Project: A Login Control GUI using Python and SQL

By Krzysztof Wawrzyn

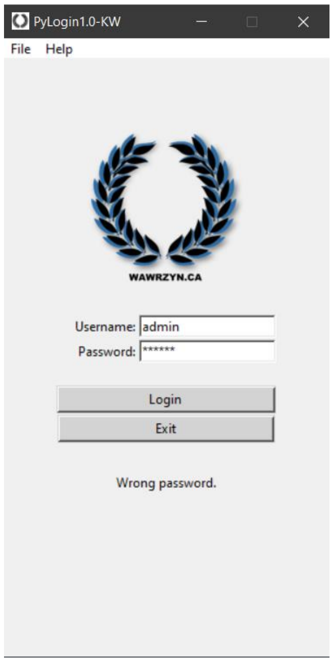


Figure 1: Login window.

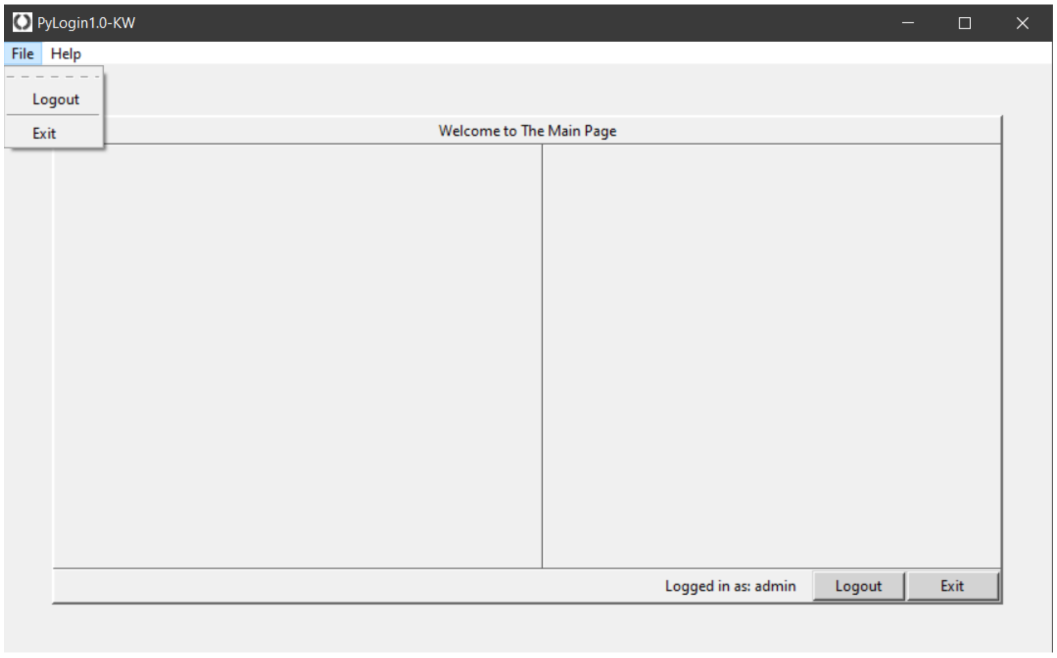
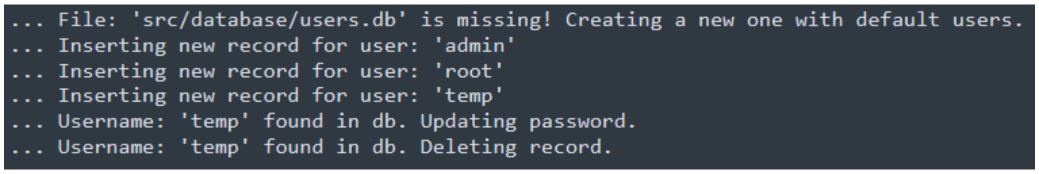


Figure 2: Main window.

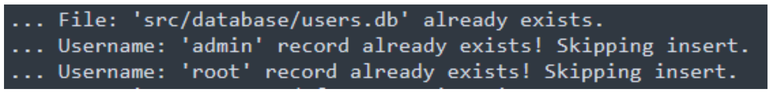
First run:



```
... File: 'src/database/users.db' is missing! Creating a new one with default users.
... Inserting new record for user: 'admin'
... Inserting new record for user: 'root'
... Inserting new record for user: 'temp'
... Username: 'temp' found in db. Updating password.
... Username: 'temp' found in db. Deleting record.
```

Figure 3: Results if database is missing.

Second run:



```
... File: 'src/database/users.db' already exists.
... Username: 'admin' record already exists! Skipping insert.
... Username: 'root' record already exists! Skipping insert.
```

Figure 4: Results if database exists.

<u>Objective:</u> To create a GUI that acts as a baseline for future Python-related projects.

<u>Technical Details:</u>

I developed this application using both Python programming and SQL together. I applied the Model-View-Controller (MVC) software design pattern concepts for the benefit of separation of concern, which results in easier code maintenance, re-use, and independent testing. This was done by applying object-oriented programming with each MVC element separated into its own class file.

For the View (i.e. frontend), I used the GUI library TKinter due to its reputation as being an easy and fast method to implement GUI's in Python. Two main windows were designed: (1) An initial login screen for the user to input the login credentials (Figure 1), and (2) a main menu window that is mostly empty but acts as a container for content of future applications (Figure 2).

For the Model (i.e. backend), I used SQLite for its capability as a lightweight database management system. I used Python with SQL to create & access the main database, create & access tables to store user login information, populate the tables with default user credentials, read user data, update user records, and delete user records. The Model also handles authentication of login credentials.

The Controller interfaces between the View and Model to help with the authentication process. When a user inputs the username and password (Figure 1), the Controller will detect when the login button is clicked and send the authentication request to the Model to check if there is a match between the credentials that were input by the user and the existing records in the database. At successful authentication, the login screen disappears, and the main menu pops up (Figure 2).

When the application is executed, it checks if the database that contains the user login credentials exists. If the database is missing, a new database gets created with default login credentials (Figure 3). If the database exists, the application will connect to the database (Figure 4). At successful connection, a simple Python-driven sequence of Create-Read-Edit-Delete (CRED) operations is applied using basic SQL queries to test the database (Figure 3 and 4).

<u>**Results/Outcomes:**</u>

The outcome successfully met the initial objectives. The GUI with buttons and entries is fully functional. The application detects if the input username/password combinations are registered in the database. If the credentials pass the authentication, the main menu is displayed. A functional navigation bar was also implemented in the main menu. All buttons and menus work properly, as intended.