Rory Folster

[Email address]

Milestone 3 - Journal

Final Milestone

Contents

[Milestone #3 2](#_Toc119592545)

[Log In Functionality 2](#_Toc119592546)

[Live-chat system 3](#_Toc119592547)

[Remote Upload 4](#_Toc119592548)

[Project Conclusion 4](#_Toc119592549)

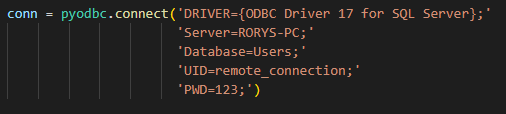
[References 6](#_Toc119592550)

# Milestone #3

For milestone 3 I have implemented log in functionality, a live-chat system and remote storage of uploaded files.

## Log In Functionality

I implemented a log in system, roughly around Milestone 2, however, this log in system used SQLite3. SQLite3 uses a database file that you can store within the directory of the application to store data and execute SQL queries to the file.

Usually, for an application such as the one I have created, this would be fine, however SQLite3 is typically non remote, and doesn’t allow for multiple connections at once, and it clearly states that the log in system needs to be remote. So, I ended up setting up an SQL Server instance on my PC at home and used ‘pyodbc’ to connect and interact with the instance. Pyodbc is an open-source Python module that is used to connect to ODBC databases. This module is very simple to use, as it only requires 5-6 lines of ‘credential’ code to connect to the database.

Text

Description automatically generatedIt is a good idea to place the connection in a variable so that you can then execute database commands in different ways, in this project, for example, I have different functions for logging in, registering and to update my ‘is\_online’ column. An example of using this connection variable is placed below.

Setting up the SQL Server remote connection was fairly easy as I have completed this task in another class (DAT701).

When a user logs into the application, a query is sent to the database to find the username & password in the Users table. If there is a match, it returns true and sets the user’s ‘is\_online’ column to 1, when a user closes the application, the ‘is\_online’ column returns to 0. The reason why I have implemented this is because I want to add functionality to the live-chat system that allows a user to see who is online. This is done by executing a query to the database to find where the ‘is\_online’ column contains 1.

The best practice for storing an important string or variable, such as the connection string above, to place the string in an environment file (.env), however, I did not find it necessary to implement this in this project.

*Notes:*

The database connection is not completely remote, the SQL Server instance needs to be on the same network as the application. I tried my hardest to open my SQL Server to remote connections, I even created a post of the Microsoft Forum (<https://learn.microsoft.com/en-us/answers/questions/1091436/sql-server-remote-connection-not-working.html>). After spending hours trying to get the database system completely remote to no avail, I have given up.

Proof of database connection working: https://youtu.be/FGPBgp3qHck

## Live-chat system

The live-chat system was created using threating and socket. Threading and socket are built in Python modules. Socket allows a client to connect to a server, in this case, it allows a client to connect to my messaging server. The client is the message sender and message receiver, the server works with the data sent by the client and depending on how it was created does something with the data. The server in this application receives and sends messages, to the client GUI.

Graphical user interface, text, application, Word

Description automatically generatedThe client portion of the live-chat system works in conjunction with the server. The server sends and receives messages, and the passes them into the GUI to be displayed.  
I implemented an SQL query into the GUI that allows a user to type ‘/users’ into the GUI and be returned the currently online users.

*Notes:*

Text

Description automatically generatedBy default, the server needs to be started separately from the application. To start the server, simply type ‘py server.py’ into the terminal that is in the application directory.

I understand an ip address is not a great way to differentiate the users, I would have liked to pass the username into the server and then used that to display who is sending the message. However, since the server needs to be started prior to the application, there is no way to pass the username to the server. The way I pass the current username across the application is by passing the username to an array when the user logs in.

I would have liked to implement a system that askes the user what they want their username to be, prior to connecting to the server, however, I ran out of time.

## Remote Upload

When a user uploads a file to the application, the file is deconstructed using Pandas, and plotted to a Matplotlib plot. If there are errors, the file will not be uploaded to the application.

If the file can be plotted, a function is called, that passes the deconstructed information to Google Sheets and stores it.

This requires a special JSON file, containing information about the credentials, to verify the user.

On Google Sheets, you need to set up an API account, and enable Google Sheets and Google Drive API, you then need to register a user into the API account. You then create an empty Google Sheets file and share access to the API account. This limits who can send information to the Sheets file.

# Project Conclusion

Working on this project was fun. I learnt a lot more about data analysis and how to achieve different goals within it. I expanded my knowledge of very popular Python frameworks such as pandas and Matplotlib. Learning more about Python made me very interested in Python as it can be very simple to complete tasks, compared to other languages such as JavaScript. For example, looping over an object and doing something with each element in Python is a lot easier than JavaScript’s for loop.

I am a bit disappointed with the structure of my application, as I would have liked to use classes and expand my knowledge with them. Instead, each DES is held in a function that is called, making it a little messy and not to an industry standard.  
  
I am also disappointed with my remote database connection as I believe I was very close to completing this task. Whether the issue resides in my home router, or a firewall somewhere, I am not sure.

As I can not seem to find any graduate jobs, or work experience, I am thinking about re-creating this application in a better way over the holidays. Which will hopefully bring my data analysis knowledge close to, or to an industry standard.

# References

*pyodbc*. (2022, November 16). PyPI. https://pypi.org/project/pyodbc/