A logo for a university

Description automatically generated

Higher diploma in science in computing

Lecturer: Patrizio Simeoni

Subject: Scripting

Lab Nine

Course: CW\_KRSIT\_H

Date: 19/2/2025

Student: John Paul Larkin

Student number: C00001754

## Summary

This lab exercise involved developing a Python script that demonstrates two separate functionalities.

### Star Wars Data Management and Visualisation

The script fetches character data from the Star Wars API. An SQLite database is used to store the characters information (name, gender, homeworld). The stored data is then visualised using a bar chart to represent the gender distribution of the characters.

### User Login System

A simple user account system was implemented that allows users to sign up, log in, change their password, or delete their account. User details are stored in a separate SQLite database with fields such as username, password, login count, and last login timestamp.

### Database Class

The Database class encapsulates methods to create tables, save data, retrieve data, and close the connection. This modular design helps in keeping database operations separate and organised.

**Properties**

connection - An SQLite connection object that establishes a connection to the specified database.

cursor - A cursor object used to execute SQL queries on the connected database.

**Methods**

**init**(self, db\_name="star\_wars.db") - Constructor that initialises the database connection and cursor. By default, it connects to or creates a database named "star\_wars.db".

create\_table(self, table\_name) - Creates a table in the database if it does not already exist. The table includes columns for id, name, gender, and homeworld.

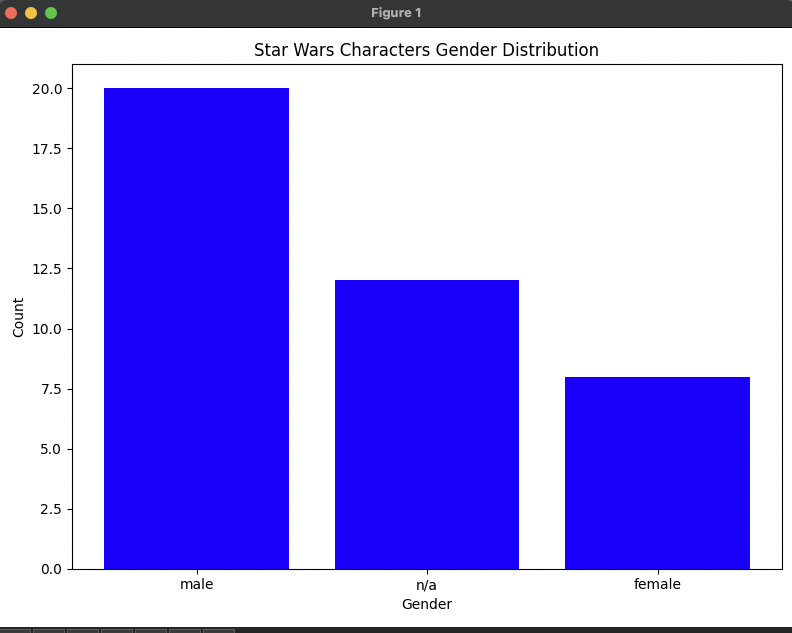
save\_data(self, table\_name, name, gender, homeworld) - Inserts a single row of data into the specified table with the provided character details.

get\_data(self, table\_name, cols) - Retrieves the specified columns of data from the given table in the database.

close(self) - Closes the database connection, ensuring that all resources are properly released.

The function fetch\_data() retrieves JSON data from the API. The script then iterates over the returned results to insert each character’s details into the database using the save\_data method.

The plot\_data function extracts gender information from the database and uses Matplotlib to create a bar chart.



### Login Class

The Login class manages user registration and authentication. It provides methods for signing up, logging in, changing passwords, and deleting accounts. The use of parameterized SQL queries ensures that the system is secure against SQL injection attacks.

**Properties**

connection - A SQLite connection object that connects to the specified database (default is "users.db").

cursor - A cursor object used to execute SQL queries on the connected users database.

**Methods**

\_\_init\_\_(self, db\_name="users.db") - The constructor initialises the connection to the users database and creates a table named users (if it doesn't already exist). This table includes columns for username, password, login\_count, and lastLogin.

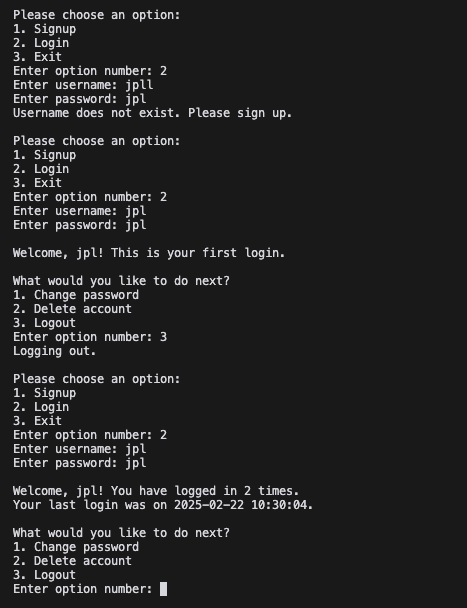
signup(self) - Allows a new user to create an account by entering a unique username and password. It checks if the username already exists before inserting the new user record into the database.

login(self) - Manages user authentication by verifying the provided username and password against stored records. Upon successful login, it updates the user's login count and last login timestamp, and then presents options for further actions, such as changing the password or deleting the account.

change\_password(self, username) - Enables a logged-in user to update their password. It prompts for a new password and then updates the corresponding record in the database.

remove\_account(self, username) - Deletes the specified user's account from the database after confirming the user's intention to delete their account.

The image below shows the login system menu. We can see that on the first login the user receives the message “Welcome johnlarkin, this is your first login.” On the subsequent logins the message changes to “You have logged in X times” along with a timestamp for the last login time.



## Conclusion

The lab demonstrated the integration of various Python libraries such as sqlite3, requests, and matplotlib to build a functional application. The project fetches and visualises data from an external API, and also implements a simple login system. Through this lab, key skills were reinforced, including database management through creating and interacting with SQLite databases, API integration by fetching and processing JSON data, data visualization using Matplotlib to create informative charts, and user authentication through a simple yet effective account management system.

The modular design, achieved by breaking the application into distinct classes like Database and Login, demonstrates the basic concepts of encapsulation and abstraction, which can be scaled up in larger projects to enhance maintainability and extensibility. Basic error handling measures, such as checking API response status and duplicate username verification, contribute to the robustness of the application.