**Team: Art Nguyen & Shreejit Poudyal**

Deliverables: readme.txt, main.py, loadData.py, DELETE.py, Project1.db, check.html.

# I) Front-end

On our front end, we created a bare-bone check.html that includes text boxes for the user to enter their email and password and a button called “submit” to check if the entered email and password are in the database. Hitting the submit button after entering credentials will load a new page with the response of whether the email + password has been compromised, ONLY the email being compromised, or the email is safe(not in the database).

# II) Back-end

In part 2 of our project, we have switched over to using SQLite3 to improve the time it takes to run queries drastically. In our loadData.py file, if the table called “User” has not been created then it will be created with the same email = varchar(255) Primary key, password = varchar(255). It will then read the credentials files line by line, separating the email/password then hash them using sha256. The hashed email and password will then be loaded into the “Project1” database’s “User” table. If the email is a duplicate then it is ignored. Figure 1 shows

the total time it takes to go through ALL ~25 million entries.



Figure 1: Time elapsed to load into the database: 14:57 minutes

In our main.py file, before we launch the Flask website, we first fetch all the data from the Users table and load them into a key: value dictionary in Python. This task will take around 15 seconds as shown in Figure 2:



Figure 2: Time Elapsed to load from database to dictionary: 14 seconds

After loading all the data into the dictionary, the website’s address will be listed in the terminal and we can start testing if the email + password is compromised or not. The entered email and password from the Flask website will be encrypted through sha256 to then be compared to the Keys inside our dictionary. The three scenarios will be discussed in the next section.

# III) Testing & Query Time

Due to us loading the database into a key: value dictionary, the search time for any query is **almost instant.**

In Figure 3 and Figure 4, we entered the exact same email and password from credentials1.txt

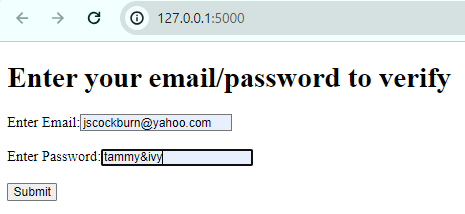


Figure 3: Entering compromised email and password

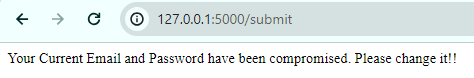


Figure 4: The result presented to us

In Figure 5 and Figure 6, we entered the exact same email from credentials1.txt but a wrong gibberish password



Figure 5: Entering compromised email but wrong password

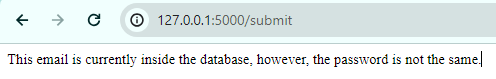


Figure 6: Result presented to us

In Figure 7 and Figure 8, we entered completely random emails and passwords.\

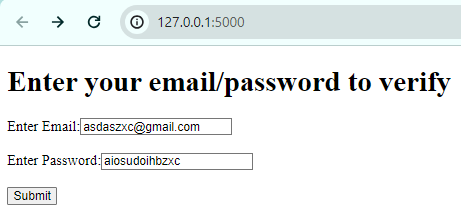


Figure 7: Entering random email and password



Figure 8: Result presented to us