Lab notebook Week 8

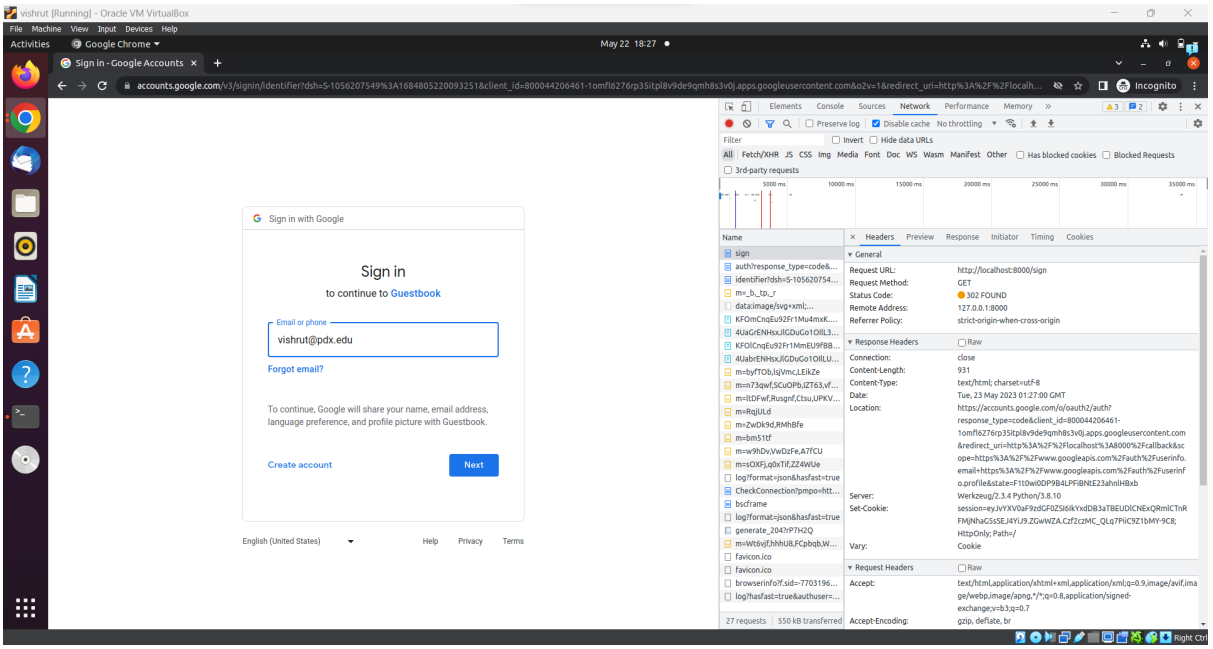
Submitted by: Vishrut Sharma (OdinID: vishrut)

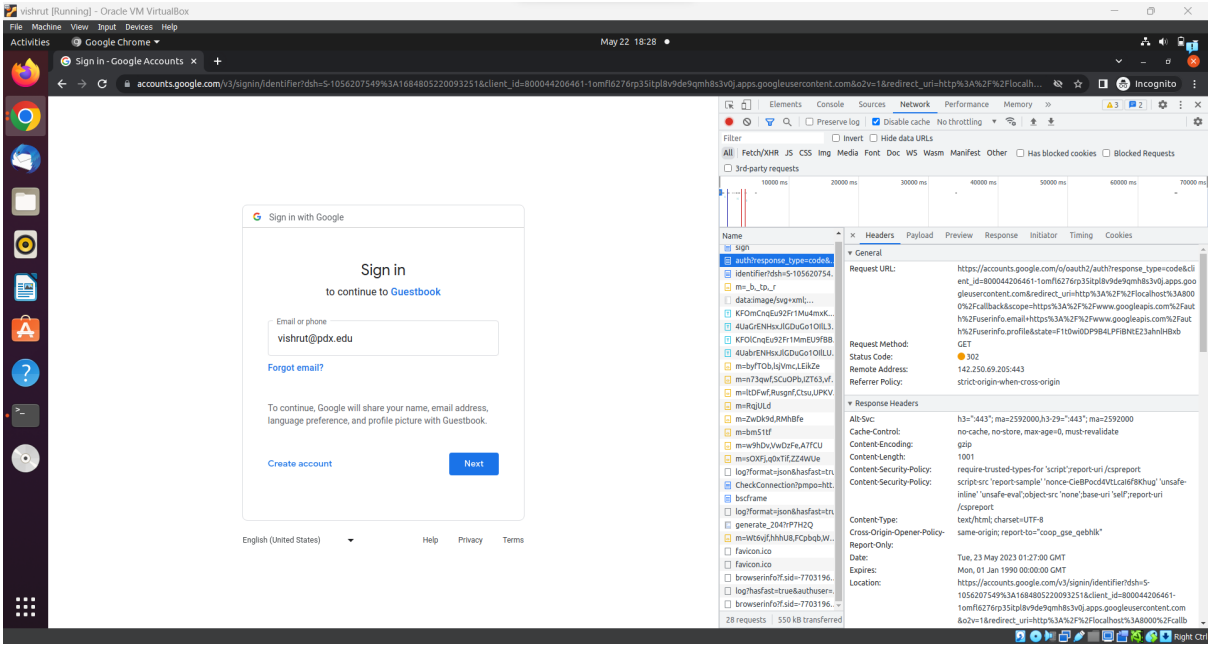
Table of Contents

08.1 OAuth2 Guestbook

12. Running the code

- **Take a screenshot of the Headers that includes the URL and the returned HTTP status code for each request 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?**

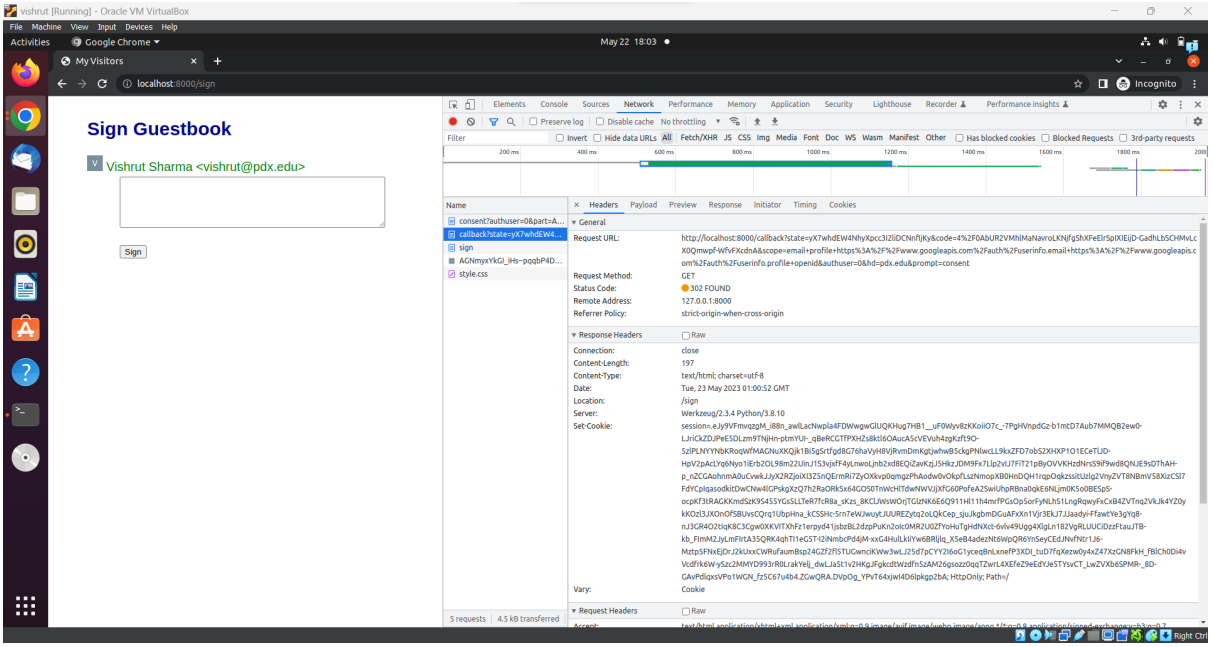
Answer: The following 3 lines of code are responsible for the second request.

1) authorization_url, state = google.authorization_url(authorization_base_url)

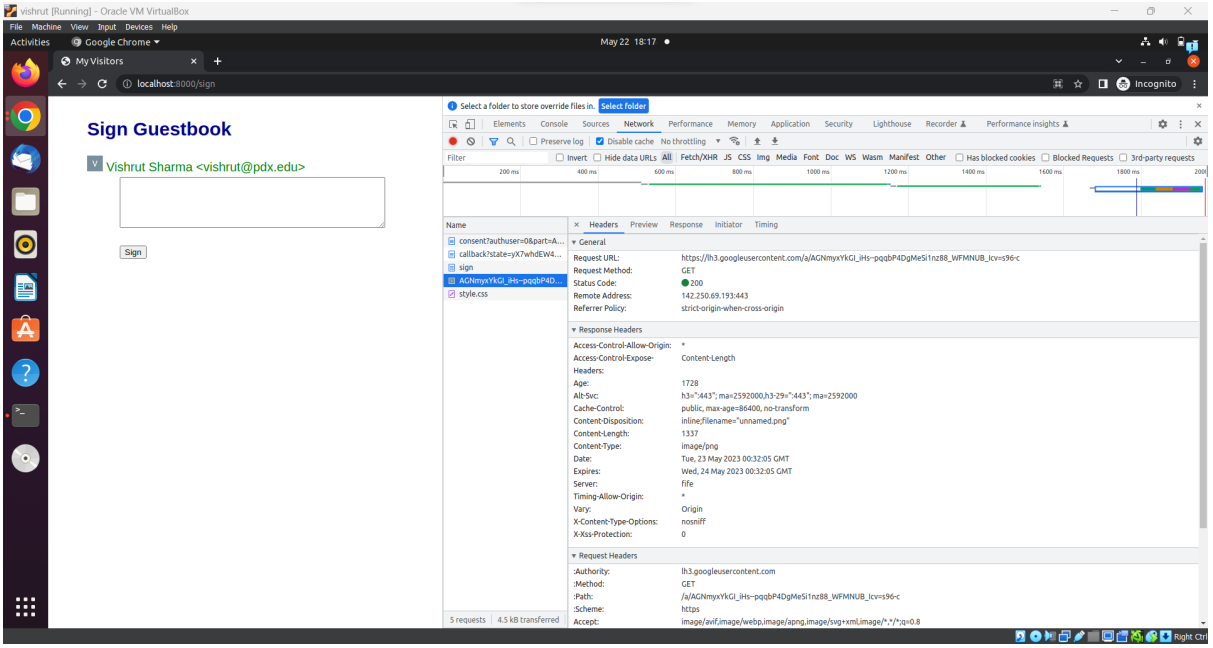
2) session['oauth_state'] = state
3) return redirect(authorization_url)

- **Take a screenshot of the Headers that includes the entire Callback URL and its returned HTTP status code. What location is the User sent to as a result of this request?**

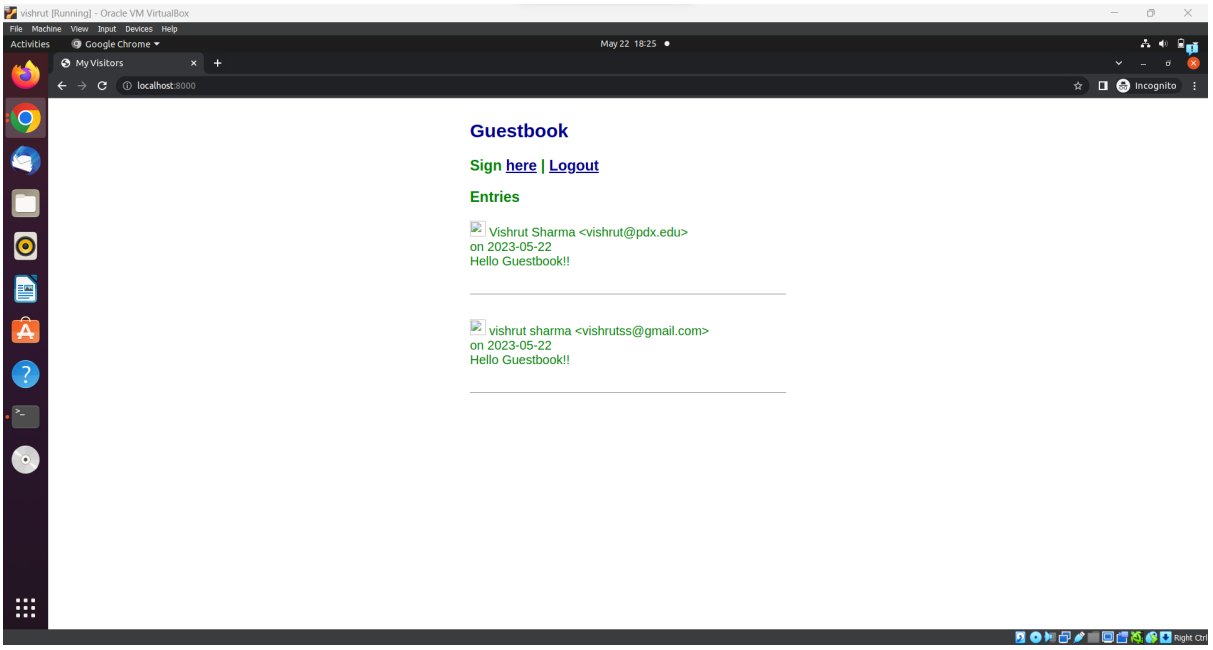
Answer: The user is sent to "/sign".



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

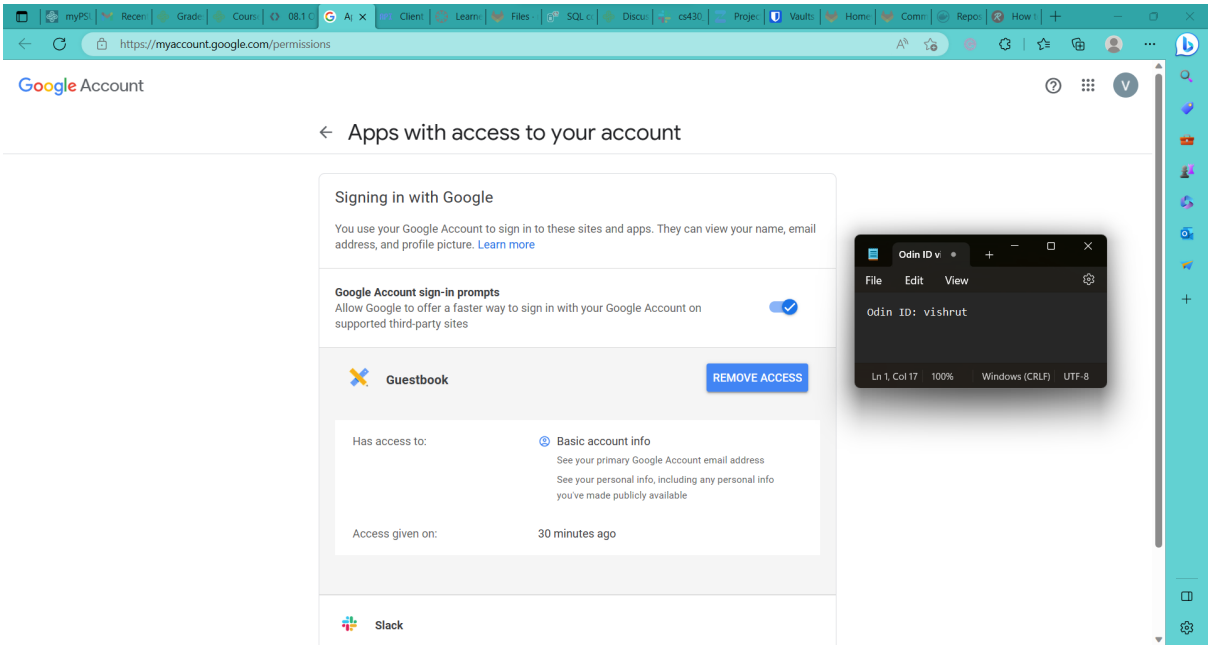


- **Take a screenshot showing multiple authenticated accounts have been able to sign the Guestbook.**



13. Removing access

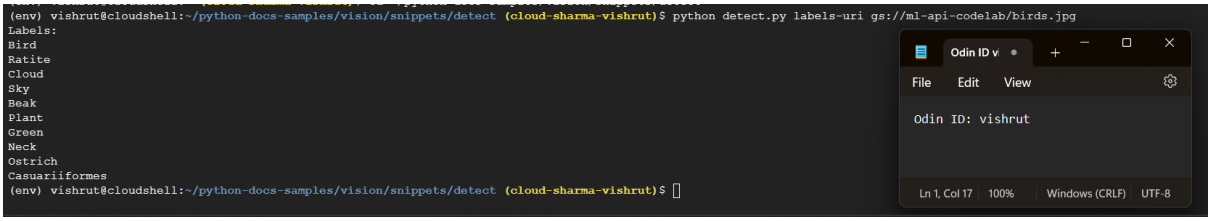
- **Take a screenshot of the expanded information that includes your OdinId for your lab notebook.**



08.2g: ML APIs

3. Vision

- **Show the output for your lab notebook**



- **What is the name of the function?**

Answer: detect_labels_uri

- **What type of Vision client is instantiated in it?**

Answer: ImageAnnotatorClient

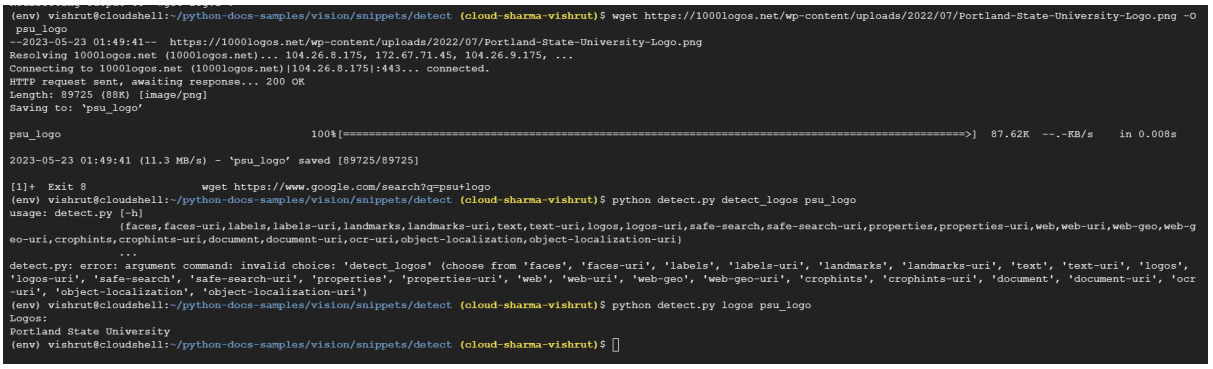
- **What method is invoked in the Vision client to perform the detection?**

Answer: label_detection(image=image)

- **What is the name of the attribute in the response object that contains the results we seek?**

Answer: labels

- **Take a screenshot of the output for the above commands**

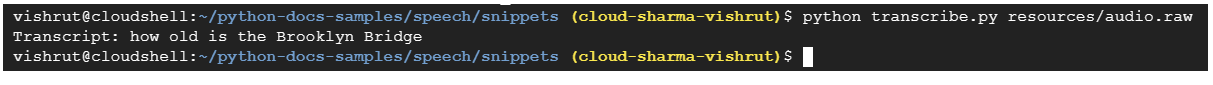


- **What method is invoked in the Vision client to perform the detection?**

Answer: detect_logos

4. Speech

- **Show the output for your lab notebook**



- **What is the name of the function?**

Answer: transcribe_file(speech_file)

- **What method is invoked in the Speech client to perform the detection?**

Answer: client.recognize(config=config, audio=audio)

- **What is the name of the attribute in the response object that contains the results we seek?**

Answer: response.results

5. Translate

- **Show the output for your lab notebook**

```
vishrut@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-sharma-vishrut)$ python snippets.py translate-text en '你有沒有帶外套'
Text: 你有沒有帶外套
Translation: do you have a coat
Detected source language: zh-TW
vishrut@cloudshell:~/python-docs-samples/translate/samples/snippets (cloud-sharma-vishrut)$
```

- **What is the name of the function?**

Answer: translate_text(target: str, text: str) -> dict

- **What method is invoked in the Translate client to perform the detection?**

Answer: translate_client.translate(text, target_language=target)

- **What is the name of the attribute in the response object that contains the results we seek?**

Answer: result["translatedText"]

6. Natural Language

- **Show the output for your lab notebook**

```
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python language.py 'homework is awful!'
"homework is awful!" has sentiment=-0.800000011920929

Entities are:
name: homework
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python language.py 'homework is ok'
"homework is ok" has sentiment=0.30000001192092896

Entities are:
name: homework
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python language.py 'homework is awesome?'
"homework is awesome?" has sentiment=0.4000000059604645

Entities are:
name: homework
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python language.py 'homework is awesome!'
"homework is awesome!" has sentiment=0.899999761581421

Entities are:
name: homework
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python language.py 'The protestors in Oregon put on gas masks and wore yellow t-shirts'
"The protestors in Oregon put on gas masks and wore yellow t-shirts" has sentiment=-0.6000000238418579

Entities are:
name: protestors
name: gas masks
name: Oregon
name: t-shirts
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$
```

8. Code

- **What is the name of the function that performs the transcription?**

Answer: transcribe_gcs

- **What is the name of the function that performs the translation?**

Answer: translate_text

- **What is the name of the function that performs the entity analysis on the translation?**

Answer: entities_text

- **What is the name of the function that performs the entity analysis on the image?**

Answer: detect_labels_uri

9. Test integration

- **If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?**

Answer: Within the "compare_audio_to_image" function, we can try to adjust the confidence threshold that determines a match between entities and labels. At present, the function only verifies if the entity name precisely matches a label. To enhance the comparison logic, we can enable partial matches or explore the possibility of utilizing a similarity metric to evaluate the connection between entities and labels.

- **If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?**

Answer: We can modify the confidence threshold for matching entities and labels. It is worth exploring different similarity metrics or considering the integration of additional APIs or models to extract more comprehensive details from both the audio and image. This could involve object detection or scene analysis to obtain more detailed information from the data.

- **If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this?**

Answer: We can reassess the confidence threshold and the matching logic employed for entities and labels. Furthermore, it is worth considering the exploration of advanced techniques, such as leveraging deep learning-based models, for audio and image analysis. These models possess the capability to capture more sophisticated features and semantics, facilitating a more profound comprehension and comparison between the audio and image data.

13. Video Intelligence

- **What are the top 3 labels that the Video Intelligence API associates with the video and what is its confidence in them?**

```
(env) vishrut@cloudshell:~ (cloud-sharma-vishrut)$ python labels.py gs://${CLOUD_STORAGE_BUCKET}/SportsBloopers2016.mp4

Processing video for label annotations:

Finished processing.
Video label description: sport venue
        Label category description: location
        Label category description: structure
        Segment 0: 0s to 178s
        Confidence: 0.3432130515575409


Video label description: audience
        Label category description: people
        Segment 0: 0s to 178s
        Confidence: 0.4254549741744995


Video label description: stadium
        Label category description: location
        Label category description: structure
        Segment 0: 0s to 178s
        Confidence: 0.5119114518165588
```

- **What is the name of the client class in the package that is used?**

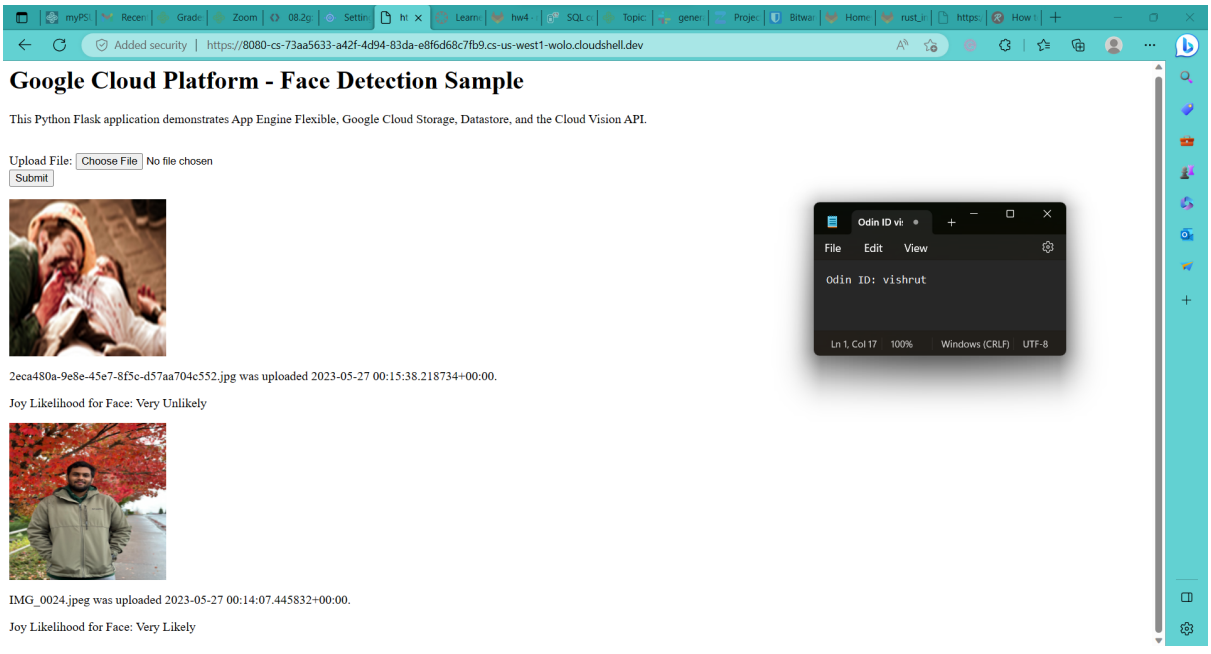
Answer: videointelligence.VideoIntelligenceServiceClient

- **What method is used in that class to perform the annotation?**

Answer: annotate_video

16. Application

- **Take a screenshot for your lab notebook that includes the URL.**



17. Code

- **What line of code creates the query for previous detections?**

Answer: Line 39. Query=datastore_client.query(kind=”Faces”)

- **What line of code sends the query to Cloud Datastore?**

Answer: Line 40. Image_entities=list(query.fetch())

- **Show the line that retrieves the name of the storage bucket to use.**

Answer: bucket = storage_client.get_bucket(CLOUD_STORAGE_BUCKET)

- **What form field is used to specify the uploaded photo?**

Answer: photo = request.files["file"]

- **Show the line that copies the photo's contents to the storage bucket.**

Answer: blob = bucket.blob(photo.filename)

blob.upload_from_string(photo.read(), content_type=photo.content_type)

- **What method in Vision's annotation client is used to perform the analysis?**

Answer: faces = vision_client.face_detection(image=image).face_annotations

- **What fields are stored in Cloud Datastore for each image?**

Answer:

blob_name: The name of the blob (image file) in the storage bucket.
image_public_url: The publicly accessible URL of the image.
timestamp: The date and time of the upload.
joy: The likelihood that the detected face displays 'joy.'

- **What happens at the end of the upload_photo route?**

Answer: Upon uploading the photo and analyzing the face, a fresh entity will be generated within the cloud datastore. This entity will contain all the relevant information pertaining to the photo and the analyzed face. Subsequently, the entity will be stored in Cloud Datastore, and we will then be redirected to the home page.

08.3g: Firebase

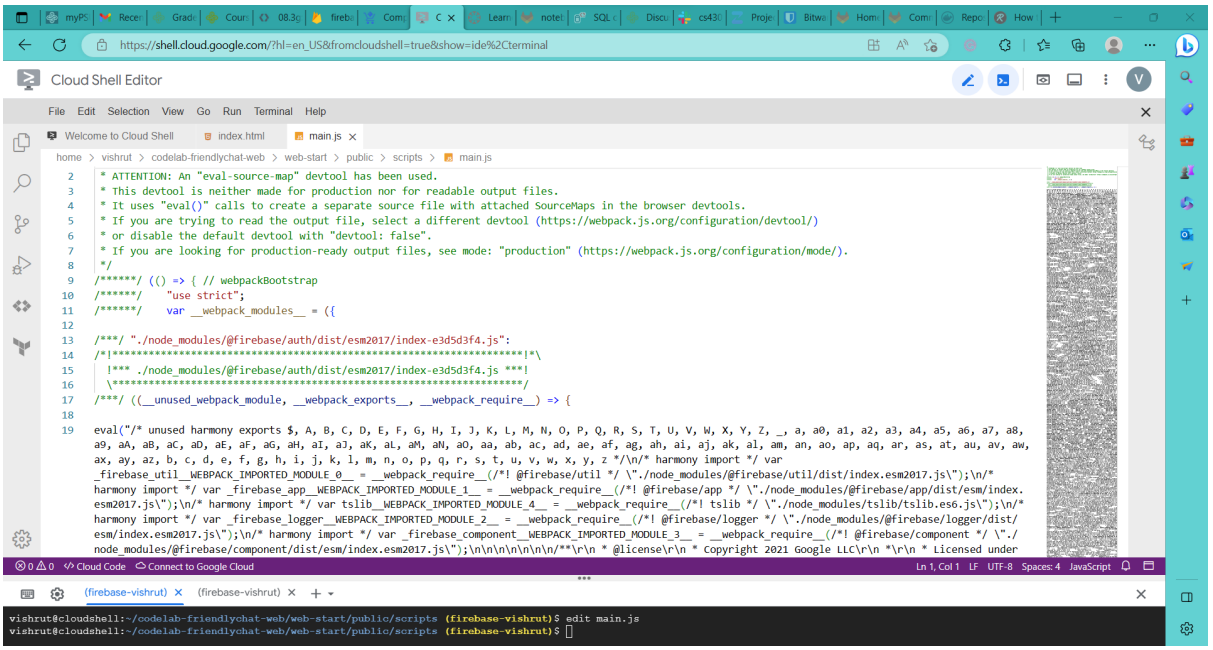
4. Authentication setup

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

Answer: localhost, fir-vishrut.firebaseapp.com, fir-vishrut.web.app

8. Bundling with Webpack

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



12. Add authentication

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

Answer: signIn, signOutUser, and initFirebaseAuth()

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

Answer: loadMessages

13. Update UI

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

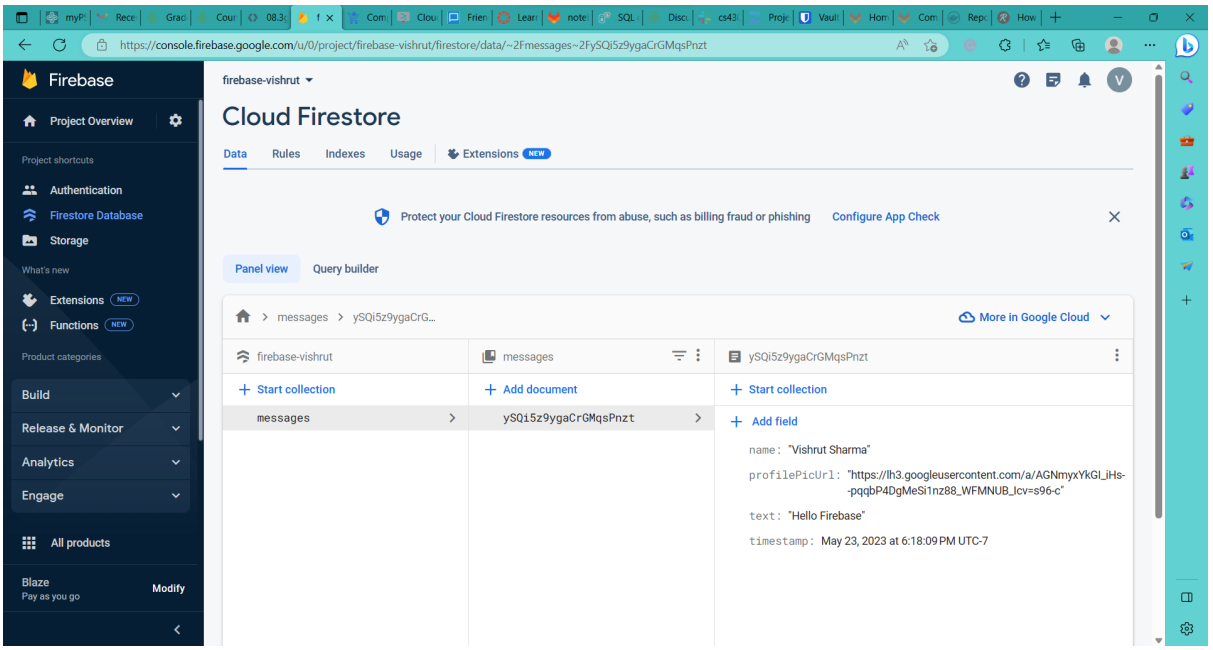
Answer: userNameElement, userPicElement and signOutButtonElement

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

Answer: signInButtonElement

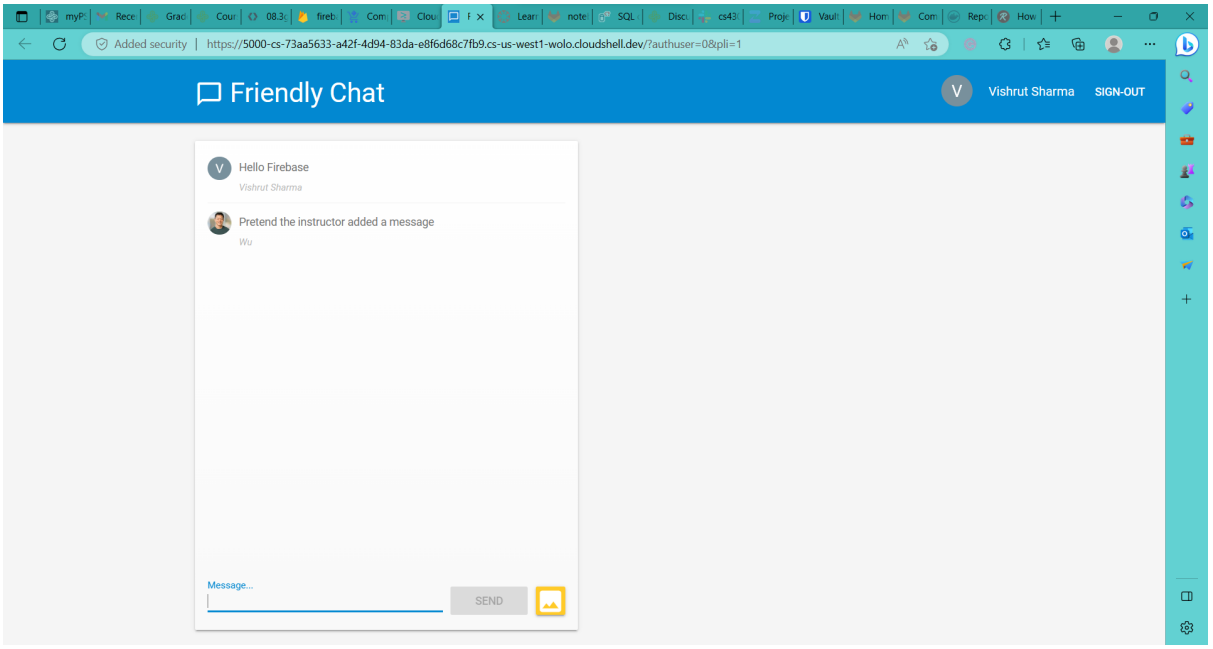
16. Test application with text messaging

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



17. Manual message insertion

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



18. Add image messaging

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

Answer: 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?**

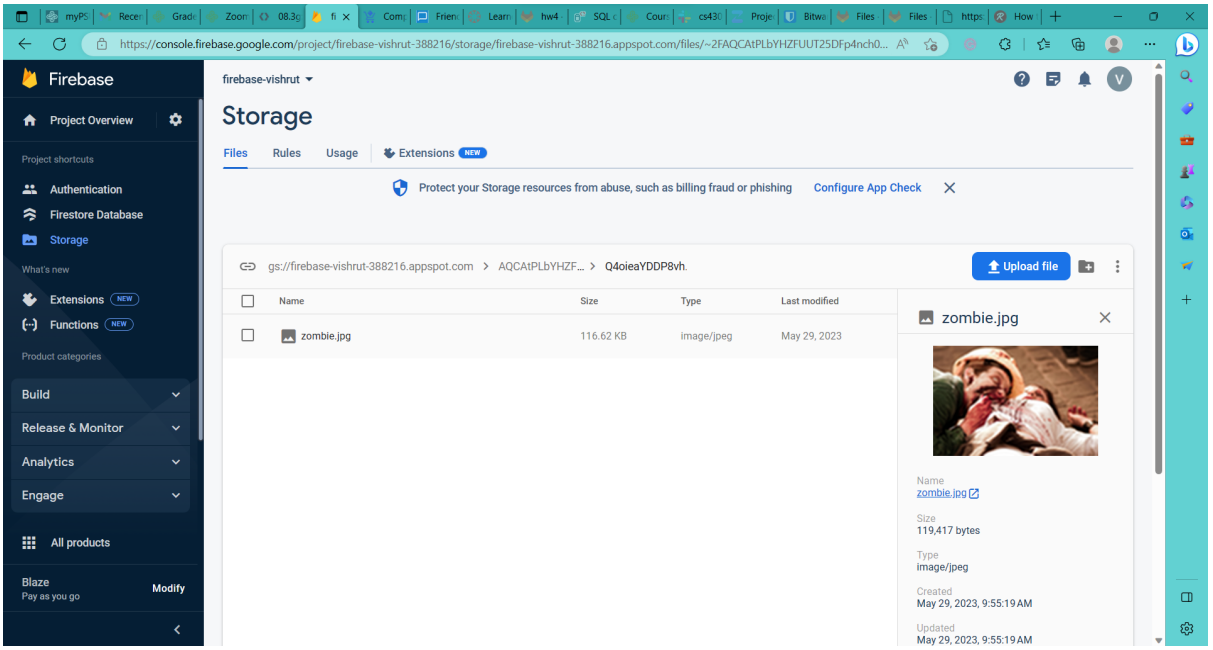
Answer: Image document has imageUrl, name, profilePicUrl, storageUri and timestamp. Text document has name, profilePicUrl, text and timestamp

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

Answer: URL: https://firebasestorage.googleapis.com/v0/b/firebase-vishrut-388216.appspot.com/o/AQCAtPLbYHZFUUT25DFp4nch0o33%2FQ4oieaYDDP8vh41Wg1fa%2Fzombie.jpg?alt=media&token=362e508a-a268-4e43-9fe8-4b0245ae6db4

Storage location: gs://firebase-vishrut-388216.appspot.com/AQCAtPLbYHZFUUT25DFp4nch0o33/Q4oieaYDDP8vh41Wg1fa/zombie.jpg

- **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?**

Answer: ~/codelab-friendlychat-web/web-start/src

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