

Lab Notebook Week 8
Submitted by: Shrikrishna Bhat

8.1 OAuth2 Guestbook	2
Card 12	2
Card 13	6
8.2 G ML APIs	7
Card 3	7
Card 4	9
Card 5	10
Card 6	11
Card 8	11
Card 9	12
Card 13	13
Card 16	14
Card 17	14
8.3G Firebase	16
Card 4	16
Card 8	17
Card 12	17
Card 13	18
Card 16	18
Card 17	19
Card 18	19
Card 19	20
Card 20	21

8.1 OAuth2 Guestbook

Card 12

8.1.1 Take a screenshot of the Headers that includes the URL and the returned HTTP status code for each request for your lab notebook.

The screenshot shows a browser window with a Google sign-in dialog titled "Sign in with Google". The dialog asks for permission to "continue to guestbook". Below the dialog, there are fields for "Email or phone" and "Forgot email?", and buttons for "Create account" and "Next". At the bottom, there are links for "English (United States)", "Help", "Privacy", and "Terms". Above the dialog, the browser's address bar shows a URL starting with "accounts.google.com/v3/signin/did...".

Overlaid on the browser window is the developer tools Network tab. The Network tab displays a list of requests. The first request, labeled "E sign", has the following details:

- URL: <http://localhost:8000/sign>
- Request Method: GET
- Status Code: 302 FOUND
- Remote Address: 127.0.0.1:8000
- Referrer Policy: strict-origin-when-cross-origin

The Network tab also shows other requests with various URLs and methods, such as "KFOmCnqEu9frf1MuKmxk.wof2", "4UAGrENHnxJlCdGo1Oll3Qw...", and "KF0ICnqEu9frf1MuKmxk.wof2". The "Response Headers" section for the first request shows the following headers:

- Connection: close
- Content-Length: 931
- Content-Type: text/html; charset=utf-8
- Date: Tue, 23 May 2023 01:13:53 GMT
- Location: [https://accounts.google.com/o/oauth2/auth?state=jk\\$EMFz2IKWe3odB6rhmgA2zhdvjetd5s1tpg.apps.googleusercontent.com&redirect_uri=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email&https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile&state=jk\\$EMFz2IKWe3obghopYGrfxuA8Z4](https://accounts.google.com/o/oauth2/auth?state=jk$EMFz2IKWe3odB6rhmgA2zhdvjetd5s1tpg.apps.googleusercontent.com&redirect_uri=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email&https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile&state=jk$EMFz2IKWe3obghopYGrfxuA8Z4)
- Server: Werkzeug/2.3.4 Python/3.10.6
- Set-Cookie: sessioneyjxV0af92dGF0ZS6ijVRUJBQYVLU3WGICSFhOVmh1aJ2R1Nb3AA...nV4QUAnNCJ9.ZGwTUQ_c6OuTkT-GUEoVmXvIcXFoez4; HttpOnly; Path=/; Vary: Cookie
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
- Accept-Encoding: gzip, deflate, br
- Accept-Language: en-GB,en;q=0.9
- Connection: keep-alive
- Cookie: sessioneyjxV0af92dGF0ZS6ijVRUJBQYVLU3WGICSFhOVmh1aJ2R1Nb3AA...NURUJ9.ZGwTMg_3KMN4xb5yI0y@TEtqQcTrTbJsc; localhost:8000
- Host: http://localhost:8000/
- Referer: <http://localhost:8000/>
- Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A-Brand";v="24"
- Sec-Ch-Ua-Mobile: 0
- Sec-Ch-Ua-Platform: "Linux"

The Network tab also lists other requests with their respective URLs and details. The status bar at the bottom of the browser shows "25 requests 112 kB transferred" and the time "6:14 PM".

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

This code is found in the sign.py and the lines which say are
authorization_url, state = google.authorization_url(authorization_base_url)
session['oauth_state'] = state
return redirect(authorization_url)

8.1.3 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: User is sent to /sign location once he successfully signs in and now the user can sign the guestbook.

My Visitors +
localhost:8000/sign

Sign Guestbook

Shrikrishna Bhat <shbhat@pdx.edu>

Sign

Request URL: https://accounts.google.com/o/oauth2/auth/consent#authuser=&start=AjBh...
Request Method: GET
Status Code: 302
Remote Address: [2607:fbb0:400a:805::100d]:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Connection: close
Content-Length: 197
Content-Type: text/html; charset=utf-8
Date: Tue, 23 May 2023 01:00:35 GMT
Location: http://localhost:8000/callback?state=QfDceWPNDpKXLdmUlVtTznnNem&code=4%2fOAURzVNubkHC-%QfRgRpOpNnhGz2Af_LqZvzKcXW48ezxgCVLecmihd76NAtw&co...
Server: gunicorn/2.3.4 Python/3.10.6
Set-Cookie: session-e.xVWhMydkQ_4HrTyTCHeHtJU5ExCR4fbJBzBBfV3xtu9LsVuUDfLorM_o_mOVHskijox_cj7oM...
HTTP/2.0 302 FOUND

My Visitors +
localhost:8000/sign

Sign Guestbook

Shrikrishna Bhat <shbhat@pdx.edu>

Sign

Request URL: http://localhost:8000/callback?state=QfDceWPNDpKXLdmUlVtTznnNem&code=4%2fOAURzVNubkHC-%QfRgRpOpNnhGz2Af_LqZvzKcXW48ezxgCVLecmihd76NAtw&co...
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:8000
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Content-Type: application/javascript; charset=UTF-8
Date: Tue, 23 May 2023 01:00:35 GMT
Server: gunicorn/2.3.4 Python/3.10.6
Set-Cookie: session-e.xVWhMydkQ_4HrTyTCHeHtJU5ExCR4fbJBzBBfV3xtu9LsVuUDfLorM_o_mOVHskijox_cj7oM...
HTTP/2.0 200 OK

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

The screenshot shows a web browser window with the following details:

- Title Bar:** My Visitors → localhost:8000/sign
- Page Content:** A guestbook form titled "Sign Guestbook". The input field contains "Shrikrishna Bhat <shbhat@pdx.edu>". Below the input field is a "Sign" button.
- Network Tab (Developer Tools):** Shows a request to "https://lh3.googleusercontent.com/a/AGNmyxY_Fc4JBWYKewdmnTKEaCsdz0krHkHFBgE=s96-c" with status code 200 (from memory cache). Headers include "Content-Type: image/jpeg" and "Date: Tue, 23 May 2023 00:37:42 GMT".
- Address Bar:** Shows "localhost:8000/sign"
- Bottom Status Bar:** 3 requests | 1.2 kB transferred | 7.4

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

The screenshot shows a guestbook application running at `localhost:8000`. The page displays a single entry from "Shrikrishna Bhat <shbhat@pdx.edu>" on May 22, 2023, with the message "hello guestbook". Below the entry is another from "Shrikrishna Bhat <shrikrishna.bht@gmail.com>" on May 22, 2023, with the message "hello from personal mail". A yellow box highlights the status bar information: "odin id: shbhat" and "email: shbhat@pdx.edu".

On the right, the Network tab of the developer tools is open, showing a POST request to `/sign` with a status of 200 OK. The request headers include `Content-Type: application/x-www-form-urlencoded` and `Cookie: session=...`. The response body contains the JSON object `{"id": 1, "name": "Shrikrishna Bhat", "email": "shbhat@pdx.edu", "entry": "hello guestbook", "date": "2023-05-22T14:15:00Z", "ip": "127.0.0.1:8000"}.`

Card 13

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

The screenshot shows the "Google Account" settings page for "Apps with access to your account". It lists several apps with access: GitLab, guestbook, Repl.it, Slack, and Zoom. A yellow box highlights the status bar information: "odin id: shbhat" and "email: shbhat@pdx.edu".

8.2 G ML APIs

Card 3

8.2.1 Show the output for your lab notebook

8.2.2 Answer the following questions:

- What is the name of the function?

```
def detect_labels_uri(uri):
```

- What type of Vision client is instantiated in it?

imageAnnotatorClient

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

```
client.label_detection(image=image)
```

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

labels

response.label_annotations

8.2.3 Take a screenshot of the output for the above commands

Cloud Shell — Mozilla Firefox

08:2g: ML APIs x Welcome - cloud-Shrik... x Cloud Shell x Lab 8 - Google Docs x psu logo - Google Search x WhatsApp x 9161f59a2f170a27897f8185eb377cf.jpeg x psu logo - Google Search x

Cloud Shell Editor

cloud-shrikirshna-shbhat x + -

```
(env) shbhat@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-shrikirshna-shbhat)$ wget https://www.logolynx.com/images/logolynx/91/9161f59a2f170a27897f8185eb377cf.jpeg -O image
--2023-05-23 01:51:50 - 0:00.00s (0 KB/s)
Resolving www.logolynx.com (www.logolynx.com)... 45.141.56.116
Connecting to www.logolynx.com (www.logolynx.com)|45.141.56.116|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 128362 (125K) [image/jpeg]
Saving to: 'image'

image                                         100%[=====] 125.35K   315KB/s  in 0.4s

2023-05-23 01:51:51 (315 KB/s) - 'image' saved [128362/128362]

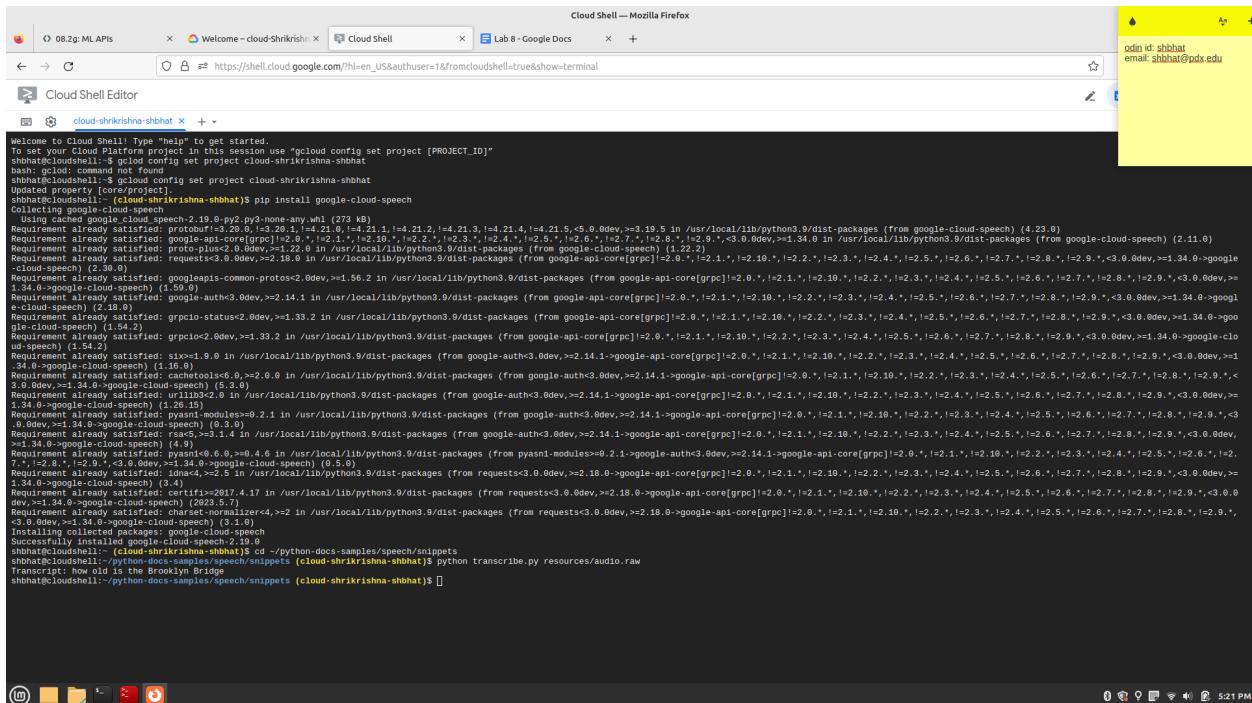
(env) shbhat@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-shrikirshna-shbhat)$ python detect.py logos image
Logos:
Portland State University
(env) shbhat@cloudshell:~/python-docs-samples/vision/snippets/detect (cloud-shrikirshna-shbhat)$
```

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

logo_detection()

Card 4

8.2.5 Show the output for your lab notebook



The screenshot shows a Cloud Shell terminal window in Mozilla Firefox. The terminal output is as follows:

```
Welcome to Cloud Shell! Type "help" for get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
shbhat@cloudshell:~$ gcloud config set project cloud-shrirkishna-shbhat
bash: gcloud: command not found
shbhat@cloudshell:~$ gcloud config set project cloud-shrirkishna-shbhat
Uploading configuration...
shbhat@cloudshell:~$ (cloud-shrirkishna-shbhat)$ pip install google-cloud-speech
Collecting google-cloud-speech
  Using cached google-cloud-speech-2.18.0-py2.py3-none-any.whl (723 kB)
Requirement already satisfied: protobuf<3.20.0,!=3.20.1,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5.5n /usr/local/lib/python3.9/dist-packages (from google-cloud-speech) (4.23.0)
Requirement already satisfied: google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0 in /usr/local/lib/python3.9/dist-packages (from google-cloud-speech) (2.11.0)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.0 in /usr/local/lib/python3.9/dist-packages (from google-cloud-speech) (1.22.2)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.9/dist-packages (from google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (2.30.0)
Requirement already satisfied: googleapis-common-protos<2.0.0dev,>=1.56.2 in /usr/local/lib/python3.9/dist-packages (from google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (1.59.0)
Requirement already satisfied: google-auth<3.0.0dev,>=2.14.1 in /usr/local/lib/python3.9/dist-packages (from google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (2.18.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.2 in /usr/local/lib/python3.9/dist-packages (from google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (2.18.2)
Requirement already satisfied: grpcio<2.0.0dev,>=1.33.2 in /usr/local/lib/python3.9/dist-packages (from google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (1.34.2)
Requirement already satisfied: six<1.9.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=2.14.1>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (1.16.0)
Requirement already satisfied: cachetools<0.2.0.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=2.14.1>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (0.15.0)
Requirement already satisfied: urllib3<2.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=2.14.1>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (1.26.15)
Requirement already satisfied: pyasn1<0.14.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=2.14.1>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (0.14.0)
Requirement already satisfied: pyasn1-modules<0.1.0 in /usr/local/lib/python3.9/dist-packages (from pyasn1<0.14.0>google-cloud-speech) (0.1.0)
Requirement already satisfied: rsa<3.1.4 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=2.14.1>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (4.0)
Requirement already satisfied: certifi<2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (2023.5.7)
Requirement already satisfied: idna<4.0,!=4.0.0 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (4.0)
Requirement already satisfied: certifi<2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[gRPC]<2.0.0,!=2.1.0,!=2.2.0,!=2.3.0,!=2.4.0,!=2.5.0,!=2.6.0,!=2.7.0,!=2.8.0,!=2.9.0,!=3.0.0dev,>=1.34.0>google-cloud-speech) (0.1.0)
Installing collected packages: google-cloud-speech
  Skipping google-cloud-speech (from -r /tmp/tmp.1qjwvz3f9t/requirements.txt) (because it is already installed)
  Skipping google-cloud-speech (from -r /tmp/tmp.1qjwvz3f9t/requirements.txt) (because it is already installed)
shbhat@cloudshell:~$ (cloud-shrirkishna-shbhat)$ cd /python-docs-samples/speech/snippets
shbhat@cloudshell:~/python-docs-samples/speech/snippets$ (cloud-shrirkishna-shbhat)$ python transcribe.py resources/audio.raw
Transcript: how old is the Brooklyn Bridge
shbhat@cloudshell:~/python-docs-samples/speech/snippets$ (cloud-shrirkishna-shbhat)$
```

8.2.6 Answer the questions:

- **What is the name of the function?**
transcribe_file(speech_file)
- **What method is invoked in the Speech client to perform the detection?**
client.recognize(config=config, audio=audio)
- **What is the name of the attribute in the response object that contains the results we seek?**
response.results

Card 5

8.2.7 Show the output for your lab notebook

8.2.8 Answer the questions

- What is the name of the function?

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

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

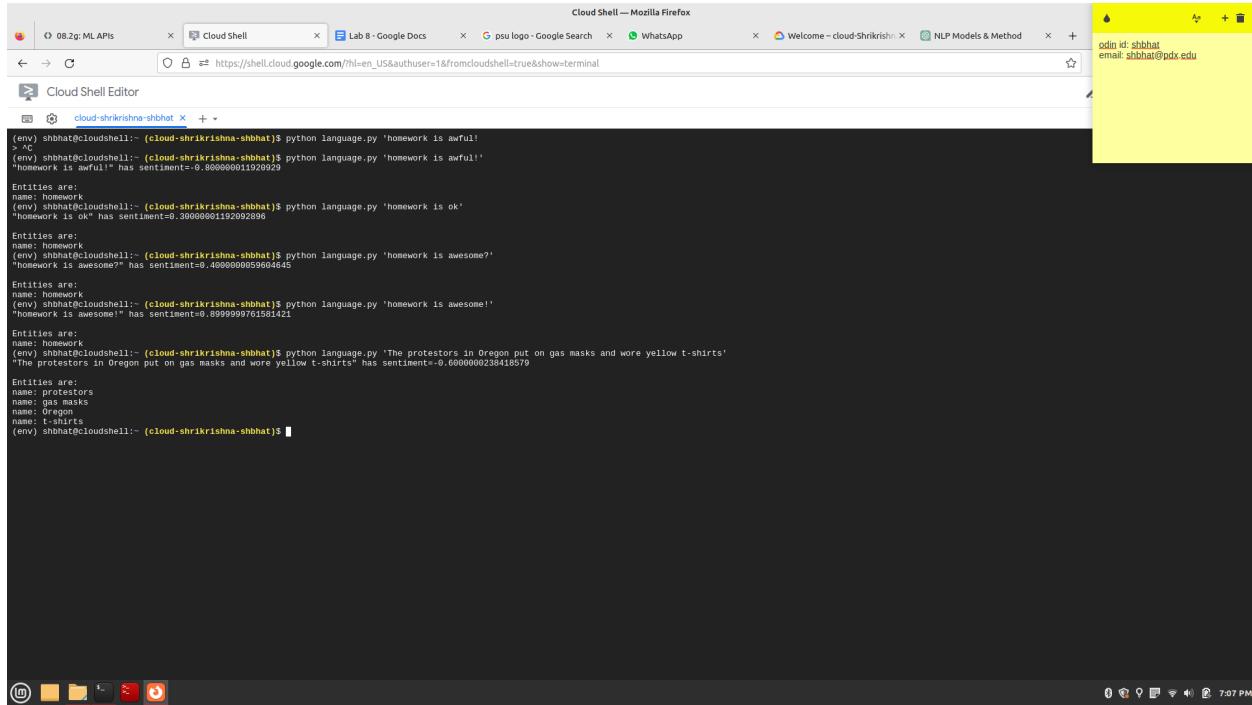
```
translate_client.translate(text, target_language=target)
```

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

result["translatedText"]

Card 6

8.2.9 Show the output for your lab notebook



```
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ python language.py 'homework is awful!'
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ python language.py 'homework is awful!
"homework is awful!" has sentiment=-0.800000011920929
Entities are:
name: homework
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ python language.py 'homework is ok'
"homework is ok" has sentiment=0.3000000119392096
Entities are:
name: homework
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ python language.py 'homework is awesome?'
"homework is awesome?" has sentiment=0.4000000059604645
Entities are:
name: homework
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ python language.py 'homework is awesome!'
"homework is awesome!" has sentiment=0.8999999761581421
Entities are:
name: protestors
name: gas masks
name: oregon
name: t-shirts
(env) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$ 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) shbhat@cloudshell:~ (cloud-shrkrishna-shbhat)$
```

Card 8

8.2.10 Answer the questions:

- **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()

Card 9

8.2.11 If the program deems them unrelated, then based on the results from the APIs, what must be changed in the program to address this? (3 answers are as below)

python solution.py de-DE gs://ml-api-codelab/de-ball.wav gs://ml-api-codelab/football.jpg

In the compare_audio_to_image function, we can try adjusting the confidence threshold for considering a match between entities and labels. Currently, it only checks if the entity name exactly matches a label. We can modify the comparison logic to allow for partial matches or we can consider using a similar metric to determine the relatedness between entities and labels.

python solution.py tr-TR gs://ml-api-codelab/tr-bike.wav gs://ml-api-codelab/bicycle.jpg

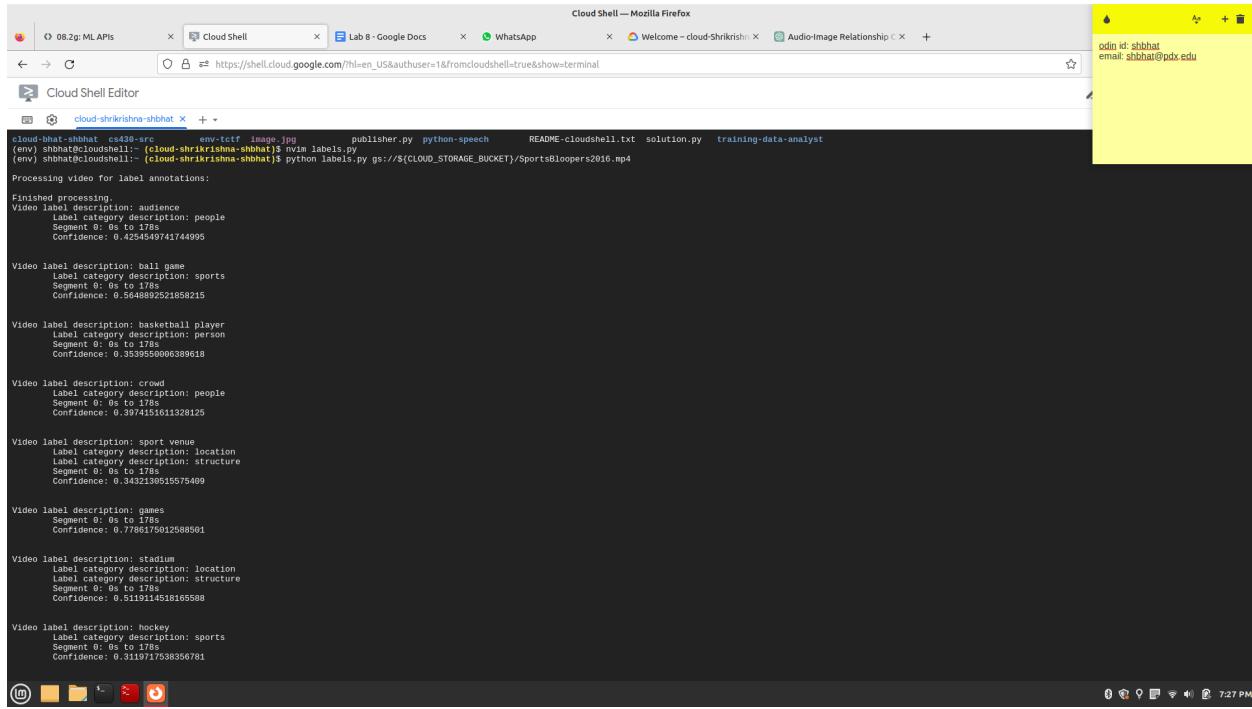
This is related to the first answer. We must adjust the confidence threshold for matching entities and labels. We can also experiment with different similarity metrics or consider using additional APIs or models to extract more detailed information from the audio and image, such as object detection or scene analysis.

python solution.py tr-TR gs://ml-api-codelab/tr-ostrich.wav gs://ml-api-codelab/birds.jpg

Even here, we can review the confidence threshold and matching logic between entities and labels. We can also explore using more advanced techniques like deep learning-based models for audio and image analysis. These models can capture higher-level features and semantics, enabling better understanding and comparison between the audio and image.

Card 13

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



```
Cloud Shell — Mozilla Firefox
Cloud Shell Editor
cloud-shrikrishna-shbhat cloud-shell + 
cloud-shrikrishna-shbhat cs430-src env-icif image.jpg publisher.py python-speech README-cloudshell.txt solution.py training-data-analyst
(env) shbhat@cloudshell:[cloud-shrikrishna-shbhat]$ nvim labels.py
(env) shbhat@cloudshell:[cloud-shrikrishna-shbhat]$ python labels.py gs://$(CLOUD_STORAGE_BUCKET)/SportsBloopers2016.mp4

Processing video for label annotations:
Finished processing.
Video label description: audience
Label category description: people
Segment 0: 0s to 17s
Confidence: 0.422459741744995

Video label description: ball game
Label category description: sports
Segment 0: 0s to 17s
Confidence: 0.504682523058215

Video label description: basketball player
Label category description: person
Segment 0: 0s to 17s
Confidence: 0.3539559006389619

Video label description: crowd
Label category description: people
Segment 0: 0s to 17s
Confidence: 0.3974151613128125

Video label description: sport venue
Label category description: location
Label category description: structure
Segment 0: 0s to 17s
Confidence: 0.3432130515575409

Video label description: stadium
Label category description: location
Label category description: structure
Segment 0: 0s to 17s
Confidence: 0.5119114518165588

Video label description: hockey
Label category description: sports
Segment 0: 0s to 17s
Confidence: 0.3119717538356781
```

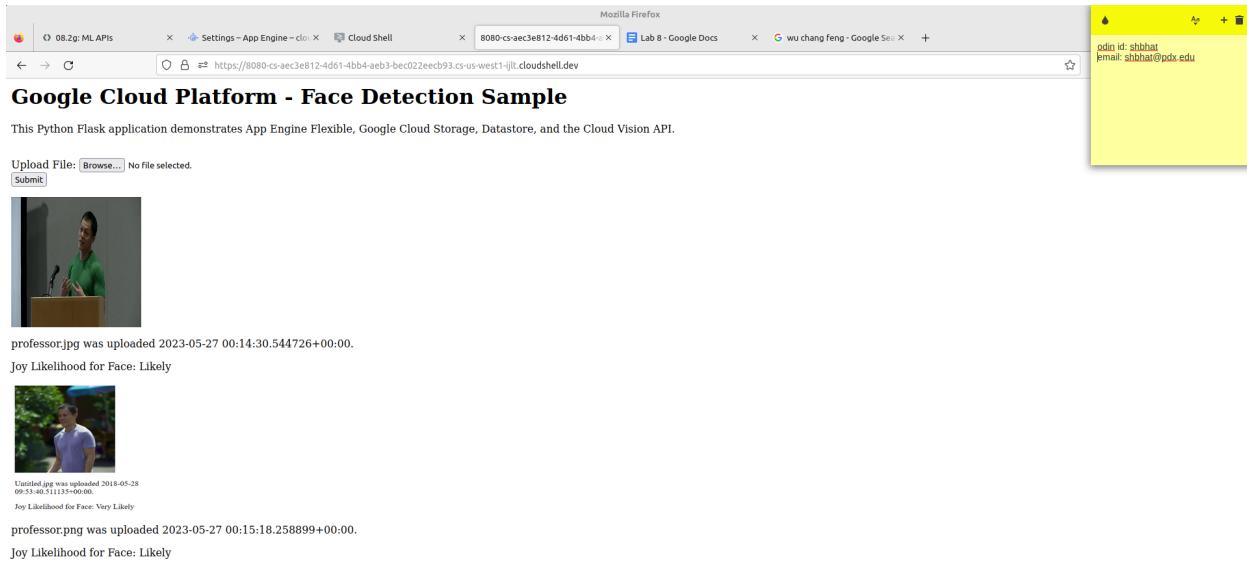
8.2.13 What is the name of the client class in the package that is used? `videointelligence.VideoIntelligenceServiceClient`.

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

The method used in that class to perform the annotation is `annotate_video`.

Card 16

8.2.15 Take a screenshot for your lab notebook that includes the URL



Card 17

8.2.16 Answer the following questions.

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

Answer: `query = datastore_client.query(kind="Faces")`, this line creates query for previous detection.

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

Answer: `image_entities = list(query.fetch())` this line sends the query to cloud database.

- 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.

```
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?**

`faces = vision_client.face_detection(image=image).face_annotations`

This line calls the `face_detection` method of the `vision_client` and retrieves the detected faces as `face_annotations`.

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

`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?**

A new entity will be created in the cloud datastore with the information about the uploaded photo and the analyzed face.

The entity is saved to Cloud Datastore and we will be redirected to the home page.

8.3G Firebase

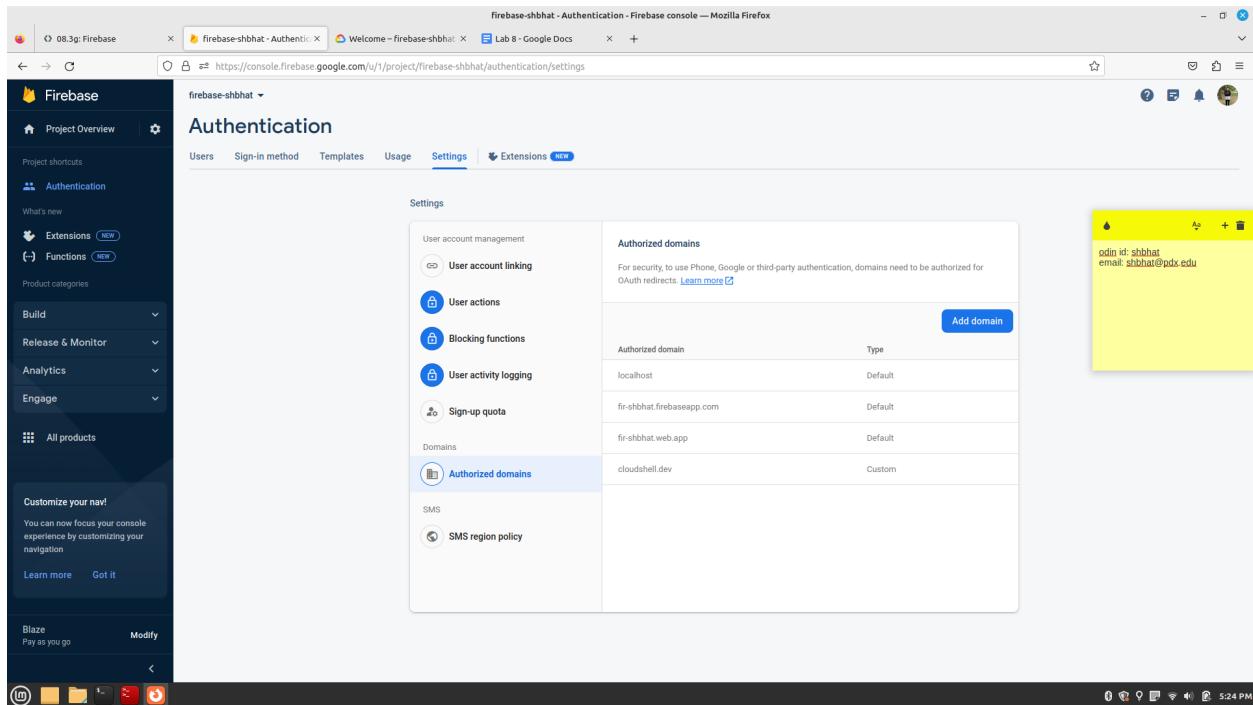
Card 4

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

localhost

fir-shbhat.firebaseio.com

fir-shbhat.web.app



The screenshot shows the Firebase Authentication settings page in the Firebase console. The left sidebar has 'Authentication' selected. The main area shows the 'Settings' tab is active. Under 'Authorized domains', there is a table with three rows:

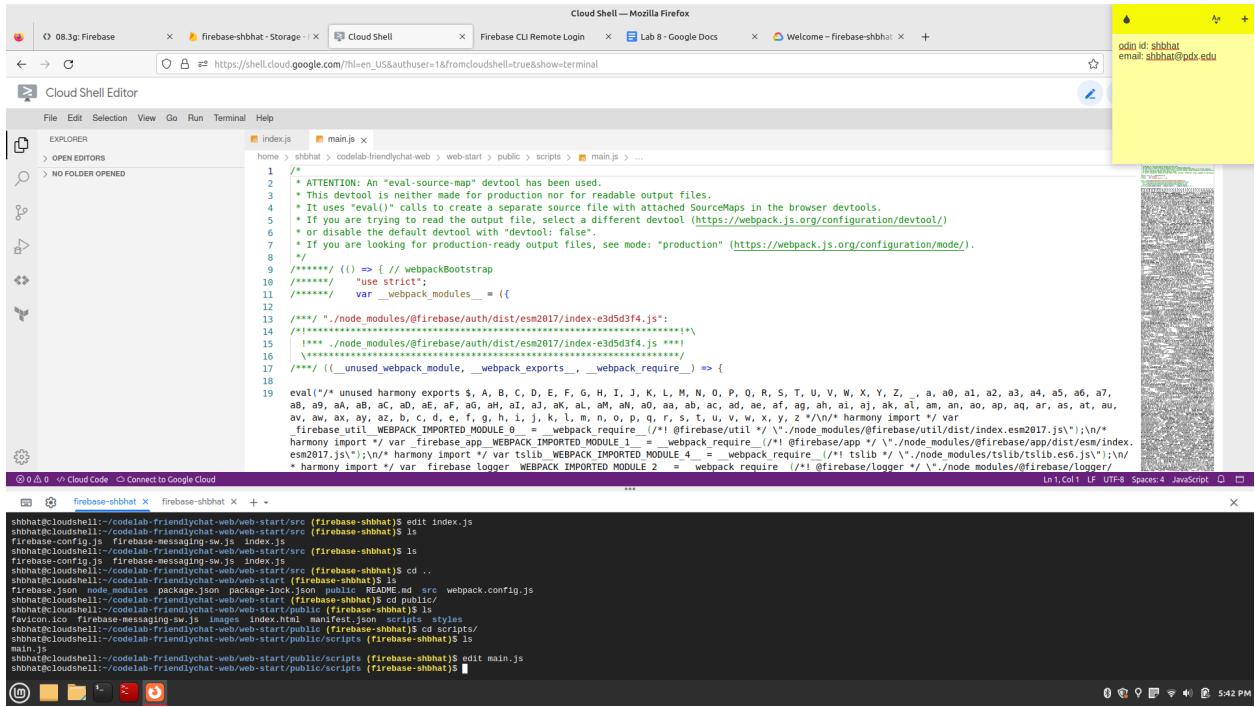
Authorized domain	Type
localhost	Default
fir-shbhat.firebaseio.com	Default
fir-shbhat.web.app	Default
cloudshell.dev	Custom

A yellow sticky note is overlaid on the right side of the screen, containing the following text:

odin id: shbhat
email: shbhat@pdx.edu

Card 8

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



Card 12

8.3.3 Answer the questions.

- What missing functions deal with user authentication?

`signIn()`: Implements Google Sign-In functionality using Firebase authentication.

`signOutUser()`: Signs out the user from Firebase.

`initFirebaseAuth()`: Initializes Firebase authentication and subscribes to the user's signed-in status.

`getProfilePicUrl()`: Returns the signed-in user's profile picture URL.

`getUserName()`: Returns the signed-in user's display name.

`isUserSignedIn()`: Returns true if a user is signed-in, otherwise false.

`authStateObserver(user)`: Triggers when the auth state changes, such as when the user signs in or signs out

- What missing functions deal with sending and receiving messages?

`saveMessage(messageText)`: Saves a new message to Cloud Firestore.

`loadMessages()`: Loads the chat message history and listens for new messages

`saveImageMessage(file)`: Saves a new message containing an image to Firebase, including uploading the image to Firebase Storage.

deleteMessage(id): Deletes a message from the UI

`createAndInsertMessage(id, timestamp)`: Creates and inserts a message in the UI.

`displayMessage(id, timestamp, name, text, picUrl, imageUrl)`: Displays a message in the UI.

Card 13

8.3.3 Answer the questions.

- 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`

Card 16

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

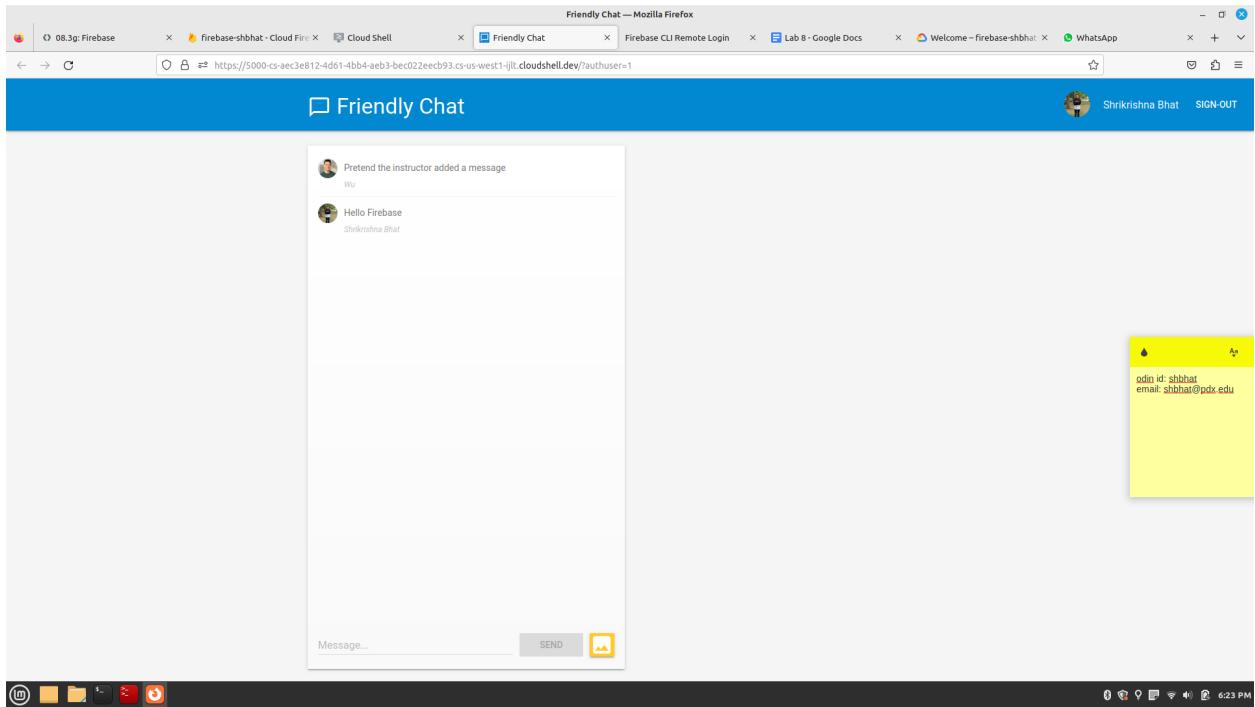
The screenshot shows the Firebase Cloud Firestore console in Mozilla Firefox. The URL is `https://console.firebaseio.google.com/u/1/project.firebaseio.shbhat.firebaseio/data/-2F5t0ba5aR0LrMPQVq3i2y`. The left sidebar shows project navigation with 'Extensions' selected. The main area displays a 'Cloud Firestore' view with a 'Data' tab selected. A message document is shown under the 'messages' collection. The document ID is `5t0ba5aR0LrMPQVq3i2y`. The document fields are:

- `name`: "Shrikrishna Bhat"
- `profilePicUrl`: "`https://lh3.googleusercontent.com/a/AGNmyY_Fc4LBWYKewdmnTKEaGCsdZs0kRyhKFBlgvE=s96-c`"
- `text`: "Hello Firebase"
- `timestamp`: May 23, 2023 at 6:17:53 PM UTC-7

A yellow box highlights the right side of the document card, specifically the field names and their values.

Card 17

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



Card 18

8.3.6 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>

Card 19

8.3.7 Answer the questions

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

Text document has name, profilePicUrl, text, timestamp

Image document has imageUrl, name, profilePicUrl, storageUri, timestamp

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

URL:

https://firebasestorage.googleapis.com/v0/b.firebaseio-shbhat-388216.appspot.com/o/Q39L0uuZ3MShdEKHT6ZkJfMLN33%2FMdmaNV1uIycSGkKHjIfT%2Fzombie-949916_1280.jpg?alt=media&token=be51a4a2-51a7-4133-95c6-ebf67144fb59

Storage Location:

gs://firebase-shbhat-388216.appspot.com/Q39L0uuZ3MShdEKHT6ZkJfMLN33/MdmaNV1uIycSGkKHjIfT/zombie-949916_1280.jpg

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

The screenshot shows the Firebase Storage console interface. On the left, there's a sidebar with project navigation and settings. The main area displays a list of files under a specific storage reference. One file is selected: "zombie-949916_1280.jpg". To the right of the file list is a preview of the image, which is a person lying down. Below the preview, detailed metadata for the file is shown, including its name, size (119,417 bytes), type (image/jpeg), creation date (May 29, 2023, 9:55:22 AM), and update date (May 29, 2023, 9:55:22 AM). At the bottom, there are sections for file location (storage location: gs://firebase-shbhat-388216.appspot.com/Q39L0uuZ3MShdEKHT6ZkJfMLN33/MdmaNV1uIycSGkKHjIfT/zombie-949916_1280.jpg) and access token (Access token: be51a4a2-51a7-4133-95c6-ebf67144fb59). A yellow sticky note is pinned to the top right of the interface, containing the text: "odin id: shbhat jemail: shbhat@pdx.edu".

Card 20

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

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

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

