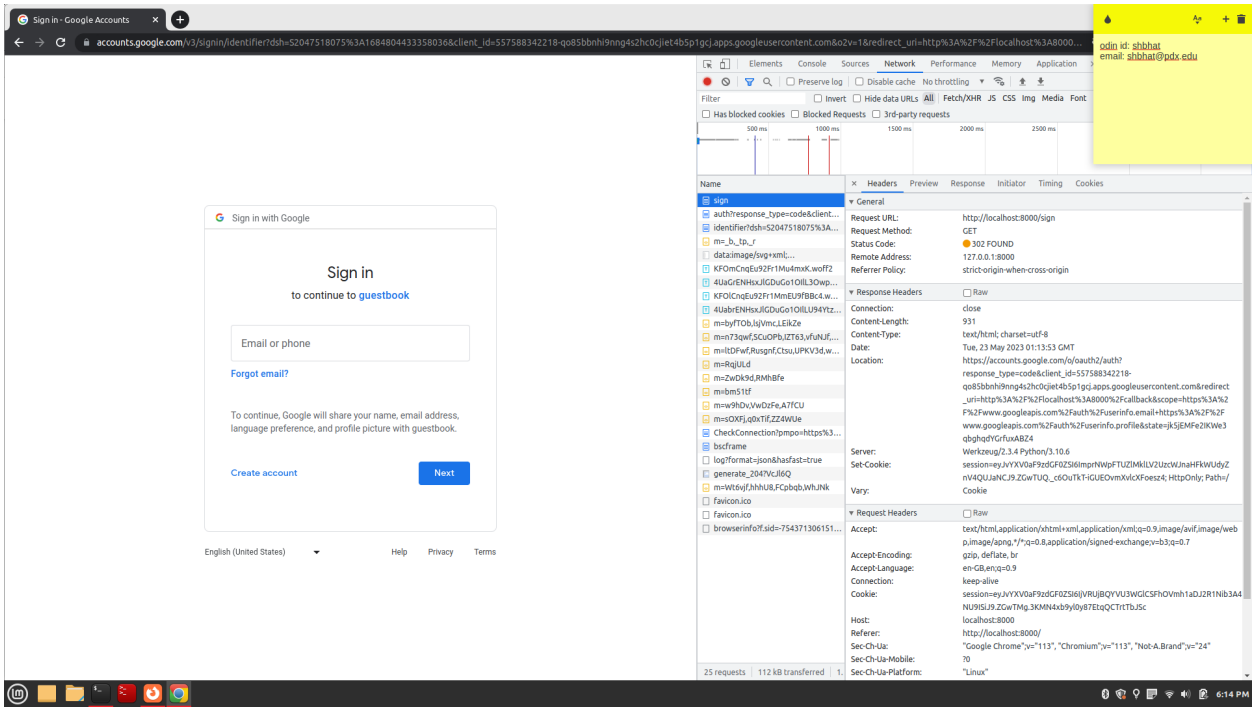
Lab Notebook Week 8

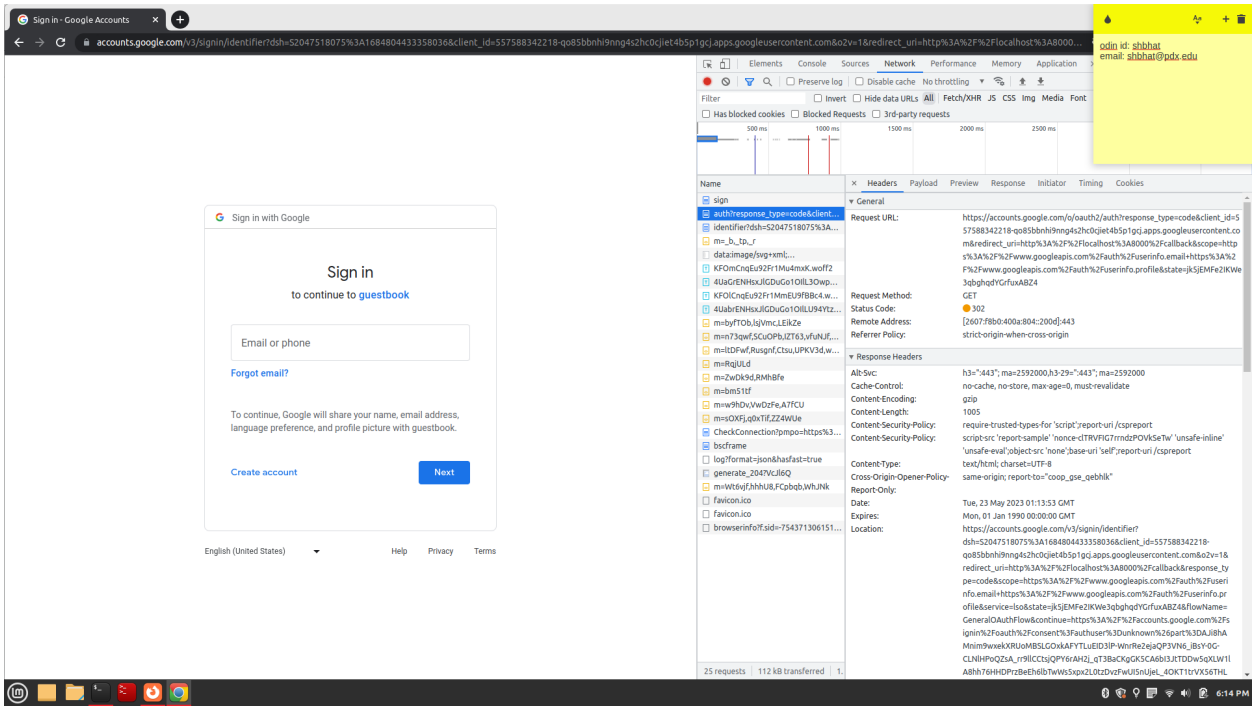
Submitted by: Shrikrishna Bhat

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.**



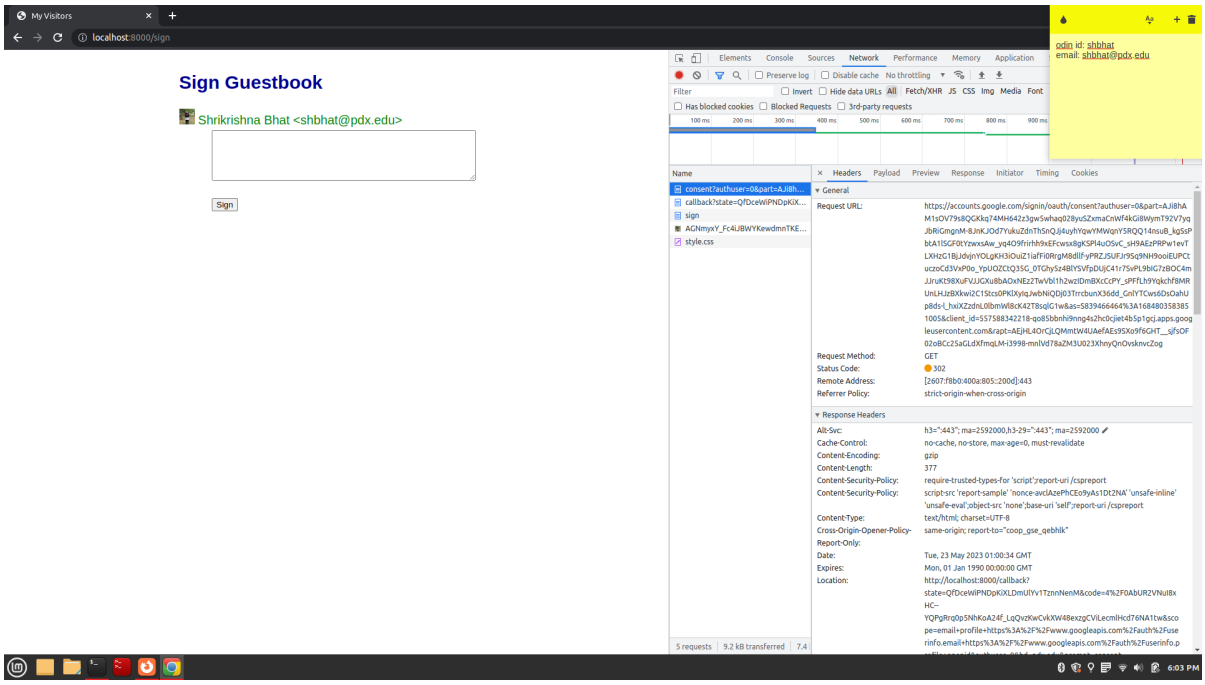


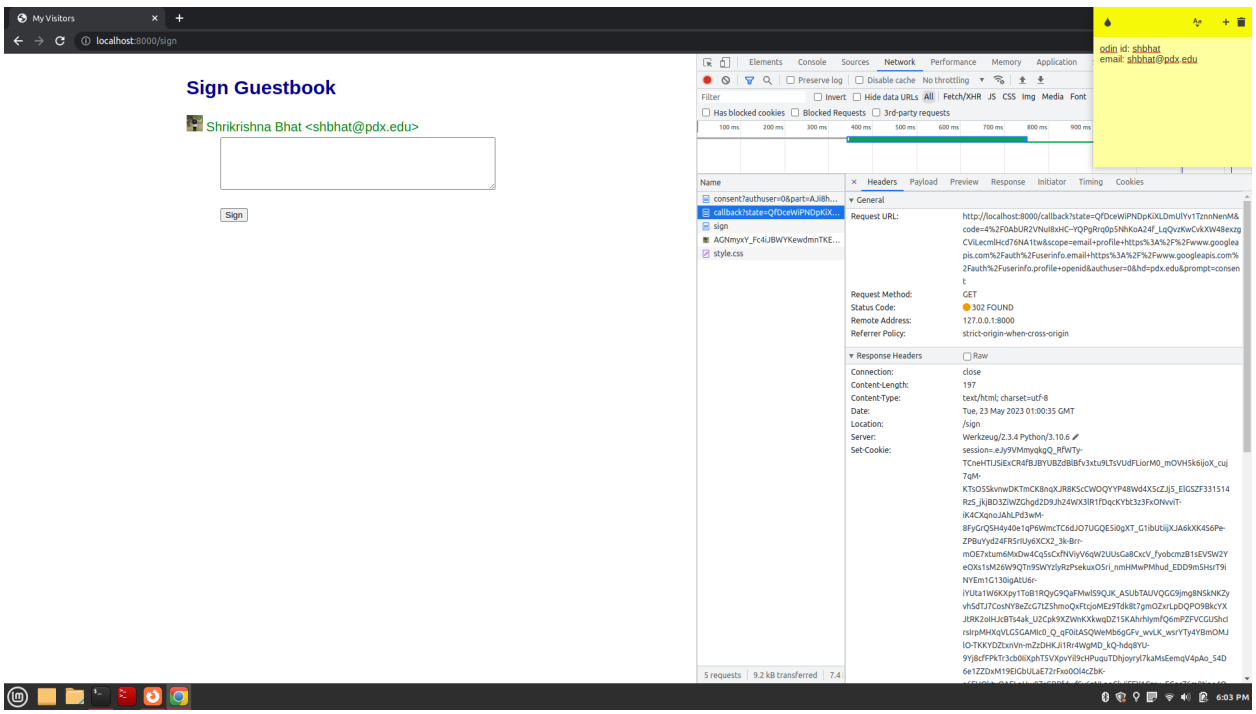
8.1.2 Based on the description of the source code, what lines of code in our application are responsible for the second request shown?

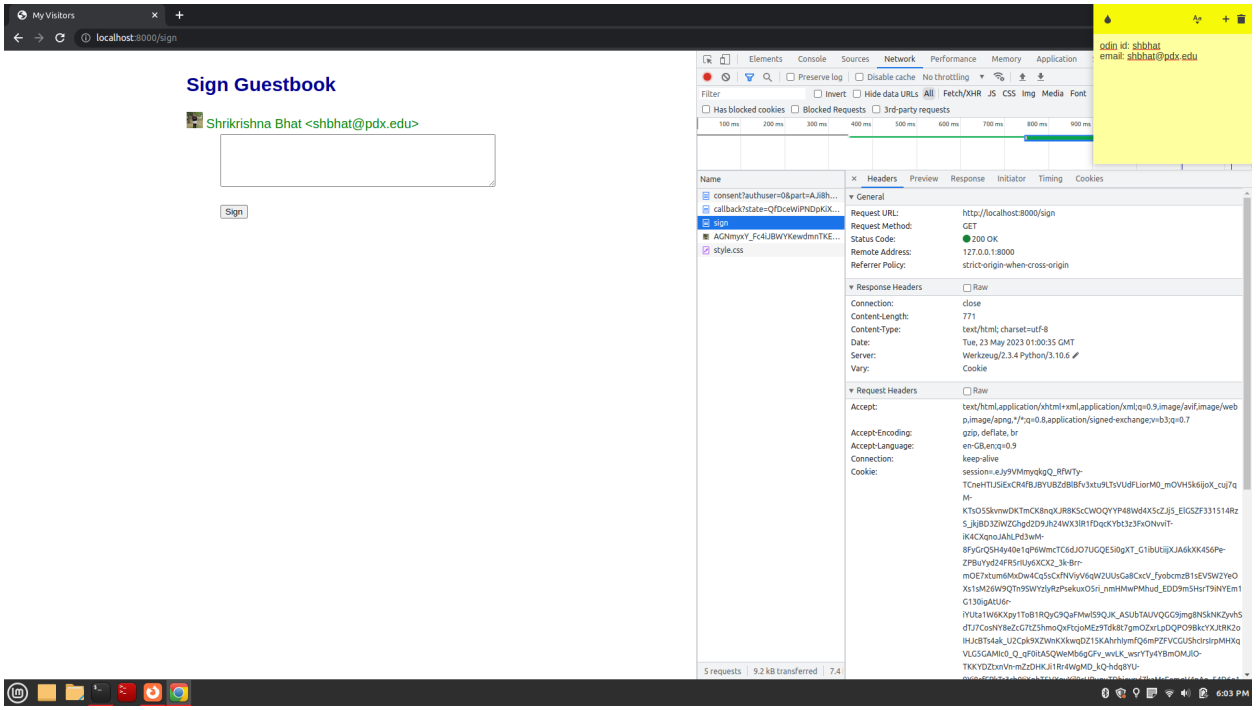
This code is found in the sign.py and the lines which say are
authorization_url, state = google.authorization_url(authorization_base_url)
session['oauth_state'] = state
return redirect(authorization_url)

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?

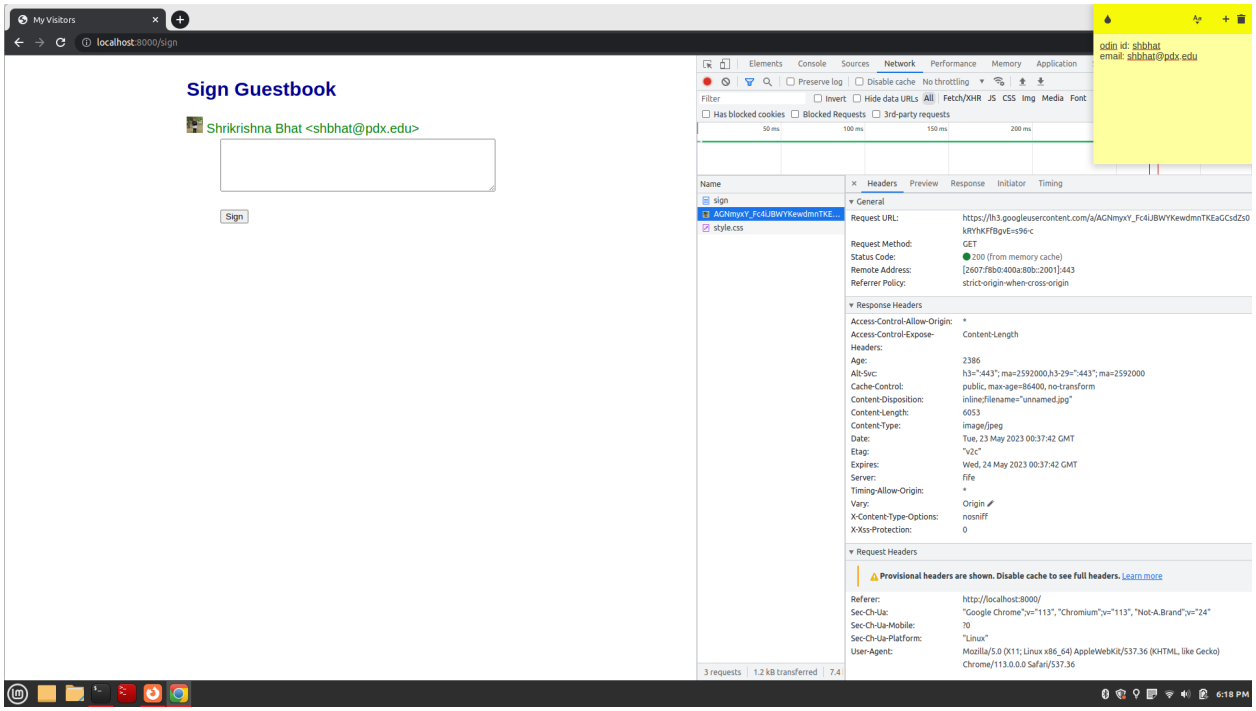
Answer: User is sent to /sign location once he successfully signs in and now the user can sign the guestbook.



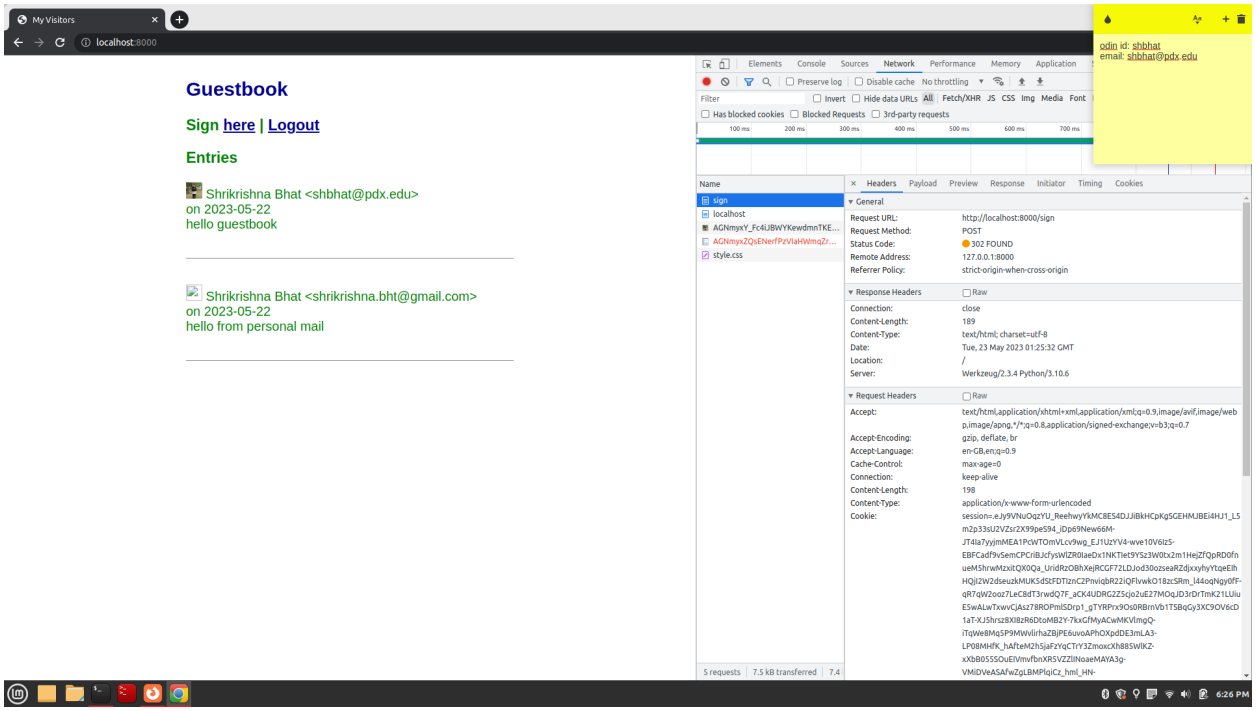




8.1.4 Find the request within Developer Tools that fetches the embedded image and take a screenshot of its URL.

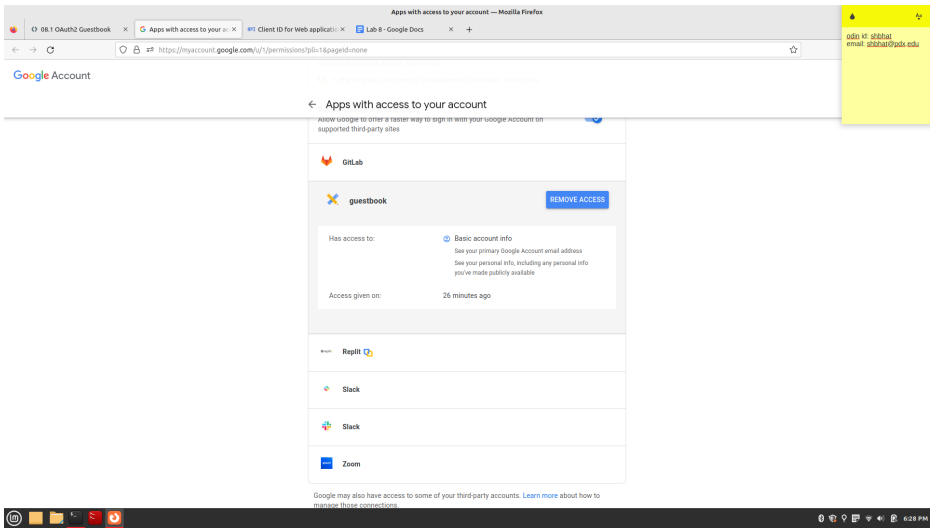


8.1.5 Take a screenshot showing multiple authenticated accounts have been able to sign the Guestbook.



Card 13

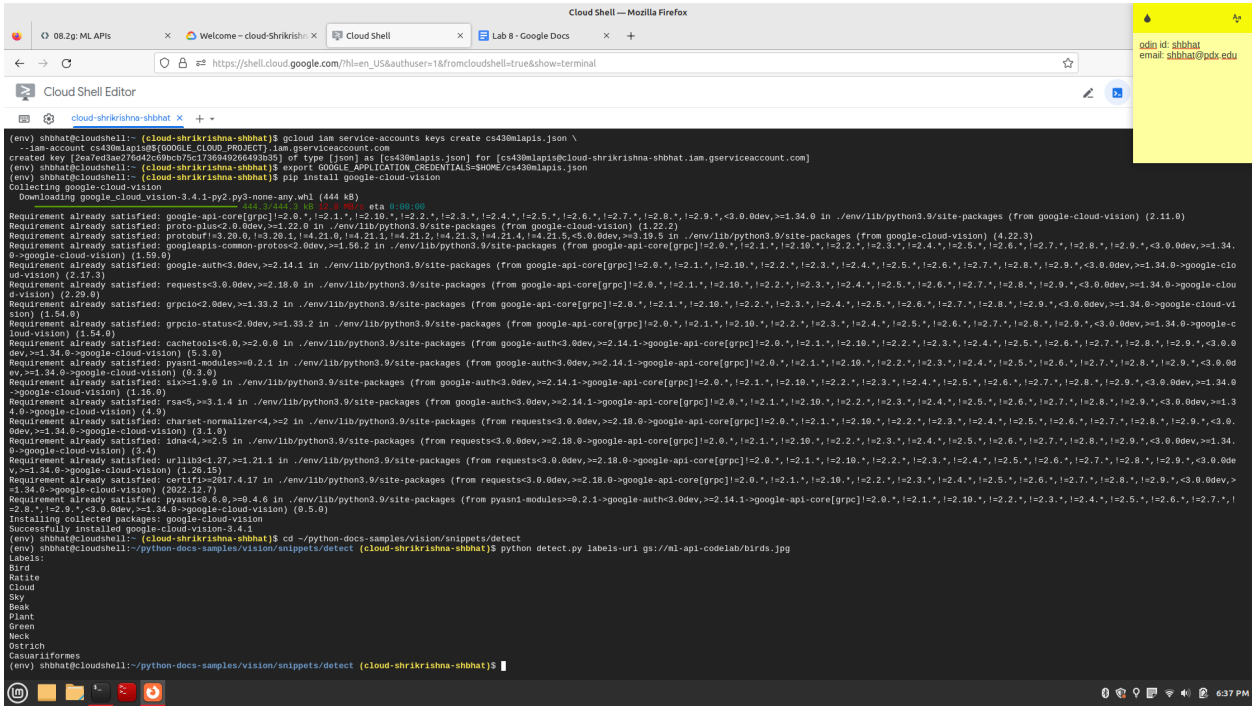
Take a screenshot of the expanded information that includes your OdinId for your lab notebook.



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?**

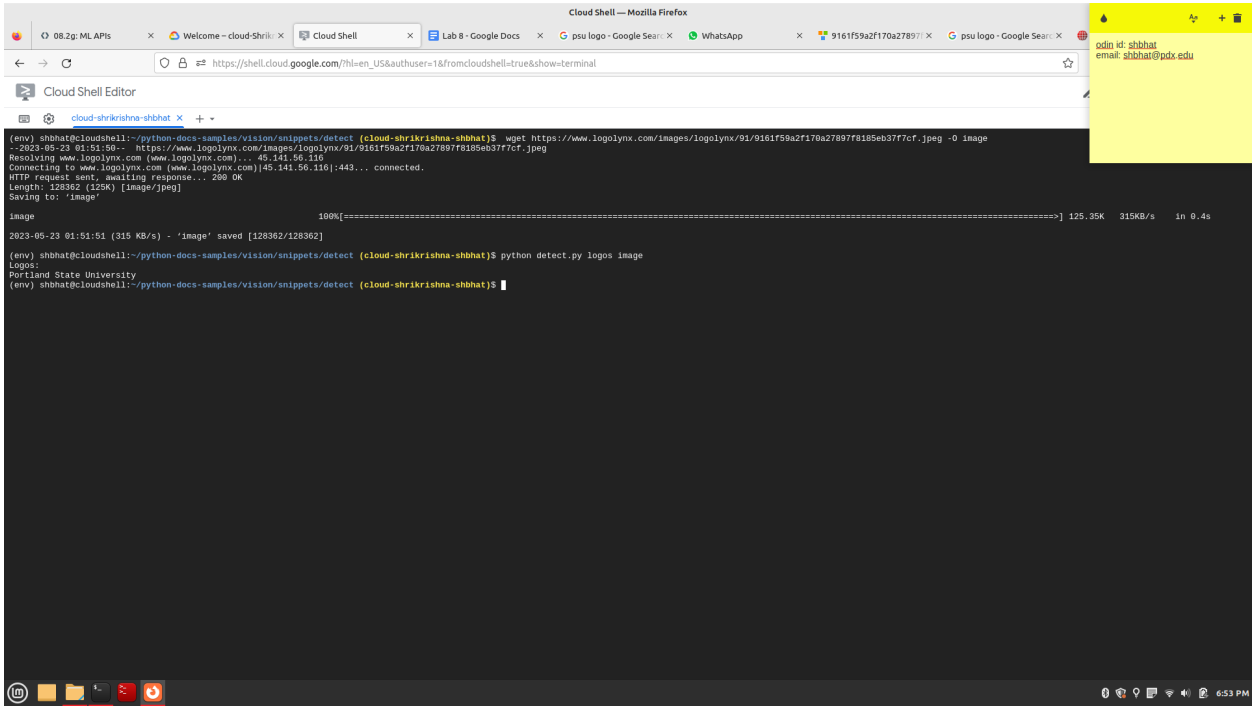
  client.label_detection(image=image)

- **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

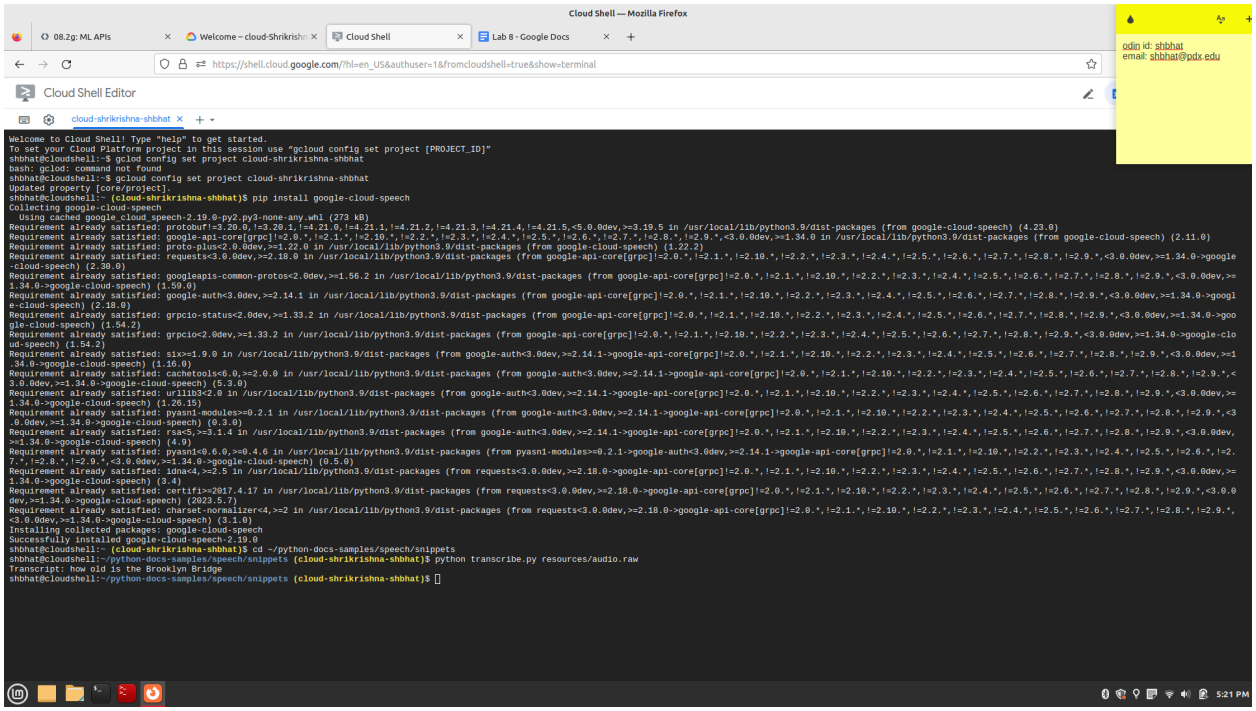


8.2.4 What method is invoked in the Vision client to perform the detection?

logo_detection ()

Card 4

8.2.5 Show the output for your lab notebook

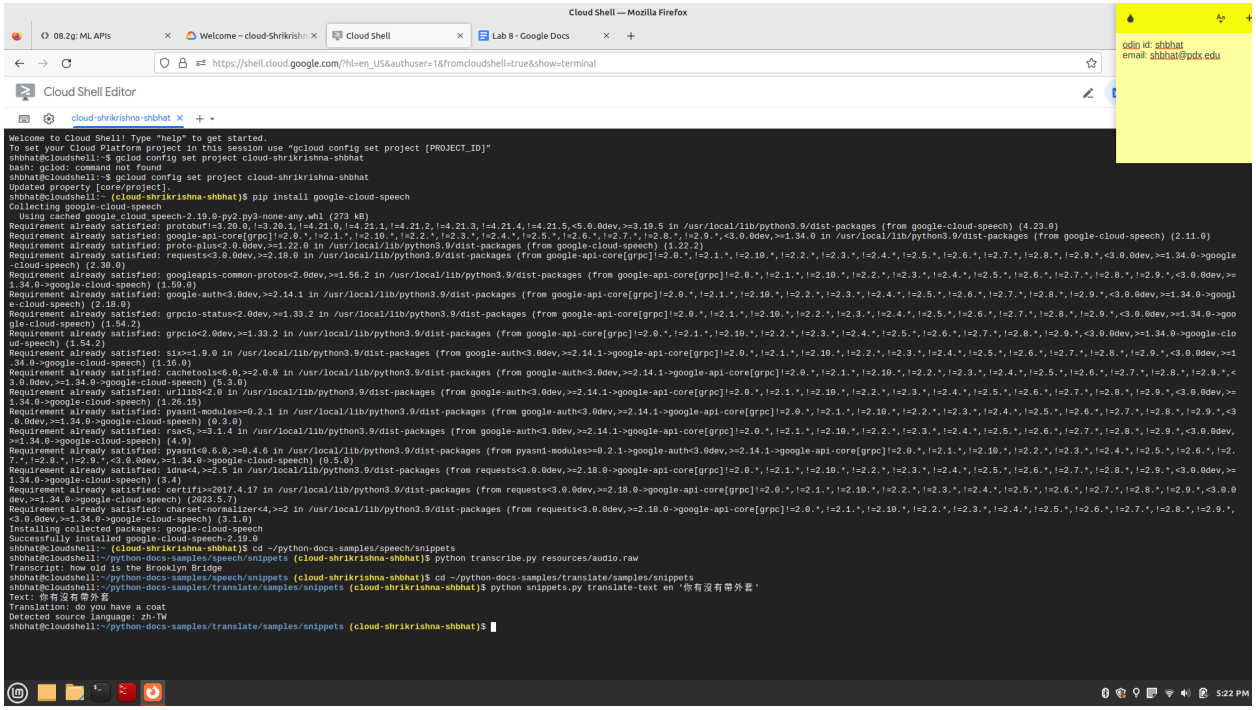


8.2.6 Answer the questions:

- **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?**

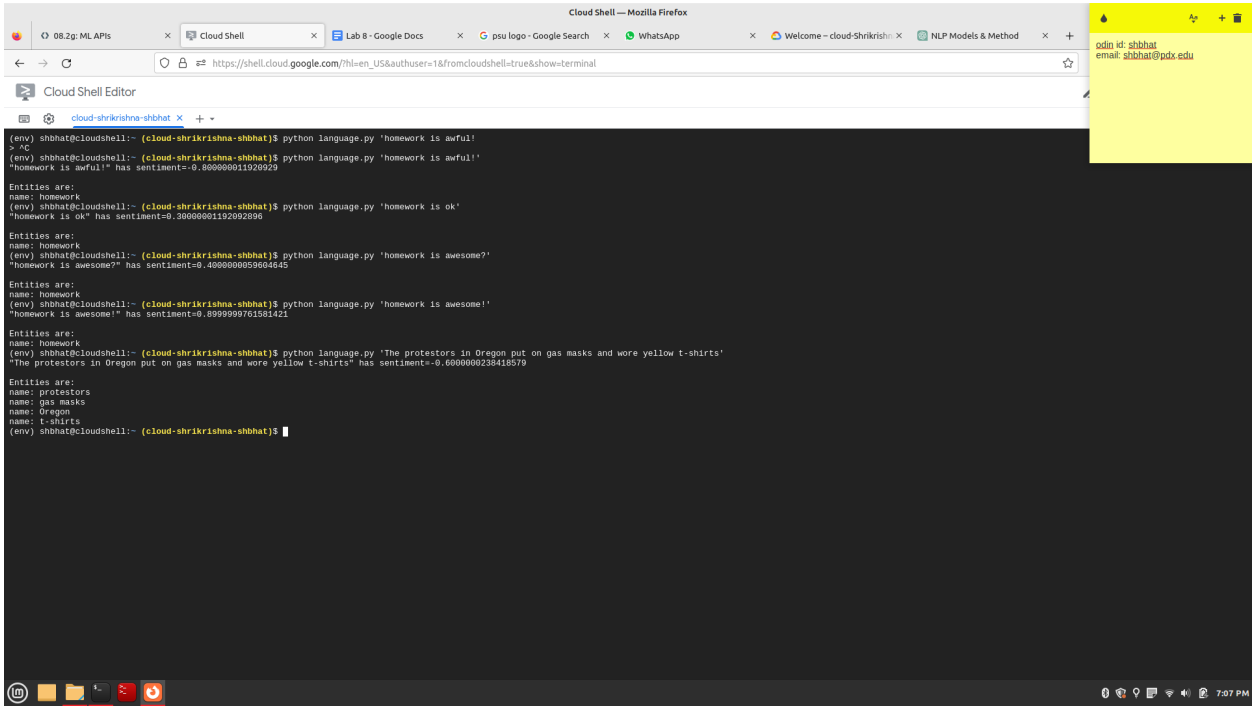
  translate_client.translate(text, target_language=target)

- **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**



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: entitities_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)

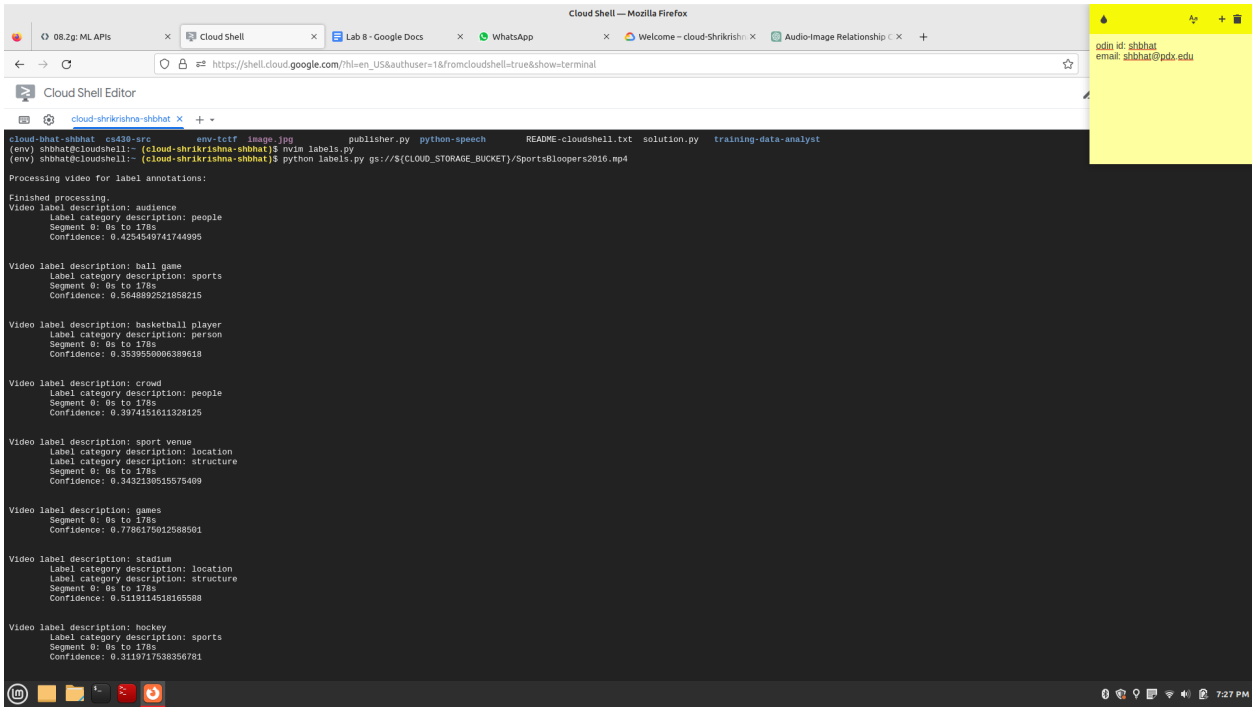
python solution.py de-DE gs://ml-api-codelab/de-ball.wav gs://ml-api-codelab/football.jpg
In the compare_audio_to_image function, we can try adjusting the confidence threshold for considering a match between entities and labels. Currently, it only checks if the entity name exactly matches a label. We 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?

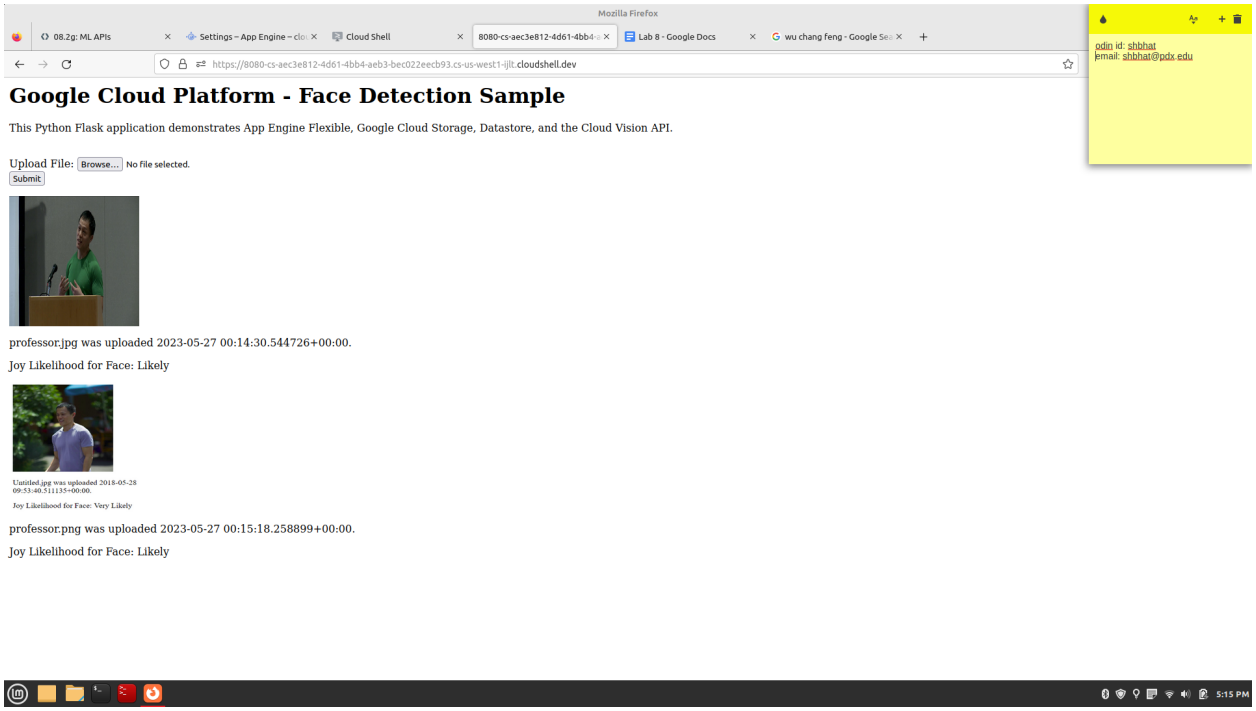


8.2.13 What is the name of the client class in the package that is used?
videointelligence.VideoIntelligenceServiceClient.

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

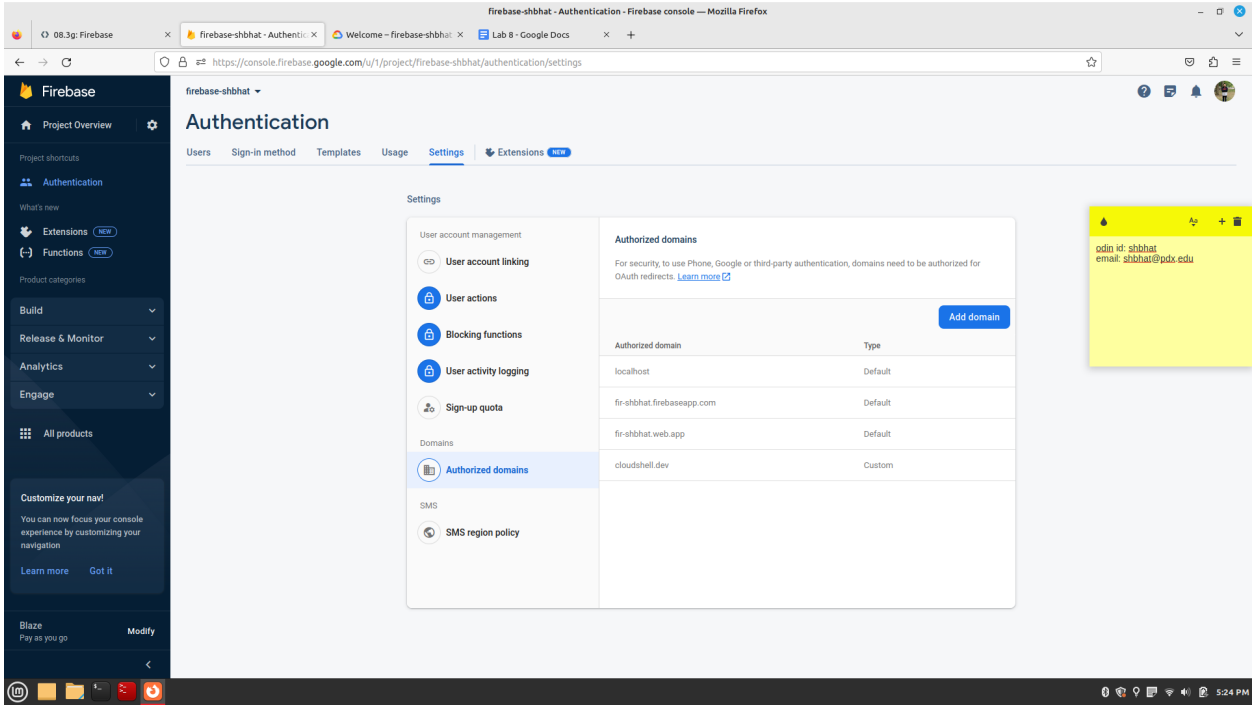
Card 4

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

localhost

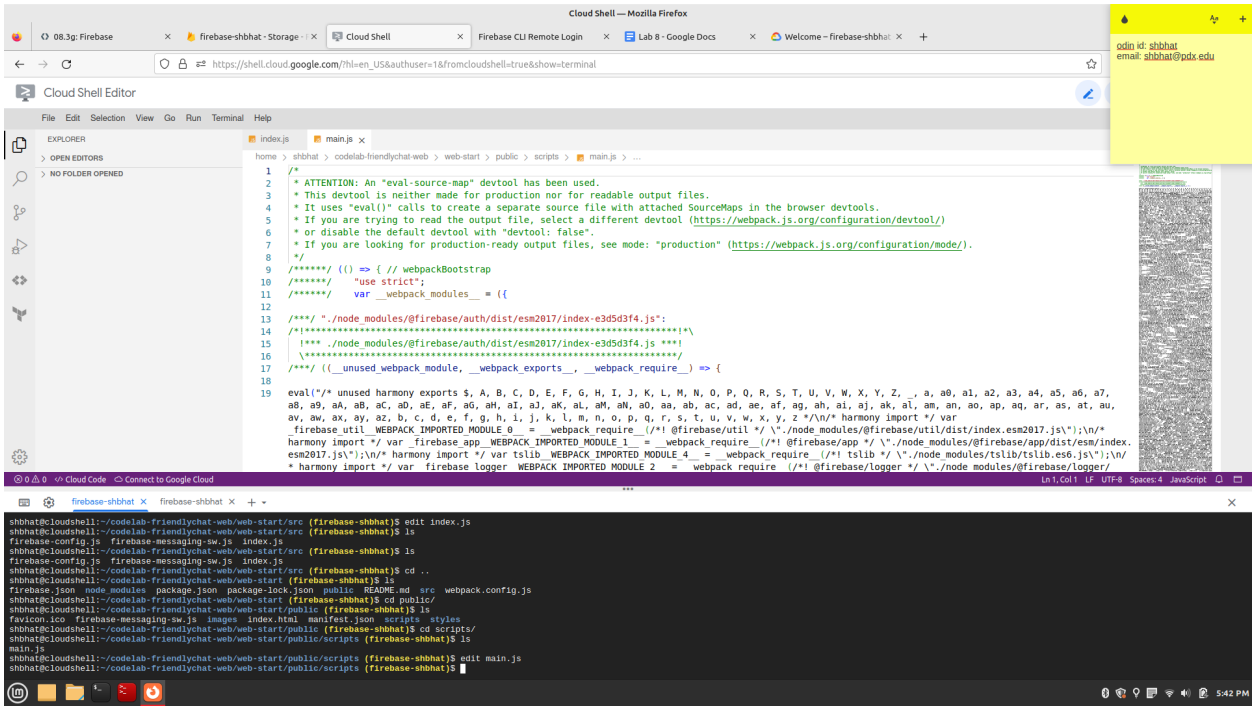
fir-shbhat.firebaseapp.com

fir-shbhat.web.app



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?**

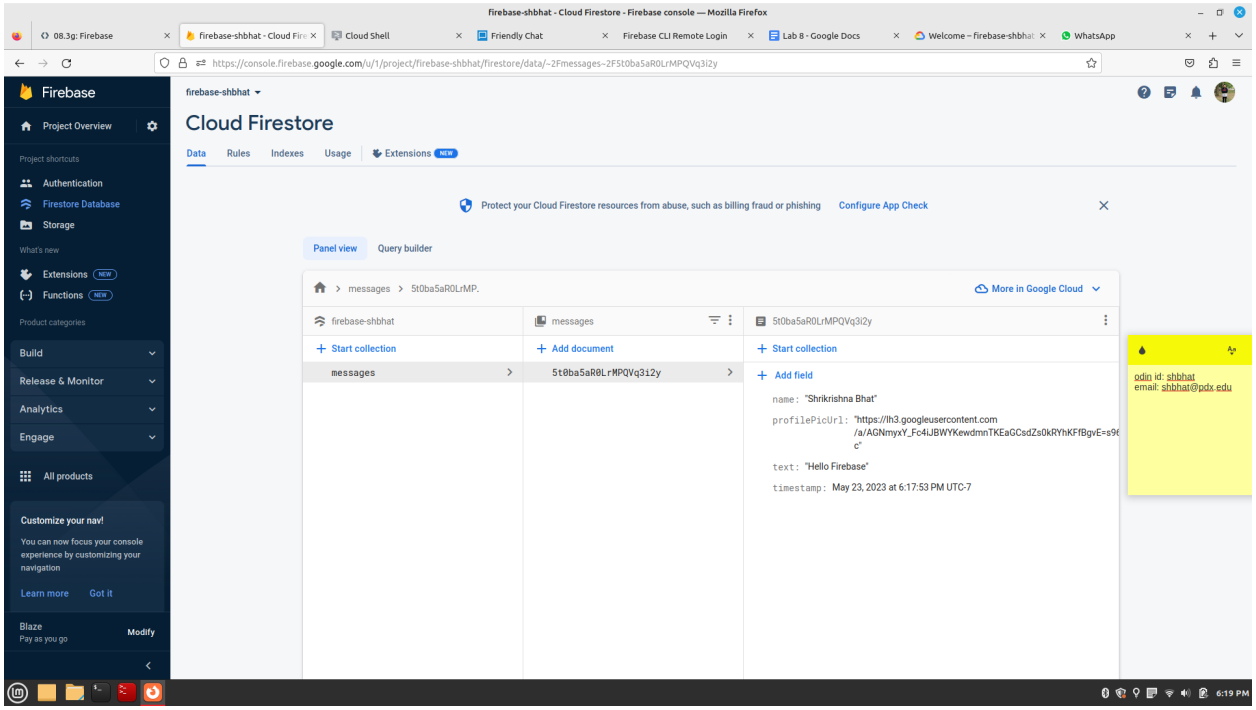
  userNameElement, userPicElement, signOutButtonElement

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

  signInButtonElement

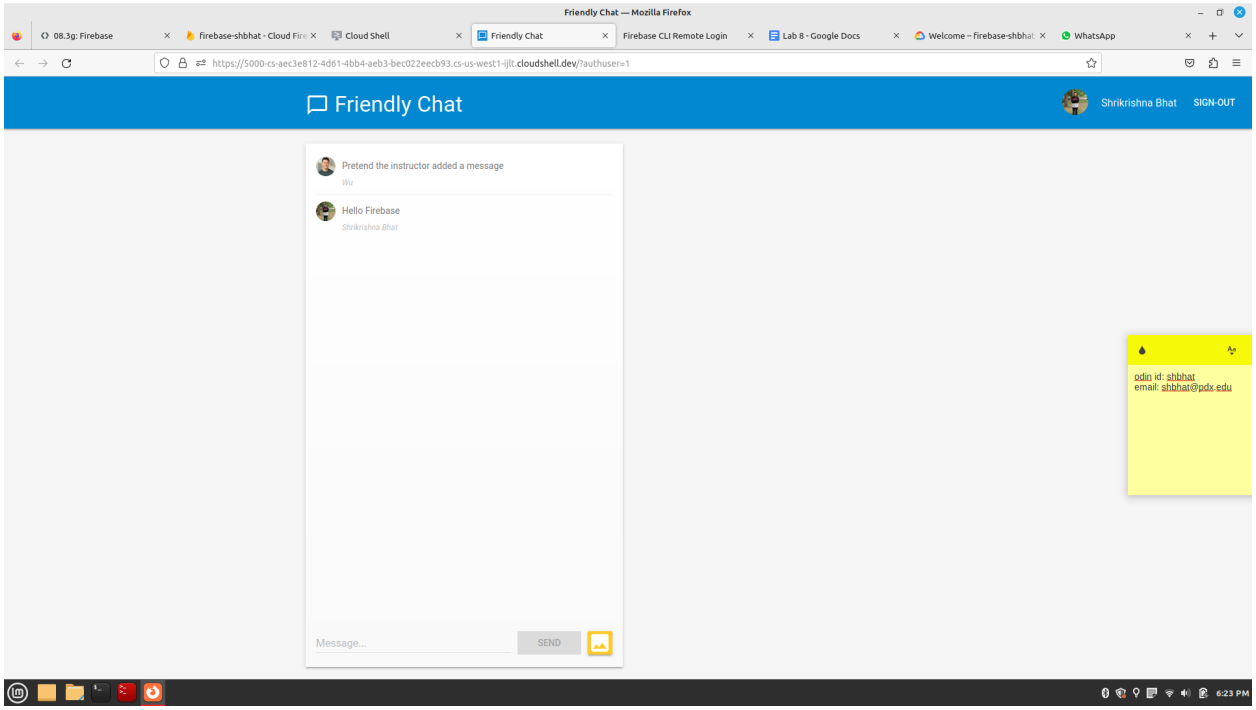
Card 16

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



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

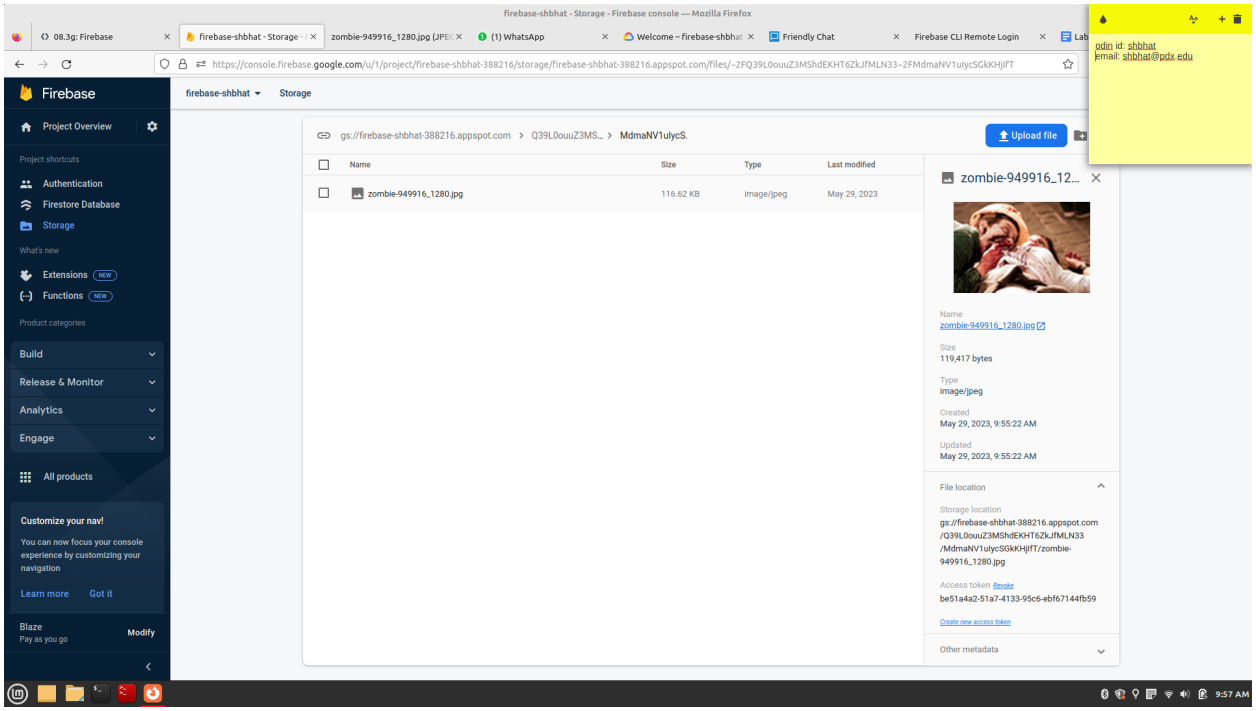
  Image document has imageUrl, name, profilePicUrl, storageUri, timestamp

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

  URL:
  https://firebasestorage.googleapis.com/v0/b/firebase-shbhat-388216.appspot.com/o/Q39L0ouuZ3MShdEKHT6ZkJfMLN33%2FMdmaNV1uIycSGkKHjIfT%2Fzombie-949916_1280.jpg?alt=media&token=be51a4a2-51a7-4133-95c6-ebf67144fb59

  Storage Location:
  gs://firebase-shbhat-388216.appspot.com/Q39L0ouuZ3MShdEKHT6ZkJfMLN33/MdmaNV1uIycSGkKHjIfT/zombie-949916_1280.jpg

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



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.