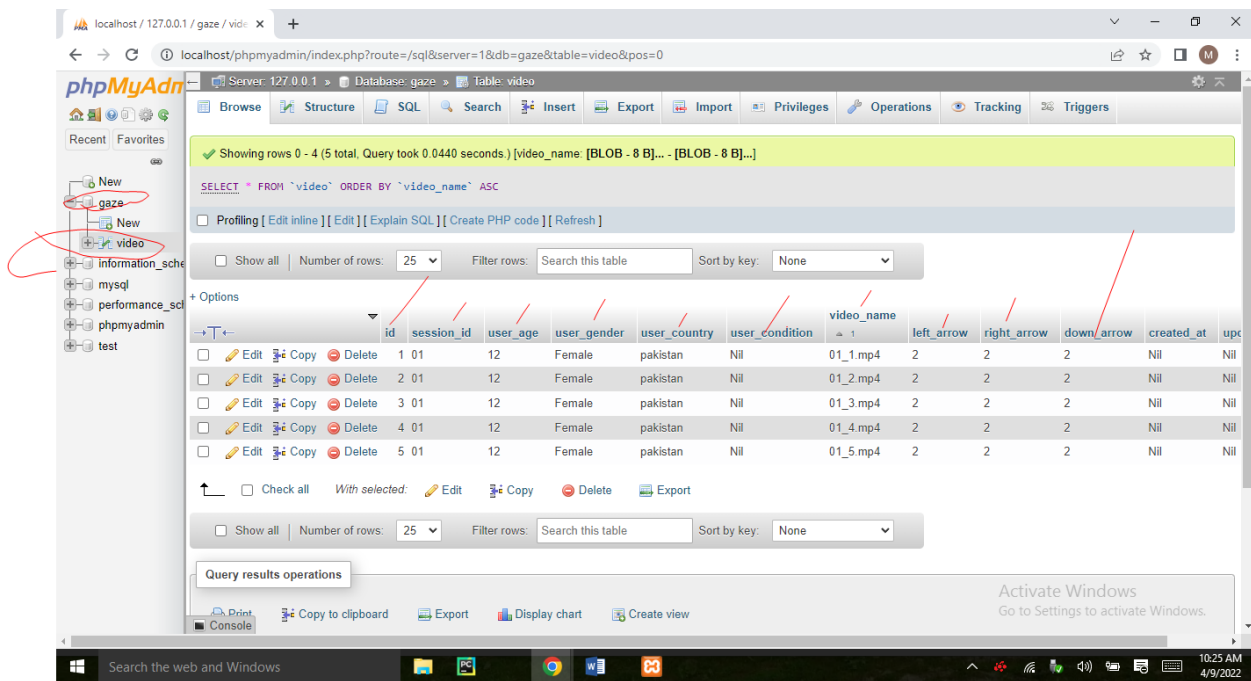
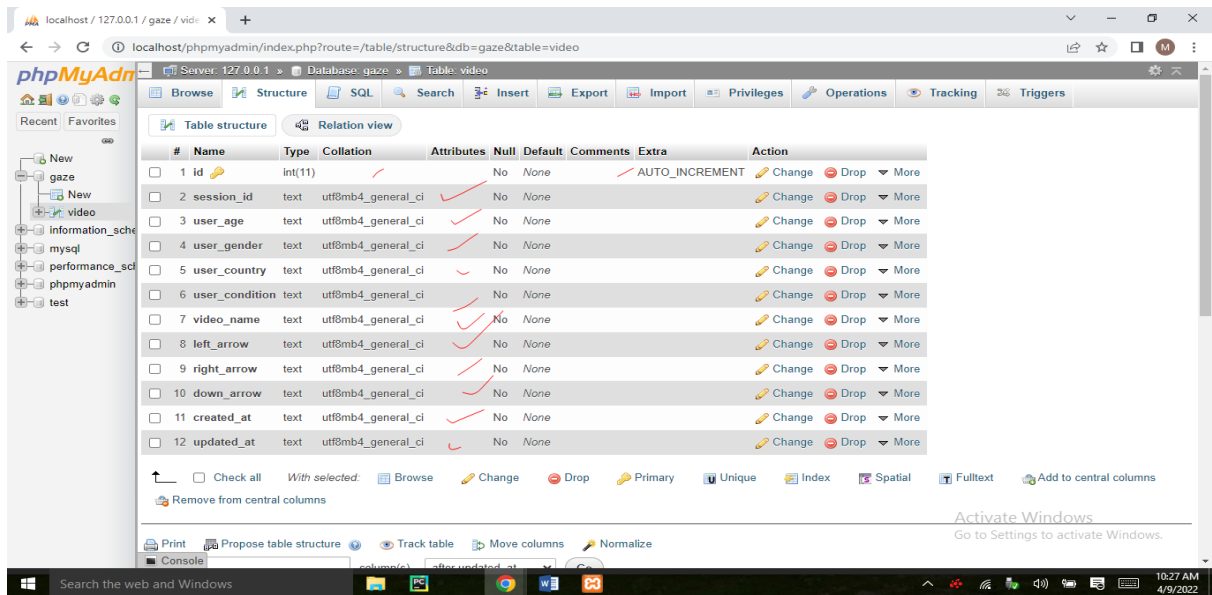


The database name is “Gaze” and table name is “video” as shown below:



- The structure view of table is given below:



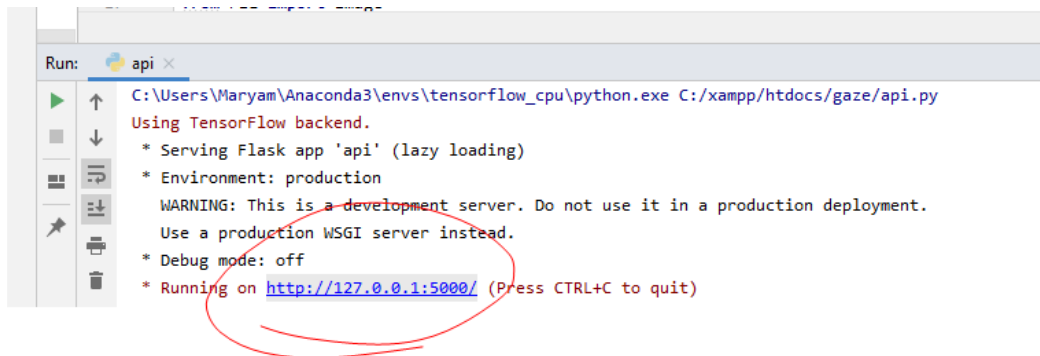
After creating the database, directory structure is created for FLASK app. For FLASK there are two folders i-e “static” and “templates” folder.

- In Flask, the static folder contains assets used by the templates, including CSS files, JavaScript files, and images.
- While in the templates folder there is only templates i-e “HTML page”.

The API.py is a python file which is main file that contains code to get output from trained model. All these files are placed in “htdocs” folder in xampp.

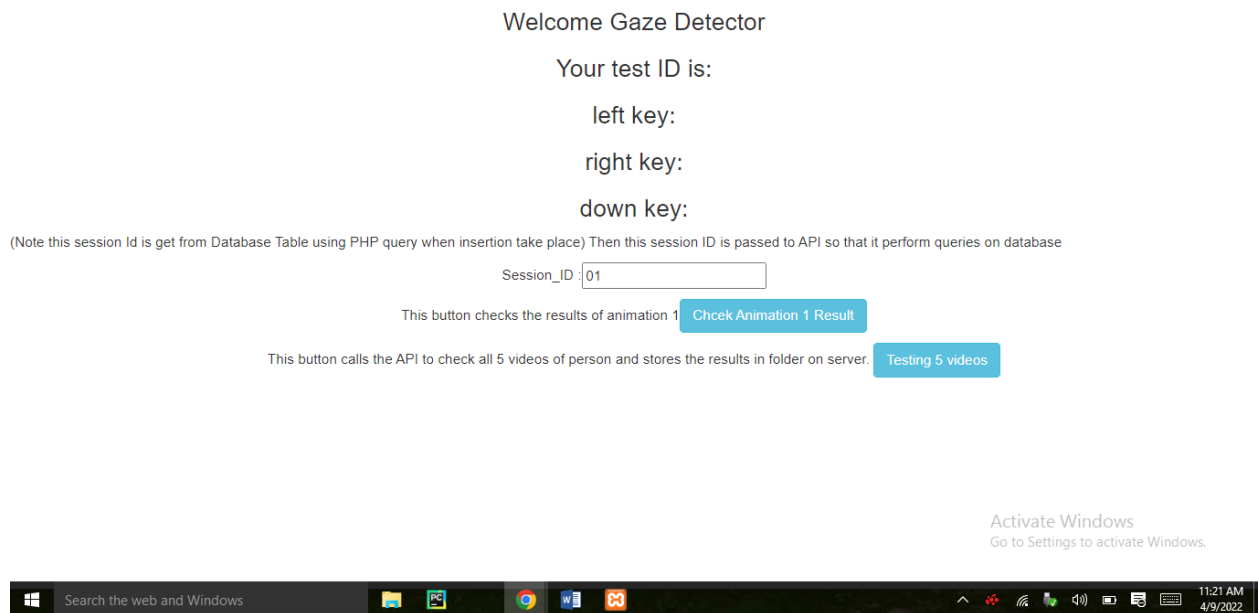
.....

- 1) In python “PYCHARM IDE”, run the “API.py” file. It generates the following link.

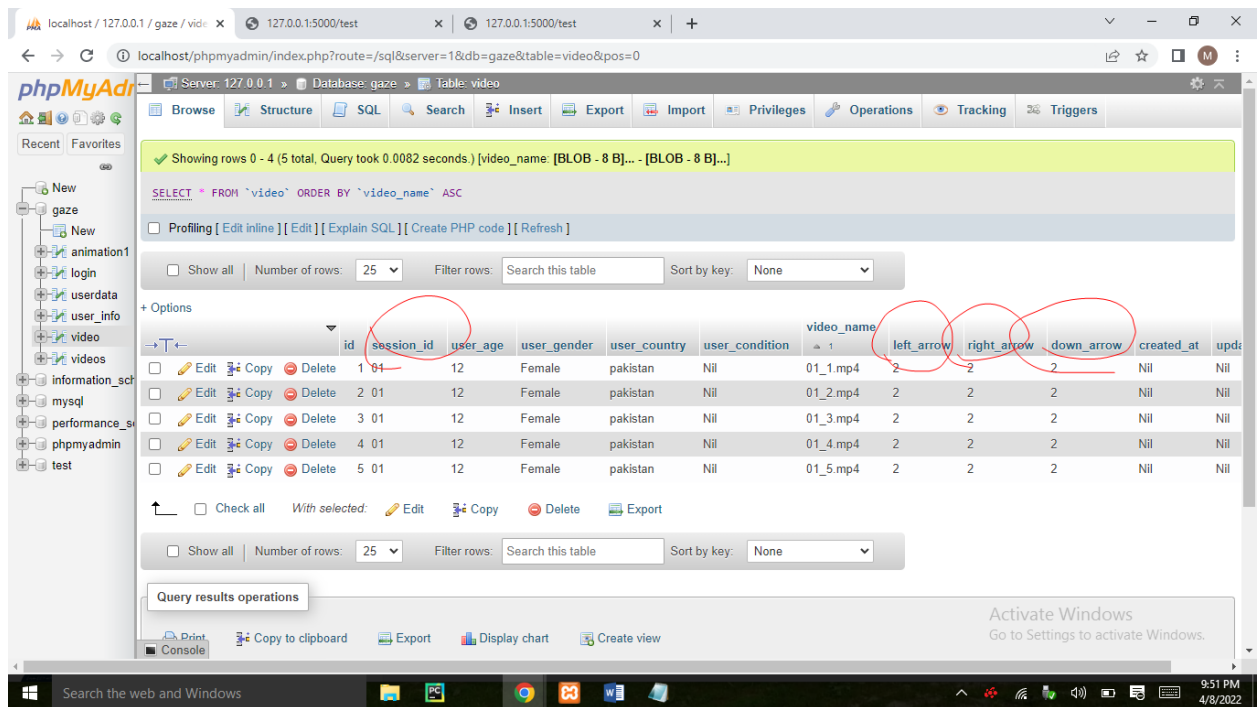


```
Run: api x
C:\Users\Maryam\Anaconda3\envs\tensorflow_cpu\python.exe C:/xampp/htdocs/gaze/api.py
Using TensorFlow backend.
* Serving Flask app 'api' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- 2) On this link, the main page of website is displayed. This *mainpage.html* is present in templates folder.



- 3) This page contains two buttons. The first button is used to check the result of animation 1.
- 4) When this button is clicked, then API used “session_id” (that is passed it) to get values of “left arrow key press time”, “right arrow key press time”, “down key arrow time” as shown below:



- 5) Notice in above table, I have use the “session_id” to get values of keys.
- 6) Now through this “session_id”, we get values of keys time from database to process it and displayed on main web page. [LIMIT is used in query because we need only one record.]

```
#####Query To Get values of time for left key #####
conn = mysql.connect()
cursor = conn.cursor()
cursor.execute("""SELECT left_arrow FROM video WHERE session_id= %s LIMIT 1""", session["id"])
left = cursor.fetchone()
left = str(left)
left = re.sub("[()]", "", left)
left = re.sub(",", "", left)
left = re.sub("'", "", left)

conn.commit()
cursor.close()
```

The table is as follows. In below table, it is observed that for a particular session the values of left, right, and down arrow keys are same (*Because the animation 1 is tested only one time while the videos for next 5 animation is recorded and each video is saved with different name*). In all these rows, the value is repeated. [Only the video name (column) is different]. Due to this, I used LIMIT in query.

	id	session_id	user_age	user_gender	user_country	user_condition	video_name	left_arrow	right_arrow	down_arrow
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	01	12	Female	pakistan	Nil	01_1.mp4	2	2	2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	01	12	Female	pakistan	Nil	01_2.mp4	2	2	2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	01	12	Female	pakistan	Nil	01_3.mp4	2	2	2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	01	12	Female	pakistan	Nil	01_4.mp4	2	2	2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	01	12	Female	pakistan	Nil	01_5.mp4	2	2	2

- 7) After getting keys values, perform subtraction of actual and pressed key times and displayed on web page.

Welcome Gaze Detector

Your test ID is: 01

left key: 1

right key: 4

down key: 9

(Note this session Id is get from Database Table using PHP query when insertion take place) Then this session ID is passed to API so that it perform queries on database

Session_ID:

This button checks the results of animation 1

This button calls the API to check all 5 videos of person and stores the results in folder on server.



- 8) Now we check the remaining 5 animations. When user press the button “Testing 5 videos” as shown above. Then API through query first check that either all 5 videos are recoded by user or not. If yes it call predict () function. This function use videos from the folder present on server that contains all five videos of user. **Note: “While saving the videos on folder, attach “session_id” with filename of videos so that API correctly fetches the videos from folder of current “session”. E-g videos should be save as:**

- 01_1.mp4
- 01_2.mp4
- 01_3.mp4
- 01_4.mp4
- 01_5.mp4

01_1.mp4 where “01” is session_id and ‘1.mp4” indicates that it is video of first animation.

- 9) The query to check that all 5 videos are present or not is given below:

```

@app.route('/test', methods=['GET', 'POST'])
def test():
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("""SELECT * FROM video WHERE session_id= %s""", session["id"])
    data = cursor.fetchall()
    print(len(data))
    # check whether user has done 5 experiments or not (i.e 5 videos)
    if len(data) == 5:
        conn.commit()
        cursor.close()
        filename10,filename11,filename12,filename13,filename14,filename15,filename16,filename17,filename18,filename19,filename20,filename21, csv_file1, csv_file2, \
        csv_file3, csv_file4, csv_file5, csv_file6, csv_file7, csv_file8, csv_file9, csv_file10, csv_file11, csv_file12=predict()

```

- 10) When there are 5 videos then, predict() function is called. Otherwise, it is not called.
- 11) This predict() picks videos from folder and returns total 24 files out of which 12 files are graphs and 12 files are csv files.
- 12) After returning, we check that all 24 files are successfully written or not as shown below:

```

1414 # To check whether all files of the current session of the user are saved in directory or no
1415 if os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/'+filename10) \
1416     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/'+filename11) \
1417     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/'+filename12) \
1418     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename13) \
1419     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename14) \
1420     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename15) \
1421     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename16) \
1422     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename17) \
1423     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename18) \
1424     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename19) \
1425     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename20) \
1426     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename21) \
1427     and os.path.isfile('C:\\xampp\\htdocs\\gaze\\static\\uploads/' + filename21) \
1428     and os.path.isfile(csv_file1) \
1429     and os.path.isfile(csv_file2) \
1430     and os.path.isfile(csv_file3) \
1431     and os.path.isfile(csv_file4) \
1432     and os.path.isfile(csv_file5) \
1433     and os.path.isfile(csv_file6) \
1434     and os.path.isfile(csv_file7) \

```

- 13) When this “if” condition is “True” then, API returns the path of these 24 files as JSON object as shown below:

14) Above is JSON object, in which “key” is name of file (i-e what files represents”) while the value is path where these 24 files are stored.

Note: We have only one folder in which all graphs and csv files of all “session_id” (i-e persons) are saved. Hence, the names of these csv and graph (.png) files are attached with “session_id” so that files of each “session_id” is easily distinguishable. E-g

- 01_left_eye_result.png
- 02_left_eye_result.png
- In above example the first two characters of filename denotes the "session_id". So that, we distinguish that “01_left_eye_result.png” is the graph of user (i-e session_id=01)