Indian Institute of Technology Tirupati
CS3210 Computer Networks Laboratory
Lab #1 Report

S. Karthik Chandra
CS17B026

V Dheeraj
CS17B028

Motivation:

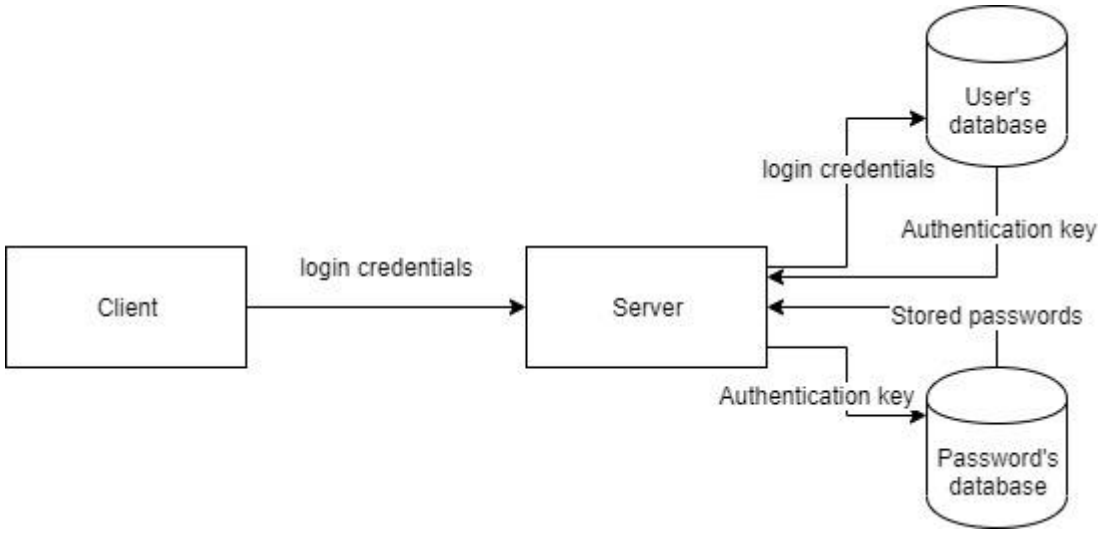
The majority of people use very weak passwords and reuse them on different websites. If you use the same login information everywhere, a leak at one website could give people access to all your accounts. For this purpose, a password manager is commonly used. Password managers store your login information for all the websites you use. We are going to implement a Password Manager using socket programming from scratch. The user will be able to view
and save all his passwords from a remote machine.

Salient features:

The main goal of the application is to create a network password manager which can store the passwords of many clients and serve their requests at once. This is achieved by creating master accounts for all the users through which they can log in and get access to their passwords.

* A multi-user password management facility using multi-threading.
* Two-factor authentication for all users.
* A database system in the backend to facilitate the storage.
* A friendly interface to make things simpler.

Design:



This application can handle multiple clients at once. This is achieved by having the main thread whose work is to create a thread each time a new client is connected. These threads send and receive the data between server and client to which they are attached. They also make database queries.

There are two databases in use, of which one stores all the credentials of the master accounts of the users along with a unique authentication key, and other stores the passwords and other required details.

When a user creates an account, his/her credentials are stored in the user's database and a unique authentication key is generated.

This authentication key cannot be seen by the users and hence the passwords cannot be directly accessed by them.

The passwords are stored along with the account name(linked to this password) and the authentication key of the user so that they can be retrieved uniquely and quickly.

Implementation:

1. *server.py* – Contains code required to set up a server and serve the queries.
2. *table.py* – Creates the database *passwords.db* and *users.db*.
3. *client.py* – Contains the code for user interface and making server requests.
4. *password.py* – API for storing, retrieving, deleting the user passwords.
5. *user.py* – API for storing the credentials of master accounts.

Technologies used:

1. Python as the programming language
2. SQLite for databases

Demonstration:

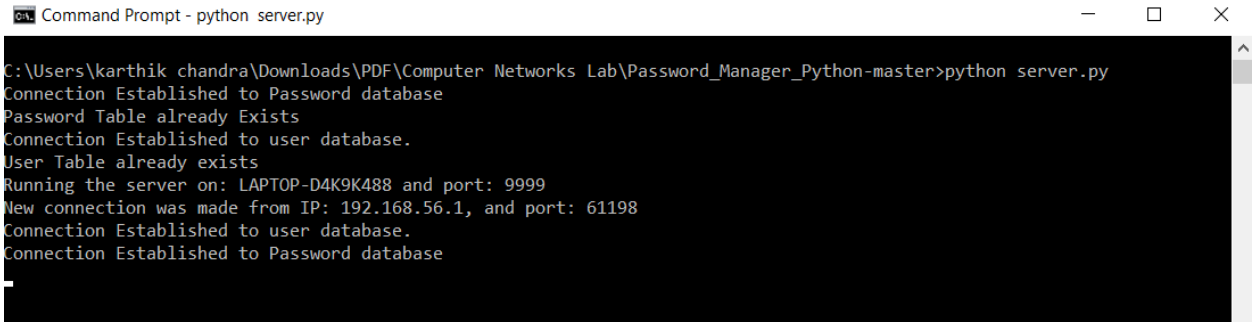


Fig-1: Server running and accepting the connections.

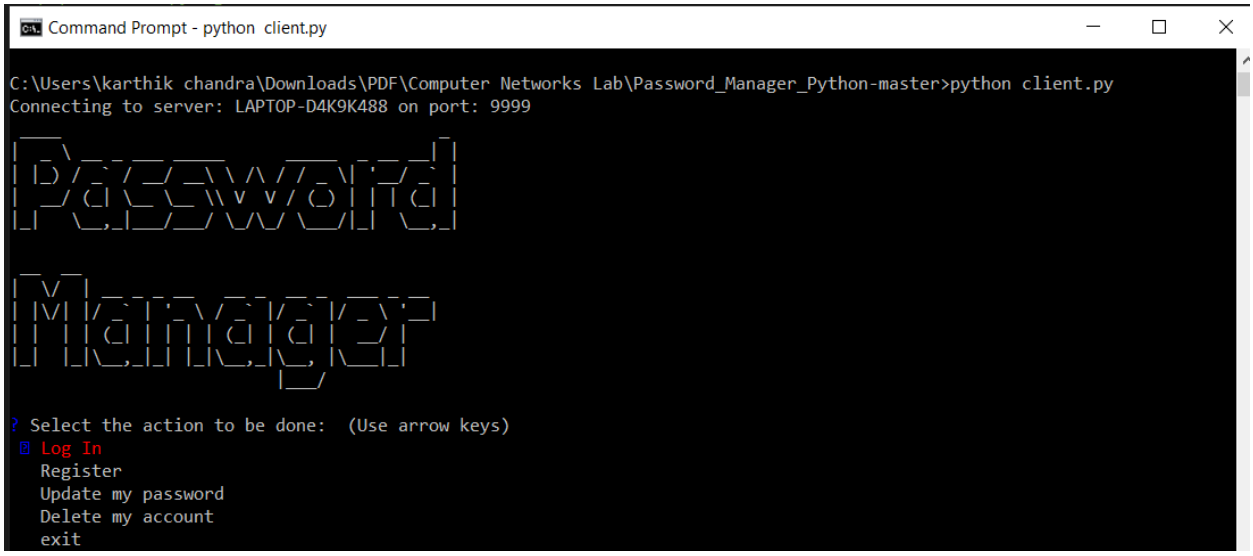


Fig-2: Client-side interface for logging into the master account, registering etc.

Fig-3: Logging into master account, interface for storing the passwords.

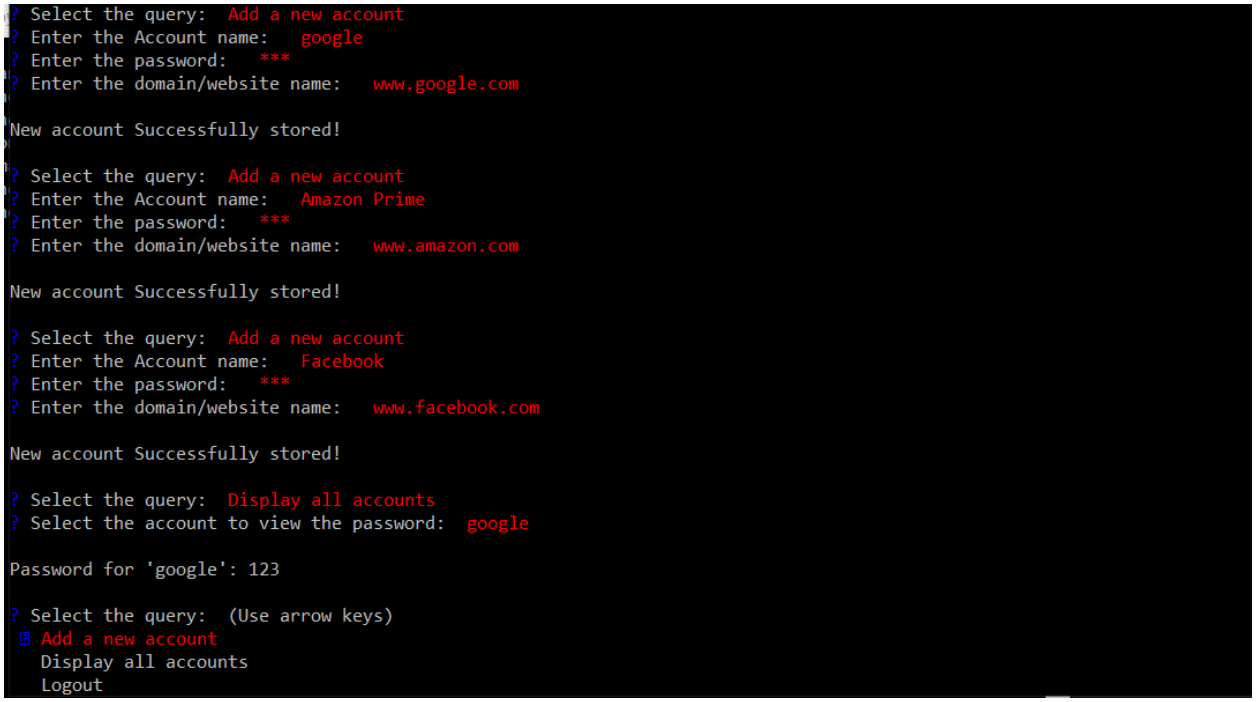


Fig-4: Storing and retrieving the passwords after logging into a master account.