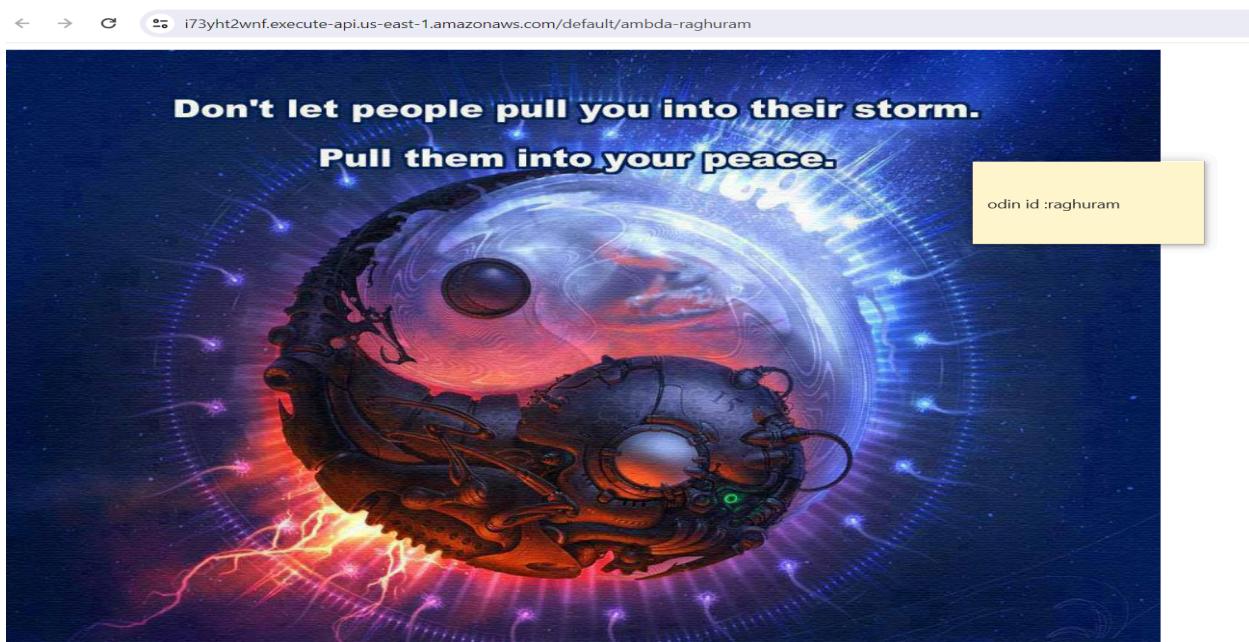


08.1a: API Gateway.....	2
4. Test code.....	3
08.2a: Lambda, API Gateway Guestbook.....	5
10. Deploy API to production and view entries.....	5
12. API endpoint for signing (2).....	5
16. Configure and Deploy the Frontend.....	6
08.2g: Cloud Functions, API Gateway Guestbook.....	7
10. Create the API Gateway.....	7
11. Test the API via Python Requests (GET).....	7
12. Test the API via Python Requests (POST).....	9
15. Version #1: Local file system.....	9
16. Version #2: Google Cloud Storage bucket.....	10
08.3g OAuth2 Guestbook.....	11
14. Visit the application.....	11
15. Secrets manager deployment.....	16
16. Removing access.....	17
08.4g: Firebase.....	17
4. Authentication setup.....	17
8. Bundling with Webpack.....	18
12. Add authentication.....	18
13. Update UI.....	19
• What is the name of the element that is not hidden when the user is signed out?.....	19
16. Test application with text messaging.....	19
17. Manual message insertion.....	20
19. Test application with image messaging.....	21
20. Deploy application.....	22

08.1a: API Gateway

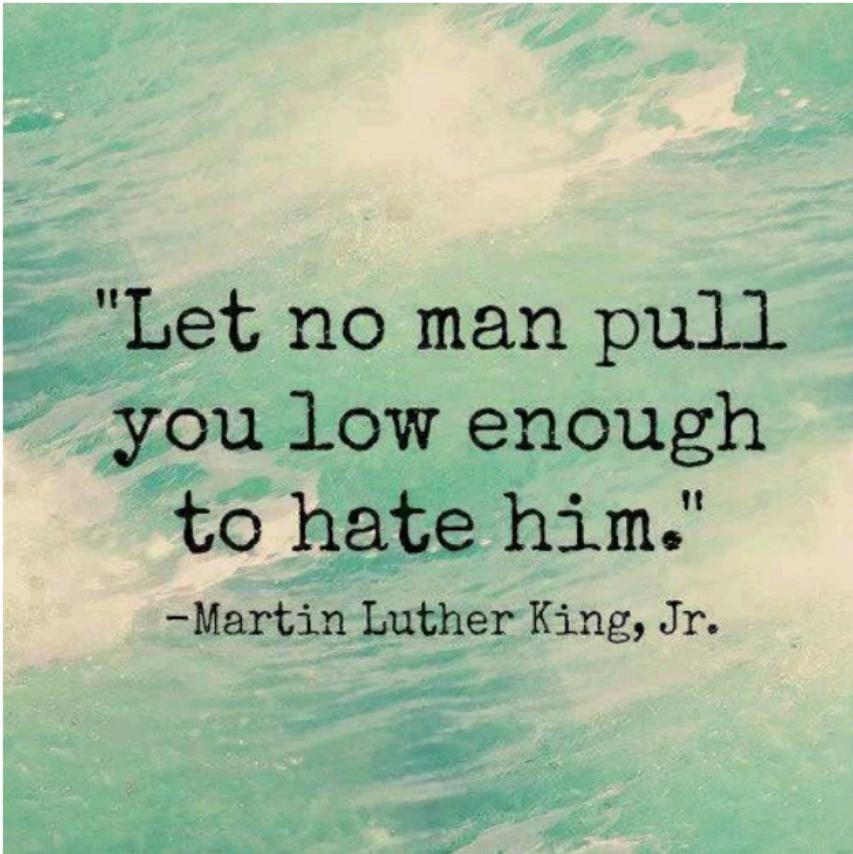
4. Test code

- 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:

← → ⌂ i73yht2wnf.execute-api.us-east-1.amazonaws.com/default/ambda-raghuram



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

```
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
Last login: Sat May 18 07:20:10 2024 from 35.235.240.145  
raghuram@course-vm:~$ curl https://v16psy3vh5.execute-api.us-east-1.amazonaws.com/default/gettime-raghuram  
{"current_time": "2024-05-25 02:51:12.538293"}raghuram@course-vm:~$ █
```

08.2a: Lambda, API Gateway Guestbook

10. Deploy API to production and view entries

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

The screenshot shows a web browser window with the title bar reading "File C:/Users/Raghuram/cs430-src/06_aws_restapi_lambda/frontend/src/index.html". The main content area displays a "Guestbook" form with fields for Name and Email, and a large Message input area. Below the form is a section titled "Entries" showing two signed messages from "Raghuram <raghuram@pdx>". A yellow sidebar on the right contains the text "odin id :raghuram".

Guestbook

Name:

Email:

Message:

Sign

Entries

Raghuram <raghuram@pdx>
signed on 2024-05-06 20:23:11.254499
Hello Cloud9!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-06 19:20:10.789282
Hello DynamoDB

+ ... ×
odin id :raghuram

12. API endpoint for signing (2)

- Take a screenshot showing that the submission worked.

The screenshot shows the AWS Lambda API Gateway console. On the left, the API structure is displayed with a tree view. Under the root path `/`, there is a resource named `/entries` with methods `GET` and `OPTIONS`. Below it is another resource named `/entry` with methods `POST` and `PUT`. A yellow callout box highlights the `POST` method for `/entry` and its body placeholder `body id :raghuram`. On the right, a detailed view of a recent request is shown. The request URL is `/entry`, the status is 200, latency is 3234 ms, and the response body contains a JSON array of messages. One message in the array is: {"message": "Hello Cloud9!", "date": "2024-05-06 20:23:11.254499", "email": "raghuram@pdx", "name": "Raghuram"}, {"message": "Hello DynamoDB", "date": "2024-05-06 19:20:10.789282", "email": "raghuram@pdx.edu", "name": "Raghuram"}, {"message": "Hello Docker DynamoDB", "date": "2024-05-06 19:34:01.798381", "email": "raghuram@pdx.edu", "name": "Raghuram"}, {"message": "Hello EC2!", "date": "2024-05-07 02:19:51.558369", "email": "raghuram@pdx.edu", "name": "Raghuram"}, {"message": "Hello Elastic Beanstalk!", "date": "2024-05-12 02:28:37.423526", "email": "raghuram@pdx.edu", "name": "Raghuram"}, {"message": "Hello ECS!", "date": "2024-05-13 21:21:55.270441", "email": "raghuram@pdx.edu", "name": "Raghuram"}, {"message": "Hello API Gateway", "date": "2024-05-26 23:38:37.017527", "email": "raghuram@pdx.edu", "name": "Raghuram"}]. The response headers include "Access-Control-Allow-Origin": "*" and "X-Amzn-Trace-Id": "Root=1-6653c7fa-721799e8319d786923ecfc2f;Parent=3b523b46b3e9a729;Sampled=0;lineage=897a86e9:0".

16. Configure and Deploy the Frontend

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

Raghuram <raghuram@pdx.edu>
signed on 2024-05-07 02:19:51.558369
Hello EC2!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-12 02:28:37.423526
Hello Elastic Beanstalk!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-13 21:21:55.270441
Hello ECS!

raghuram <raghuram@pdx.edu>
signed on 2024-05-26 23:38:37.017527
Hello API Gateway

Raghuram <raghuram@pdx.edu>
signed on 2024-05-27 00:35:41.976533
Hello S3, API Gateway and Lambda!

08.2g: Cloud Functions, API Gateway Guestbook

10. Create the API Gateway

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

```
raghuram@cloudshell:~/cs430-src/06_gcp_restapi_cloudfunctions (cloud-nataraja-raghuram)$ gcloud api-gateway gateways describe gbapigw --location=us-central1
apiConfig: projects/781452701321/locations/global/apis/gbapi/configs/gbapiconfig
createTime: '2024-05-27T01:32:39.533433673Z'
defaultHostname: gbapigw-9yzsrfd5.uc.gateway.dev
displayName: gbapigw
name: projects/cloud-nataraja-raghuram/locations/us-central1/gateways/gbapigw
state: ACTIVE
updateTime: '2024-05-27T01:36:09.660199379Z'
raghuram@cloudshell:~/cs430-src/06_gcp_restapi_cloudfunctions (cloud-nataraja-raghuram)$
```

11. Test the API via Python Requests (GET)

- Take a screenshot of its output

```

raghuram@cloudshell:/cs430-src/06_gcp_restapi_cloudfunctions (cloud-nataraaja-raghuram)$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> resp = requests.get("https://us-central1-cloud-nataraaja-raghuram.cloudfunctions.net/entries")
>>> print("Response HTTP status code:", resp.status_code)
Response HTTP status code: 200
>>> print(resp.headers)
{'Content-Type': 'application/json', 'Access-Control-Allow-Origin': '*', 'Function-Execution-Id': 'w2fs6cogiyfb4', 'X-Cloud-Trace-Context': '29a6369c8be4e8a6332650fd8b7da888;o=1', 'Date': 'Mon, 21 May 2024 03:44:19 GMT', 'Server': 'Google Frontend', 'Content-Length': '1032'}
>>> print(resp.text)
[{"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 04:04:49.085611+00:00", "message": "Hello Cloud Shell!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 03:09:44.206532+00:00", "message": "Hello Datastore"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-13 21:58:34.463629+00:00", "message": "Hello Cloud Run!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-21 05:32:12.532942+00:00", "message": "Hello Kubernetes!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 04:16:31.984718+00:00", "message": "Hello Compute Engine!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 03:44:19.226271+00:00", "message": "Hello Docker Datastore"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-12 02:53:02.174625+00:00", "message": "Hello App Engine!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-21 06:23:04.438771+00:00", "message": "Hello Cloud Build!"}]

KeyboardInterrupt
>>> for entry in data:
...     print("name:", entry["name"])
...     print("email:", entry["email"])
...     print("date:", entry["date"])
...     print("message:", entry["message"])
...     print("")  # Print an empty line after each entry
...
name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-07 04:04:49.085611+00:00
message: Hello Cloud Shell!

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-07 03:09:44.206532+00:00
message: Hello Datastore

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-13 21:58:34.463629+00:00
message: Hello Cloud Run!

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-21 05:32:12.532942+00:00
message: Hello Kubernetes!

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-07 04:16:31.984718+00:00
message: Hello Compute Engine!

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-07 03:44:19.226271+00:00
message: Hello Docker Datastore

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-12 02:53:02.174625+00:00
message: Hello App Engine!

name: Raghuram
email: raghuram@pdx.edu
date: 2024-05-21 06:23:04.438771+00:00
message: Hello Cloud Build!

```

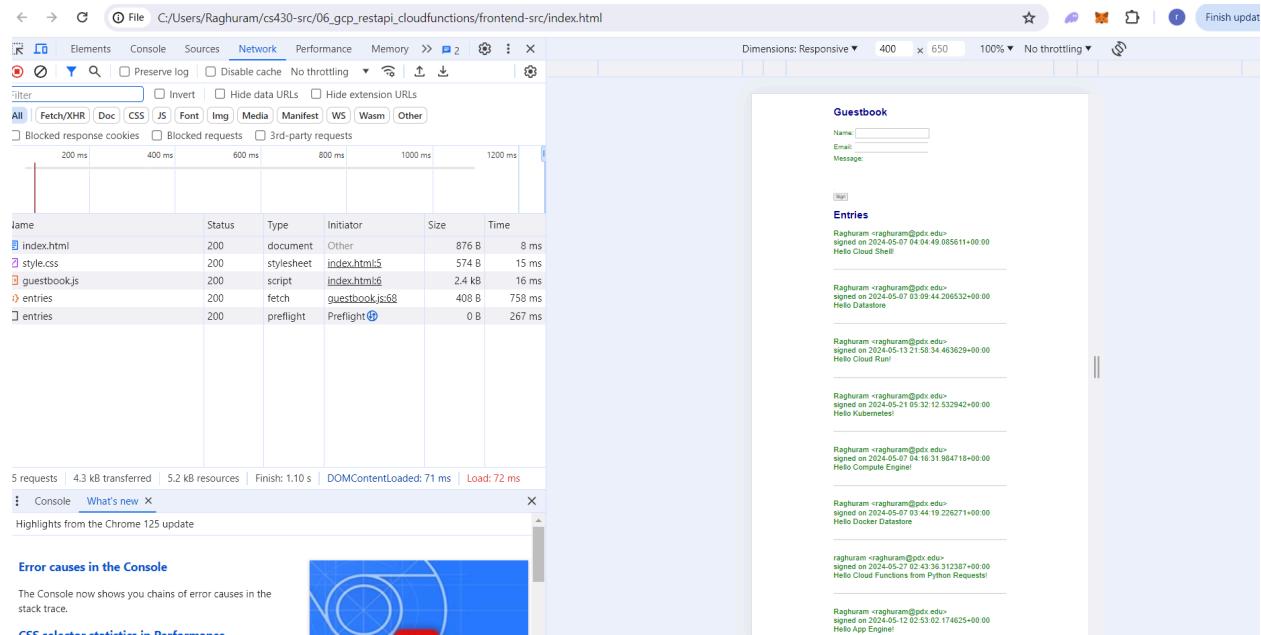
12. Test the API via Python Requests (POST)

- Take a screenshot of the output for your lab notebook

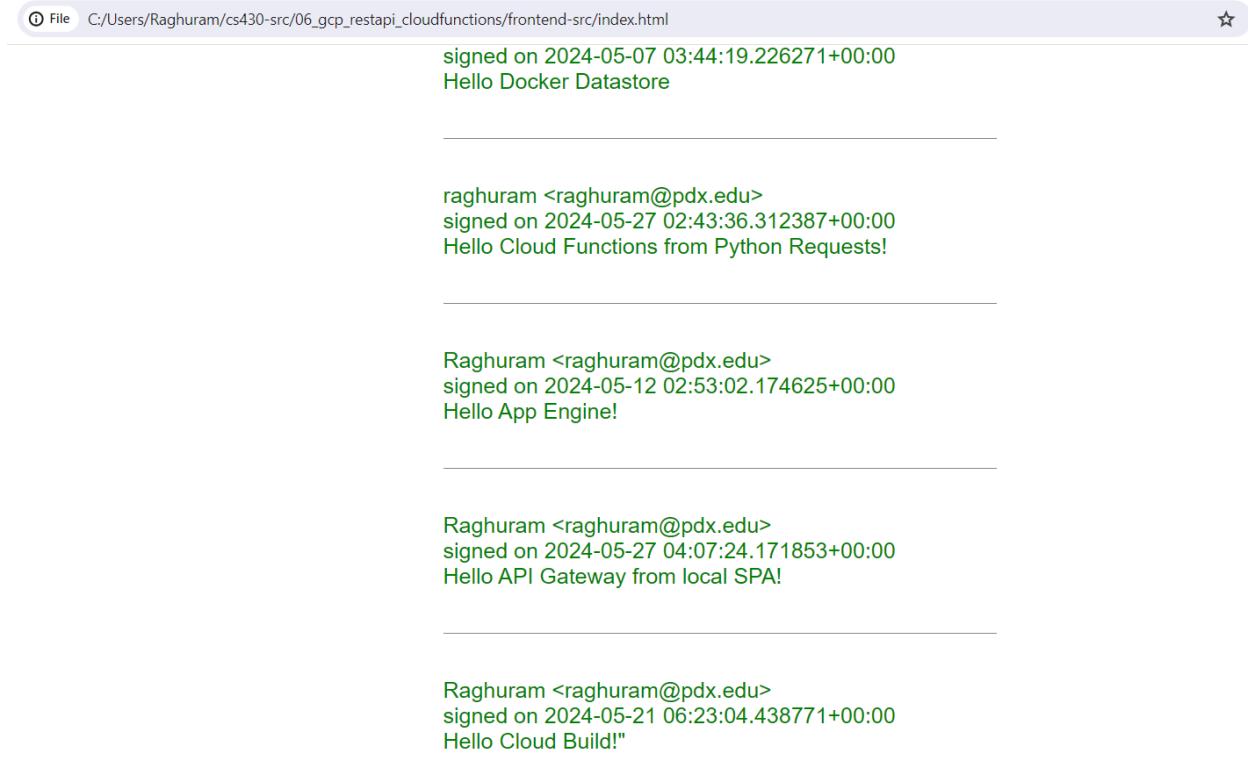
```
>>> print(resp.status_code)
200
>>> print(resp.headers)
{'Content-Type': 'application/json', 'Access-Control-Allow-Origin': '*', 'Function-Execution-Id': 'x2ptyi9ogbim', 'X-Cloud-Trace-Context': '9d179405376e29dc64d77b1883f09853;c=1', 'Date': 'Mon, 2 May 2024 02:43:36 GMT', 'Server': 'Google Frontend', 'Content-Length': '1185'}
>>> print(resp.text)
[{"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 04:04:49.085611+00:00", "message": "Hello Cloud Shell!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 04:44:20.206532+00:00", "message": "Hello Datastore"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-13 21:58:34.463629+00:00", "message": "Hello Cloud Run!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 03:44:19.226271+00:00", "message": "Hello Kubernetes!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-07 03:44:19.226271+00:00", "message": "Hello Docker Datastore"}, {"name": "raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-27 02:43:36.312387+00:00", "message": "Hello Cloud Functions from Python Requests!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-12 02:53:02.174625+00:00", "message": "Hello App Engine!"}, {"name": "Raghuram ", "email": "raghuram@pdx.edu", "date": "2024-05-21 06:23:04.438771+00:00", "message": "Hello Cloud Build!"}]
```

15. Version #1: Local file system

- Take a screenshot showing the preflight request to the API that allows API access, as well as the subsequent fetch request have been successful.



- Take a screenshot of the Guestbook including the URL.



The screenshot shows a web browser displaying a guestbook page. The URL in the address bar is `C:/Users/Raghuram/cs430-src/06_gcp_restapi_cloudfunctions/frontend-src/index.html`. The page content consists of five entries, each with a timestamp, an email address, and a message. The entries are separated by horizontal lines.

signed on 2024-05-07 03:44:19.226271+00:00
Hello Docker Datastore

raghuram <raghuram@pdx.edu>
signed on 2024-05-27 02:43:36.312387+00:00
Hello Cloud Functions from Python Requests!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-12 02:53:02.174625+00:00
Hello App Engine!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-27 04:07:24.171853+00:00
Hello API Gateway from local SPA!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-21 06:23:04.438771+00:00
Hello Cloud Build!"

16. Version #2: Google Cloud Storage bucket

- Take a screenshot of the Guestbook including the URL.

storage.googleapis.com/gbapi-raghuram/index.html

☆

GUESTBOOK

Name:

Email:

Message:
Hello API Gateway from SPA in GCS!

Entries

Raghuram <raghuram@pdx.edu>
signed on 2024-05-07 04:04:49.085611+00:00
Hello Cloud Shell!

Raghuram <raghuram@pdx.edu>
signed on 2024-05-07 03:09:44.206532+00:00
Hello Datastore

Raghuram <raghuram@pdx.edu>
signed on 2024-05-27 04:36:12.166279+00:00
Hello API Gateway from SPA in GCS!

08.3g OAuth2 Guestbook

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.

Screenshot of the Chrome DevTools Network tab showing a request to `https://gcp-oauth-gb-mfvgnqrx-a.run.app/sign`. The request method is GET, status code is 302 Found, and the response URL is `https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=781452701321-r8s8i4bl450139iuf8kcb9oa275r0mv.apps.googleusercontent.com&redirect_uri`.

The Headers panel shows the following details:

- General**
 - Request URL: `https://gcp-oauth-gb-mfvgnqrx-a.run.app/sign`
 - Request Method: GET
 - Status Code: 302 Found
 - Remote Address: [2001:4860:4802:36::35]:443
 - Referrer Policy: strict-origin-when-cross-origin
- Response Headers**
 - Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
 - Content-Length: 1007
 - Content-Type: text/html; charset=utf-8
 - Date: Mon, 27 May 2024 20:58:30 GMT
 - Location: `https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=781452701321-r8s8i4bl450139iuf8kcb9oa275r0mv.apps.googleusercontent.com&redirect_uri`

The Network tab also displays a timeline of requests, with the current request highlighted in yellow.

Error causes in the Console
The Console now shows you chains of error causes in the stack trace.

CSS selector statistics in Performance
The Performance panel can now show you CSS selector statistics for long-running Recalculate Style



Screenshot of the Chrome Network tab showing a request to accounts.google.com. The Headers section is selected, displaying the following details:

- Request URL:** https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=781452701321-r8s8ji4bl450139juf8kcp9oq275r0mv.apps.googleusercontent.com&redirect_ur...
ri=https%3A%2F%2Fgcp-oauth-gb-mfgvgnqxa-uw.a.run.app%2Fcallback&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile&state=6m1qzlKMMVkdMC7gpaqPU891Awkzwm&prompt=login
- Request Method:** GET
- Status Code:** 302 Found
- Remote Address:** [2607:f8b0:400e:c0d::54]:443
- Referrer Policy:** strict-origin-when-cross-origin

The Headers tab also shows other response headers like Alt-Svc and Cache-Control.

Error causes in the Console

The Console now shows you chains of error causes in the stack trace.

CSS selector statistics in Performance

The Performance panel can now show you CSS selector statistics for long-running Recalculate Style



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

```
google = OAuth2Session(
```

```
    client_id,
```

```
    redirect_uri="http://localhost:8000/callback",
```

```
    state=session["oauth_state"],
```

```
)
```

```
token = google.fetch_token()
```

```
token_url, client_secret=client_secret, authorization_response=request.url
```

```
)
```

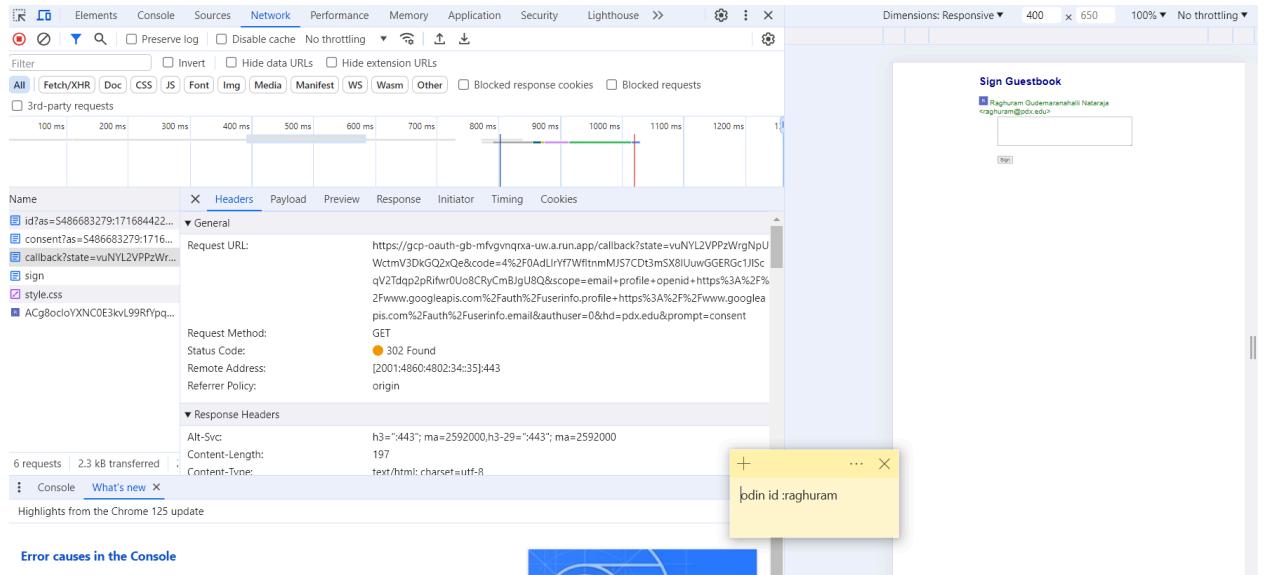
```
session["oauth_token"] = token
```

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

The screenshot shows the Chrome DevTools Network tab with a list of requests. The requests include files like samlredirect, SSO?execution=e1s1, app.css, icon_user.svg, and sso_gradient.png. The timeline shows the duration of each request. To the right, a "Single Sign-On" dialog from Portland State University is displayed, asking for "Odin Username or Email" and "Password". A yellow callout box points to the "Sign In" button.

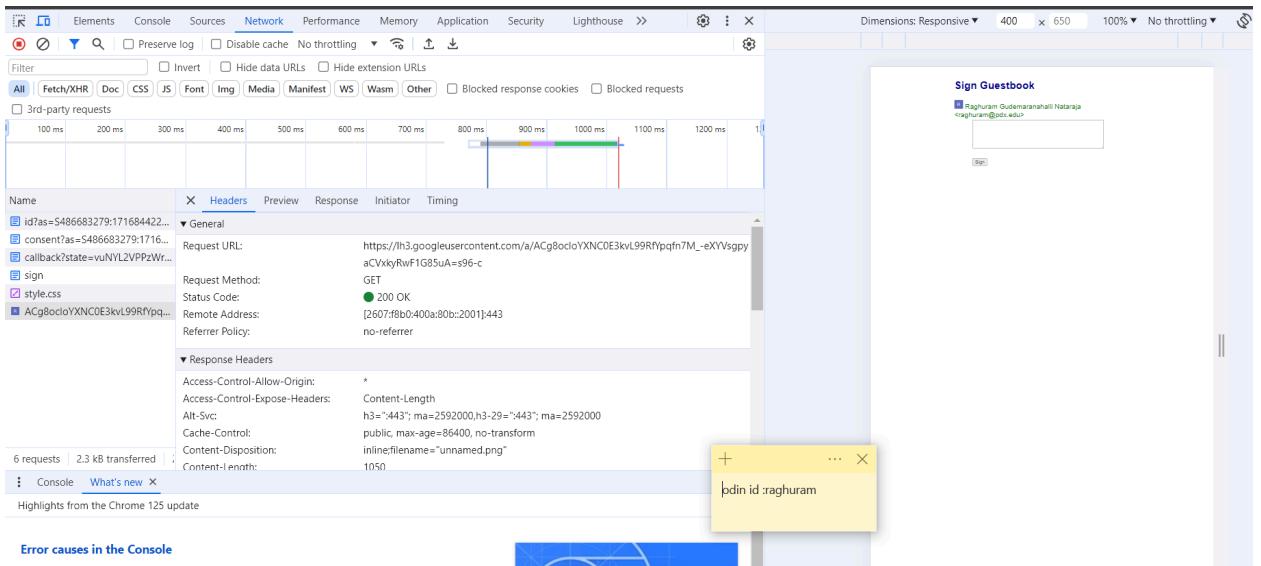
The screenshot shows the Chrome DevTools Network tab with a list of requests. The requests include idToken, consent, callback, sign, and style.css. The timeline shows the duration of each request. To the right, a "Sign Guestbook" dialog is displayed, asking for "Signer Name" and "Email Address". A yellow callout box points to the "Sign" button.

- 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.



15. Secrets manager deployment

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

The screenshot shows a web browser window with the address bar displaying "gcp-oauth-gb-mfvgnqrxa-uwa.run.app". The page title is "Guestbook". Below it are links "Sign here | Logout". The main content area is titled "Entries" and contains two entries:

- Raghuram Gudemaranhalli Nataraja <raghuram@pdx.edu> on 2024-05-27 21:18:00.071010+00:00
- Raghuram Ram <rughuram819@gmail.com> on 2024-05-27 21:26:53.351654+00:00

A yellow tooltip at the bottom right of the interface says "jodin id :raghuram".

16. Removing access

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

← How Google helps you sign in to Guestbook

Sign in with Google

You use Sign in with Google to sign in to your Guestbook account. Google uses your name, email address, and profile picture to securely sign you in. [Learn more](#) ?

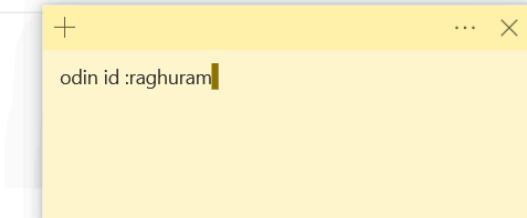
Access given on: 18 minutes ago

Access given to: gcp-oauth-gb-au5d24w6za-uw.a.run.app ?

If you stop using Sign in with Google
If Guestbook uses automatic sign-in, Google will no longer automatically sign you in to Guestbook

[Stop using Sign in with Google](#)

Guestbook uses Google Account sign-in prompts to offer a faster way to sign in
To make changes, go to [Settings](#)



08.4g: Firebase

4. Authentication setup

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

The screenshot shows the Firebase Authentication console. On the left, there's a sidebar with various options like User account management, User account linking, User actions, Blocking functions, User activity logging, Sign-up quota, Domains, Authorized domains (which is selected and highlighted in blue), and SMS. The main area is titled "Authorized domains" and contains a table of authorized domains. The table has columns for "Authorized domain" and "Type". It lists "localhost" as Default, "fir-raghuram.firebaseio.com" as Default, "fir-raghuram.web.app" as Default, and "cloudshell.dev" as Custom. Below the table is a yellow input field containing the placeholder "signin id:raghuram".

Authorized domain	Type
localhost	Default
fir-raghuram.firebaseio.com	Default
fir-raghuram.web.app	Default
cloudshell.dev	Custom

8. Bundling with Webpack

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

The screenshot shows the Cloud Shell Editor with an open file named "webpack.config.js". The code editor interface includes tabs for "index.html" and "webpack.config.js". The code itself is a webpack configuration object:

```

const path = require('path');
const rootConfig = {
  mode: 'development',
  optimization: {
    usedExports: true, // tells webpack to tree-shake
  },
  devtool: 'eval-source-map'
};
const appConfig = {
  ...rootConfig,
  entry: './src/index.js',
  output: {
    ...
  }
};

```

12. Add authentication

- What missing functions deal with user authentication?

`signIn()`

- What missing functions deal with sending and receiving messages?

`saveMessage (messageText)`

`loadMessages ()`

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`

16. Test application with text messaging

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

The screenshot shows the Firebase Cloud Firestore interface. On the left, the sidebar has 'Project Overview' selected. Under 'Firestore Database', 'messages' is selected. The main area shows a document named 'RFeuK4MtXUv2Uh3zH1o7' under the 'messages' collection. A yellow callout box highlights the 'id' field with the value 'odin id:raghuram'. The document also contains fields for 'name', 'profilePicUrl', 'text', and 'timestamp'.

name	profilePicUrl	text	timestamp
"Raghuram Gudemaranahalli Nataraja"	"https://lh3.googleusercontent.com/a/ACg8ocloYXNC0E3kvL9eXYVsgpyaCvxkyRwf1G85uA=s96-c"	"Hi"	May 27, 2024 at 9:02:19 PM UTC-7

17. Manual message insertion

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

The screenshot shows a web-based 'Friendly Chat' application. At the top, there's a header bar with a user profile icon and the name 'Raghuram Gudemaranahalli Nataraja'. The main area shows a conversation between two users: 'R' (the bot) and 'wu'. R has sent a message 'Hi'. Below the messages, there's a text input field with placeholder 'Message...', a 'SEND' button, and a file upload icon. A yellow callout box highlights the input field, which contains the text 'odin id:raghuram'.

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

```
var LOADING_IMAGE_URL =  
'https://www.google.com/images/spin-32.gif?a';
```

19. Test application with image messaging

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

imageUrl, storageUri are added in image url

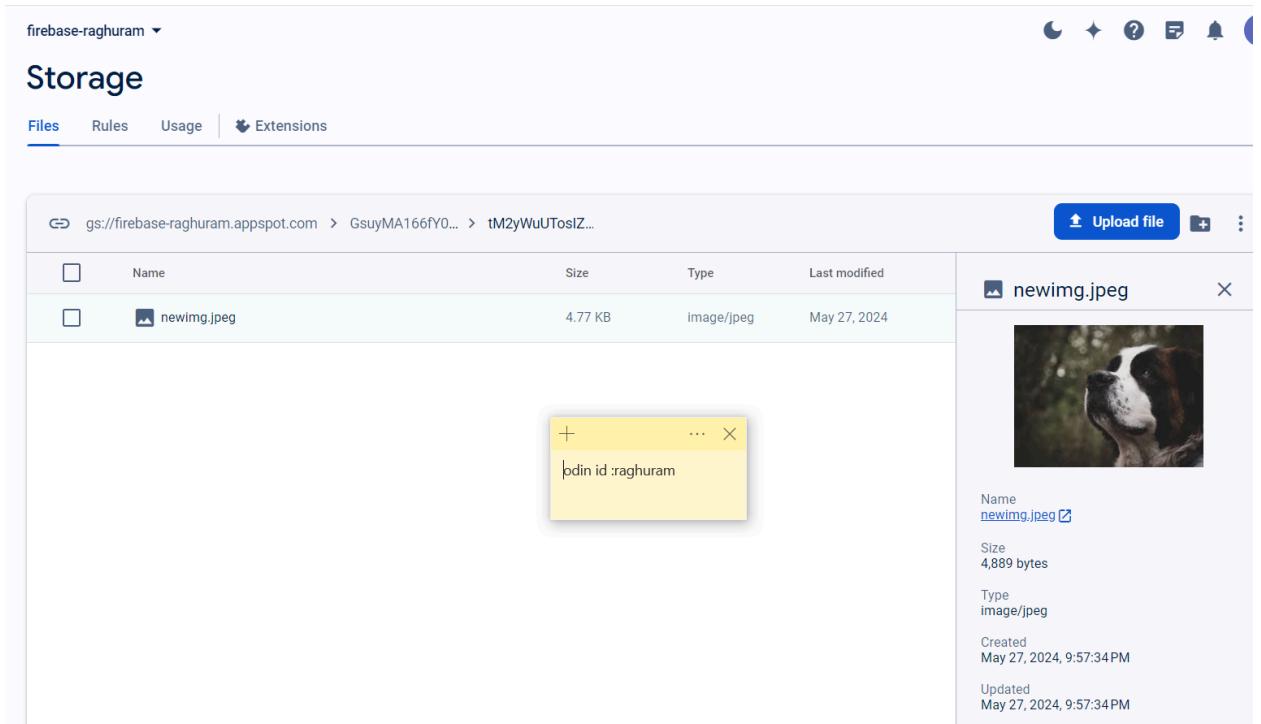
- What URL and storage location can the image be found at?

<https://console.firebaseio.google.com/u/2/project.firebaseio-raghuram/storage.firebaseio-raghuram.appspot.com/files/~2FGsuyMA166fY01ZvzVxclnf7Gtam1~2Fpv5pXPbok4eQGvzJyH2K>

storage location:

GsuyMA166fY01ZvzVxclnf7Gtam1/tM2yWuUToslZAu5S9JC_s/newimg.jpeg

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



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.

