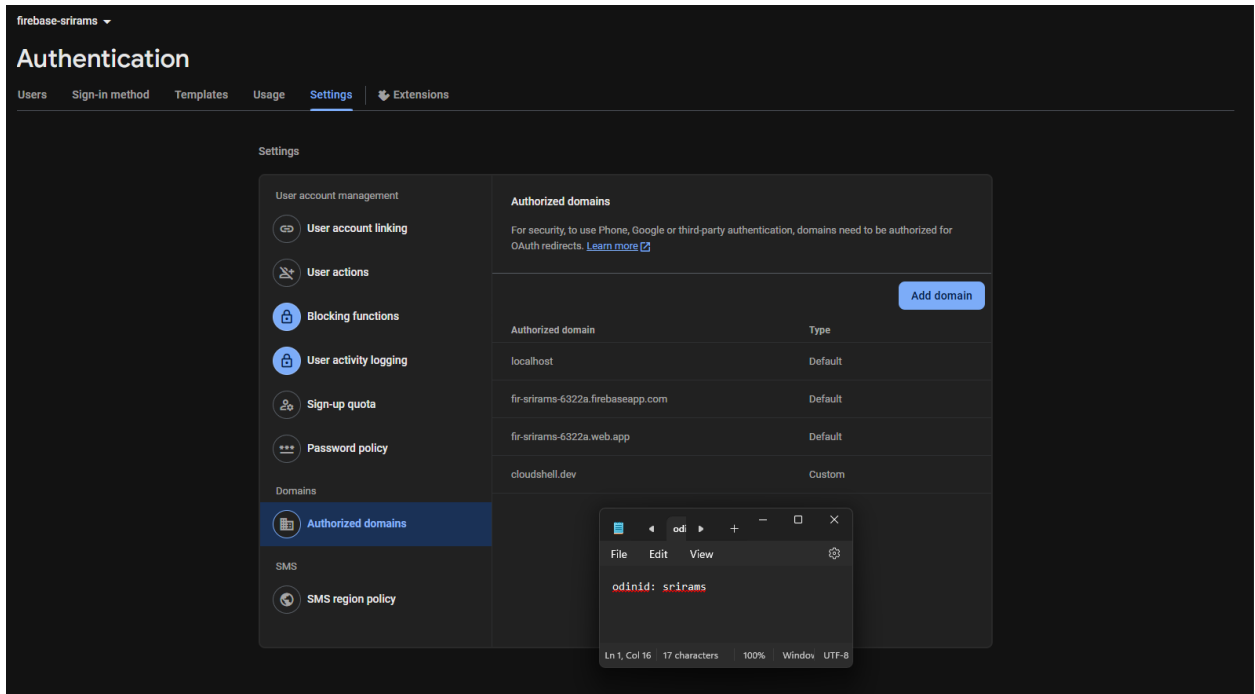
8.4.1 Firebase web application

8.4.2 Project setup

8.4.3 Application setup

8.4.4 Authentication setup

- **What other domains are given access to this Firebase project by default?**



8.4.5 Database setup

8.4.6 Storage setup

8.4.7 CLI setup

8.4.8 Bundling with Webpack

- Take a screenshot of the first 10 lines of the produced file.

```
srirams@cloudshell:~/codelab-friendlychat-web/web-start/public/scripts (firebase-srirams)$ head -10 main.js
/*
 * ATTENTION: An "eval-source-map" devtool has been used.
 * This devtool is neither made for production nor for readable output files.
 * It uses "eval()" calls to create a separate source file with attached SourceMaps in the browser devtools.
 * If you are trying to read the output file, select a different devtool (https://webpack.js.org/configuration/devtool/)
 * or disable the default devtool with "devtool: false".
 * If you are looking for production-ready output files, see mode: "production" (https://webpack.js.org/configuration/mode/).
 */
/******/ (() => { // webpackBootstrap
/******/     "use strict";
srirams@cloudshell:~/codelab-friendlychat-web/web-start/public/scripts (firebase-srirams)$
```

8.4.9 Configure Firebase within application

8.4.10 Initialize Firebase within application

8.4.11 Test application

8.4.12 Add authentication

- **What missing functions deal with user authentication?**

`signIn()`

- **What missing functions deal with sending and receiving messages?**

`saveMessage(messageText)`

`loadMessages()`

8.4.13 Update UI

- **What are the names of the elements that are hidden when the user is signed out?**

`userNameElement`

`userPicElement`

`signOutButtonElement`

- **What is the name of the element that is not hidden when the user is signed out?**

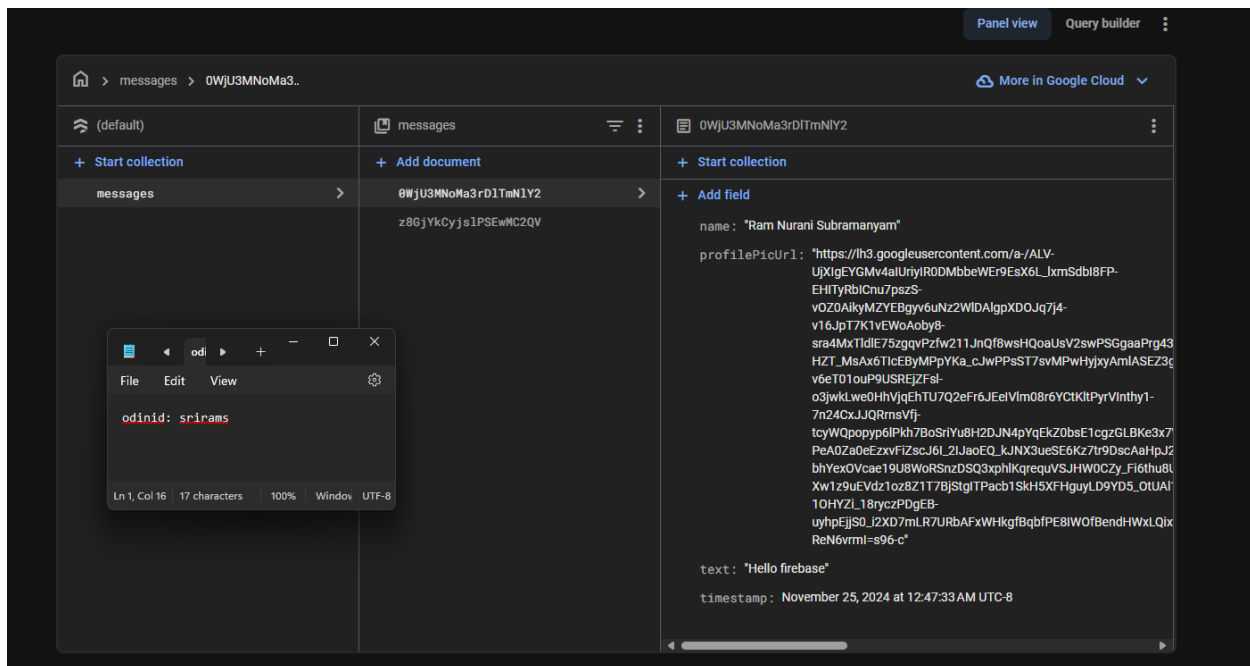
`signInButtonElement`

8.4.14 Test application with authentication

8.4.15 Add text messaging

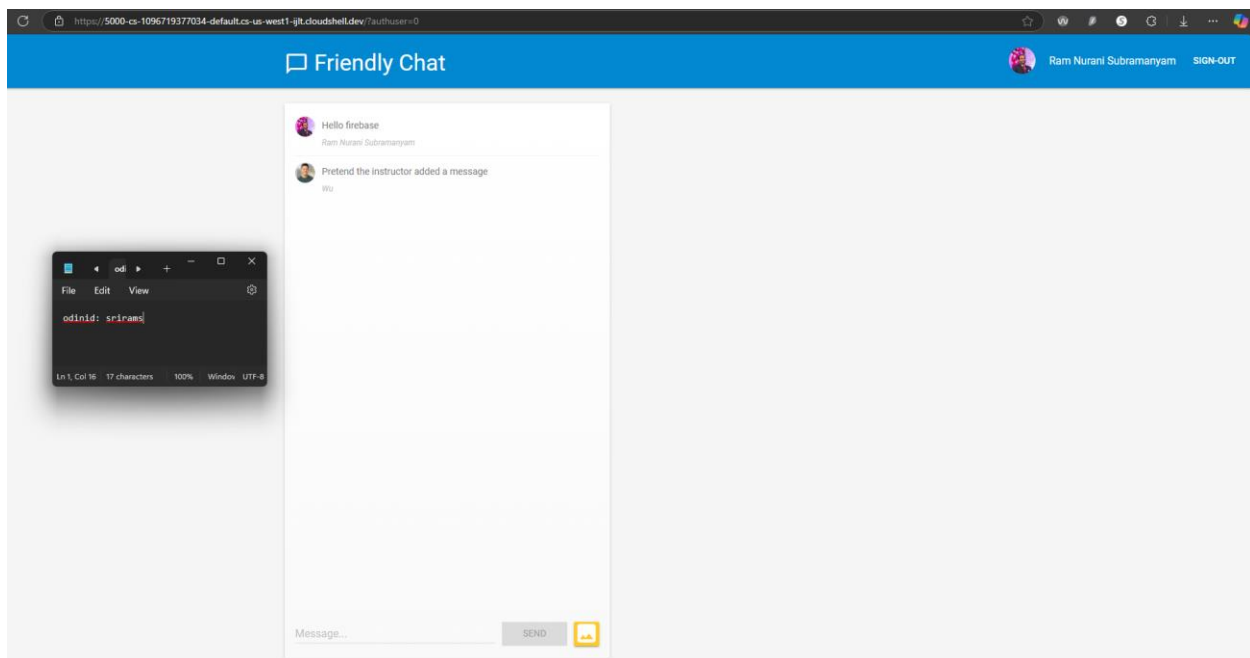
8.4.16 Test application with text messaging

- **Include a screenshot of the message and its fields in the database for your lab notebook**



8.4.17 Manual message insertion

- Include a screenshot of the application with its two messages for your lab notebook



8.4.18 Add image messaging

- What is the URL of the image that is first shown in the UI as the message is loading?

<https://www.google.com/images/spin-32.gif?a>

8.4.19 Test application with image messaging

- How do the fields in an image document differ from that of the text document?

Text Document Fields

Name, profilePicUrl, text, timestamp

Text Document Fields

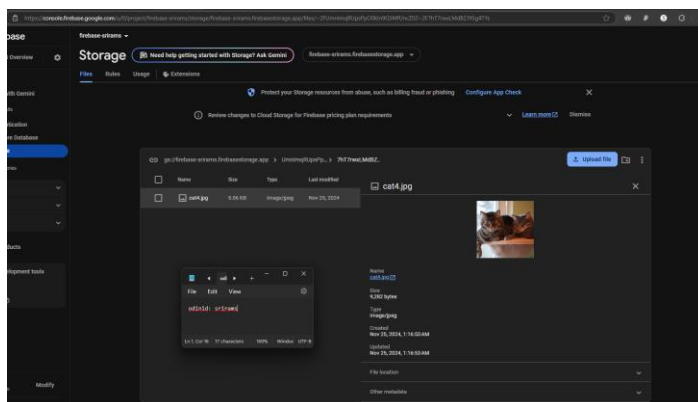
imageUrl, name, profilePicUrl, storageUri, timestamp

- What URL and storage location can the image be found at?

<https://firebasestorage.googleapis.com/v0/b/firebase-srirams.firebaseio.com/o/UmnImqRUpsPpO0kIn9G5MfUncZ02%2F7hT7rwxLMdBZJ9Sg4TYs%2Fcat4.jpg?alt=media&token=1b43841a-c633-493d-9891-ef2cc207bcf5>

StorageURI is UmnImqRUpsPpO0kIn9G5MfUncZ02/7hT7rwxLMdBZJ9Sg4TYs/cat4.jpg

- Take a screenshot of the image in the storage bucket for your lab notebook.



8.4.20 Deploy application

- What directory is the application going to be served from?

Public

- Take a screenshot of the message including the URL for your lab notebook.

```
srirams@cloudshell:~/codelab-friendlychat-web/web-start (firebase-srirams)$ firebase deploy --except functions
(node:4595) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)

== Deploying to 'firebase-srirams'...

i deploying hosting
i hosting[fir-srirams-6322a]: beginning deploy...
i hosting[fir-srirams-6322a]: found 8 files in ./public
✓ hosting[fir-srirams-6322a]: file upload complete
i hosting[fir-srirams-6322a]: finalizing version...
✓ hosting[fir-srirams-6322a]: version finalized
i hosting[fir-srirams-6322a]: releasing new version...
✓ hosting[fir-srirams-6322a]: release complete

✓ Deploy complete!

Project Console: https://console.firebase.google.com/project/firebase-srirams/overview
Hosting URL: https://fir-srirams-6322a.web.app
srirams@cloudshell:~/codelab-friendlychat-web/web-start (firebase-srirams)$
```

