Project Report

Taiwo Osinaike
MO1100339
BSc Information technology

https://github.com/osinaiketaiwo629-eng/CW2_M01100339_CST1510

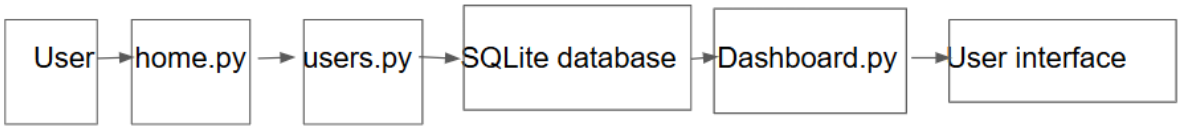
Introduction

The Multi Domain Intelligence platform is a Streamlit based web platform used to give users meaningful insight into different areas of professionalism: cybersecurity, data security and IT operations. Because in the real world organisations would typically use data intelligence to solve specific issues in regards to their domains of expertise relevant to them, the web platform seeks to acknowledge and solve each problem. For example, for cybersecurity, the dashboard responds to spikes in phishing attacks and spots workflow issues. For data science, it seeks to tame large datasets and improve the quality of data. And for t operations, the dashboard serves to see which staff or steps slow down help tickets.The user logs in or registers an account and is taken to the dashboard which provides tools and visualisations needed for their area of expertise, providing useful information. This project would be built week by week, starting with the implementation of bcrypt in regards to the hashing of passwords,building an SQLite database and CRUD operations, to the development of an interactive dashboard that provides users relevant and meaningful information to later, the inclusion of AI to enable an AI chatbot on the platform. My project aligns mainly with tier 2 but has some minimal elements of tier 3 with the ai chat page implementation.

How I Built the system

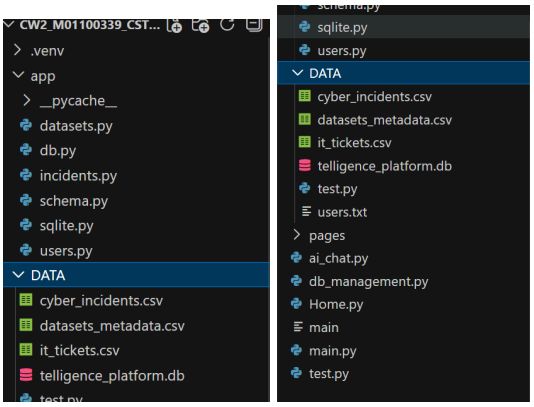
Security is a highly prioritised area of concern when creating platforms that store user data as it contributes to a highly secure authentication system. To see to this, when the user registers, the passwords provided are not stored into plain text, they are instead put into hashes which convert strings into encrypted formats that cannot be reversed or even decrypted. When compared to the password the user inputs upon logging in, the hashed passwords are compared for any similarity before verifying whether it is the correct password or not initiating a successful login print. This reflects real corporal methods of securing personal user information as hashing is a common tool used to maintain security for users who use their platform due to its overarching utility. Because of the SQL database that was built, the user-imputed information such as passwords and usernames are now stored in a secure database which includes usernames, passwords, domain specific information and datasets. There is also the implementation of CRUD operations in the project allowing the system to create, read, update and delete data, ensuring the reliability of the data provided on the platform as it is up to data for each specific domain.

As a result of these implementations the platform builds on its utility, storing passwords in hashes and comparing them for security purposes alongside storing the user inputted information alongside datasets and relevant information for each domain.



The flow chart perfectly describes the sequence of events that will take place the moment the user reaches the web platform. Upon reaching the site, the user will be asked to login or register, imminently creating interaction between the user and the home.py page. The user will enter credentials that are passed onto the users.py file where the password is hashed using bcrypyt. Once hashed the password is sent to the SQL database where it and other user information is stored alongside datasets and domain information. The hashed passwords are then compared in the login page and, if successful, the user reaches the dashboard.py. The dashboard retrieves the relevant information needed from the database using queries and processes and visualises said data into charts and diagrams that display relevant information for the user domain specificity and these are made in Streamlit.

The dashboard is the main component of the web platform as it, right after the user logs in, they have full navigational access to other aspects on the page such as the domain specific data or the ai chatbot. This is done through the use of a sidebar bar in which warrants interactivity with the user through the click of a project allowing them to navigate from one page to another. Regarding the domain-oriented intent of the users, the user loads the page through the sidebar in which Streamlit immediately provides the tools and datasets relevant to their role. The dashboard retrieves data from the SQL database and displays them in charts and diagrams that cater to the user's needs. Overarchingly the user interface is simple, convenient and intuitive, allowing for ease of use.



The platform was organised in such a way that made usability convenient and reliable. The structure is modular and distinguished and has folders for each major python file that serve differing functions to make it simpler. The users.py and schema.py are in the app folder, the pages folder includes ai chat.py and the home.py for the website and the DATA folder includes CSV files such as the SQL database and the telligence_platform.db for analysis and visualisation on the website. Moreover these folders contain files that serve separate functions such as the home.py handling logging in and the users.py handling the implementation of hashes into user credentials. While the structure does not include classes or any design patterns, this structure ensures that it is clear each file has a specific responsibility that distinguishes code from one another allowing for ease of use.

Reflection

Throughout this project I encountered many hurdles, whether it was the growing complexity in the code itself or its implementation in a neat or digestible formation for peer use. My biggest technical issue by far was my terminal and its inability to process basic commands. I in turn switched to command prompt which worked effortlessly and cohesively with me. Another issue I faced was implementation of users into my SQL database. Because of the major shift in coding language used for this project, it was difficult at first but I managed to implement it in the end ensuring that the user credentials were not just stored in a text file but were actually stored into a secure database. I also had issues with the overarching structure of my code. At first it was incredibly simple and neat but in time manifested into a more unorganised structure in which I improved upon through folder organisation such as DATA, app and pages, but it made it at first difficult to find relevant files and hold relevant code this had also led to poor mistakes, typos that made it hard to run certain code on the webpage in which inadvertently turned small issues into larger ones. However I have improved upon all of these and now the code functions as intended.

In conclusion, I have successfully created a multi-domain intelligence platform that displays relevant data to users within specific relevant domains, hashes passwords ensuring a secure authentication system and an ai chatbot that amplifies user interactivity. To improve, would most definitely seek to add more datasets, and implement the full OOP structuring. Overall I believe my attempt at building a multi-intelligence data platform was successful and has highlighted my capability at building a secure and functional data platform.

References:

Streamlit Documentation. *Streamlit — The fastest way to build data apps*. Available at: https://docs.streamlit.io
SQLite Documentation. *SQLite: Lightweight SQL Database Engine*. Available at: https://www.sqlite.org/docs.html
bcrypt Documentation. *bcrypt Password Hashing for Python*. Available at: https://pypi.org/project/bcrypt
Python Software Foundation. *Python 3 Documentation*. Available at: https://docs.python.org
Pandas Documentation. *Pandas: Python Data Analysis Library*. Available at: https://pandas.pydata.org/docs

Plotly Documentation. *Plotly Python Graphing Library*. Available at: https://plotly.com/python