SDV602 – Assessment 2

## Data Explorer Screen Application

## Alex Herbert

## 6/10/2024

### Project Brief and Description

The purpose of this project is to design and develop an application that can be used by data analysts to view graphs displaying visual representations of data, as well as the implementation of a chat system that can be used for communications between the data analysts. The application includes three separate Data Explorer Screens that are used to display the graphs in a Graphical User Interface (GUI), these screens are navigable and contain a placeholder for “Chat Windows”. The graphs displayed will have placeholder/mock data that is accessed locally to provide an example of the usability of the application.

While this project is a prototype and does not reflect the finished version of this application, the architectural design will become the foundation of the final product and has been designed accordingly. This application is intended to be used in conjunction with an http-based API, connecting to a remote data service, allowing the chat functionality to be used outside of an organization’s physical location or within their internal network.

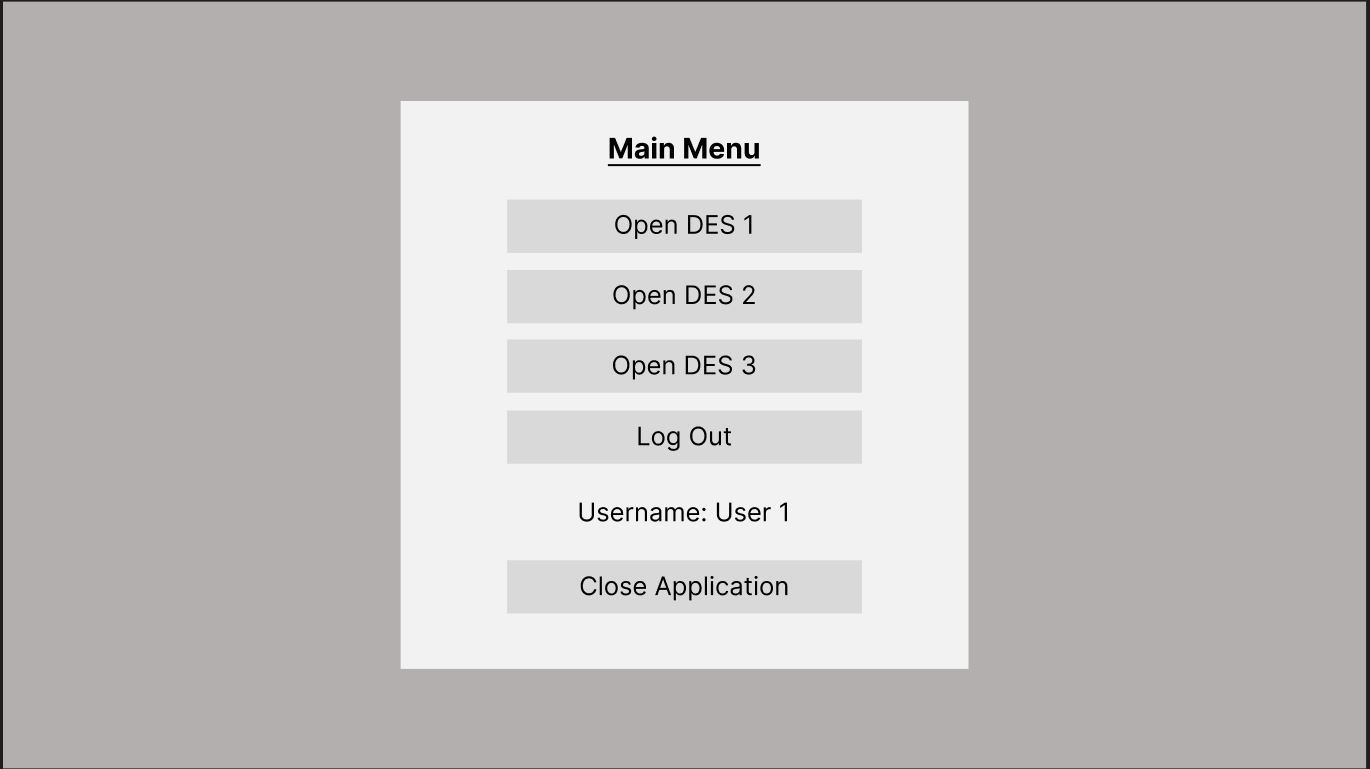
### Application Story Board and Wireframe Designs

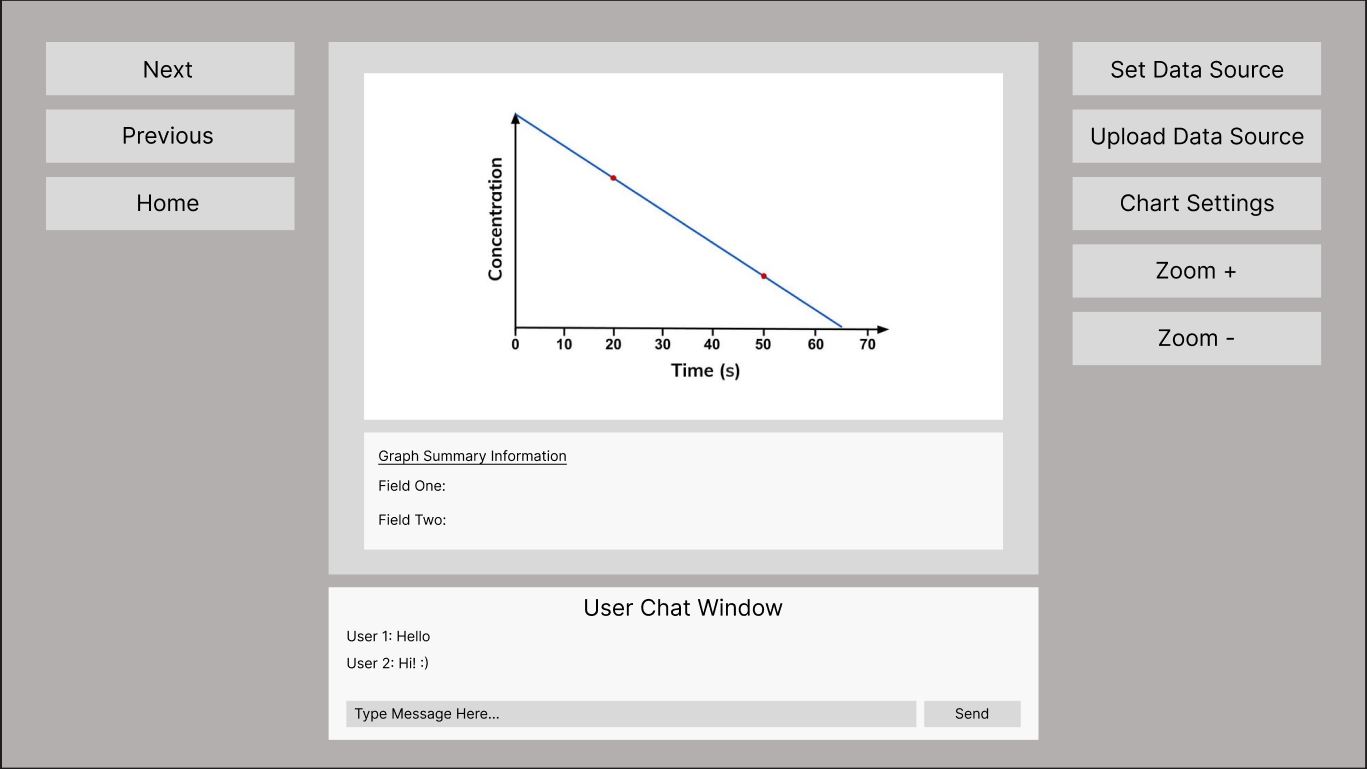
To correctly determine the functional requirements for the application and types of user interactions with the software’s GUI, the following story board and wireframe designs provide a visual representation and description of the applications use cases and functionality.

### Functional Requirements

* Application must have a Main Menu and three DES windows with different graphs.
* User must be able to cycle through different screens using GUI navigation.
* User must be able to upload and select data sources to the application to be displayed in graphs.
* The application GUI must include a Graph Summary Information section.
* The application GUI must include a user Chat window, allowing users to communicate with each other using the chat functionality.
* The application must be able to be connected to an http-based API.
* Graphs in the application must be configurable and include a Zoom in and out functionality.

### Wireframe Designs



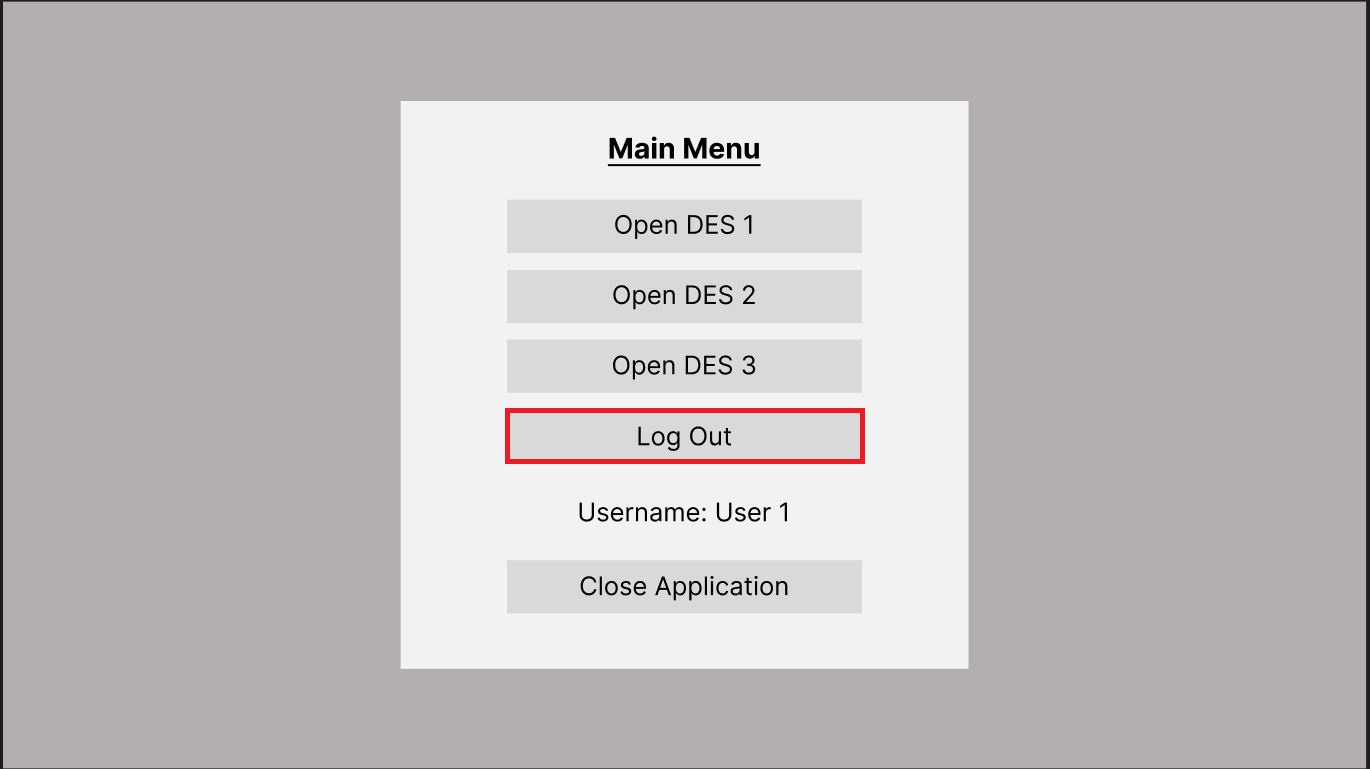


### User Story One – User selects DES 1 button from Main Menu



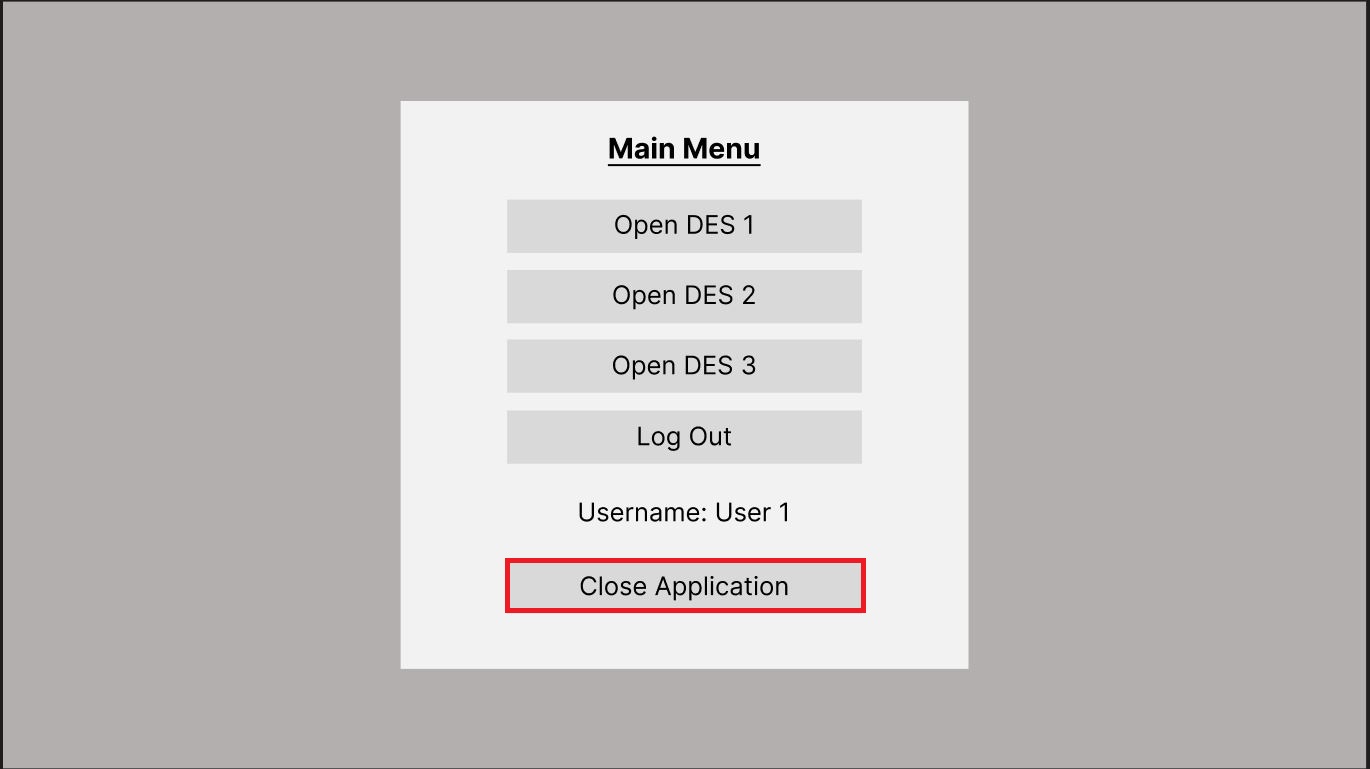
The user opens the application and is presented with the main menu, from here the user then selects the “Open DES 1” button which opens/loads the DES 1.

### User Story Two – User logs out of their account using “Log Out” Button



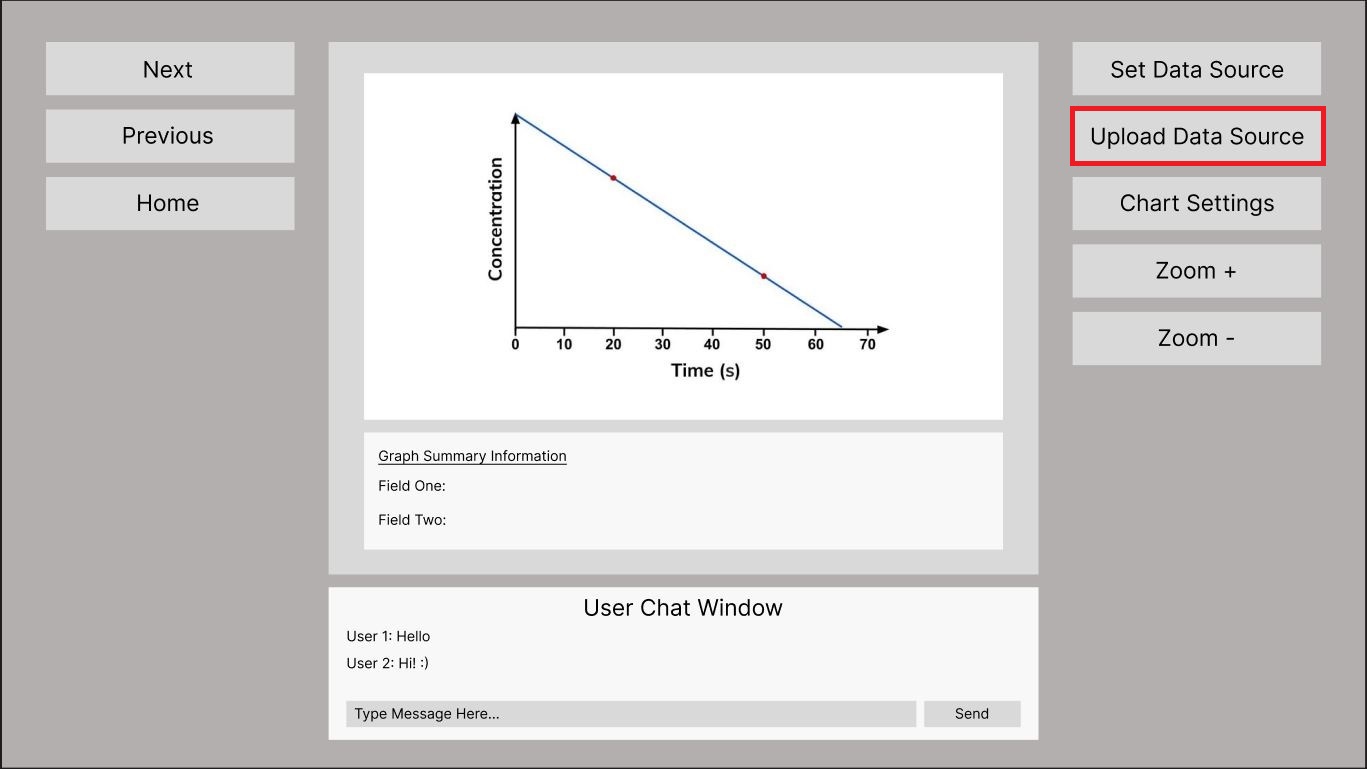
When presented with the main menu, the user logs out of their account by selecting the “Log Out” button, for this prototype this functionality is a placeholder.

### User Story Three – User Closes the Application by selecting the “Close Application” button



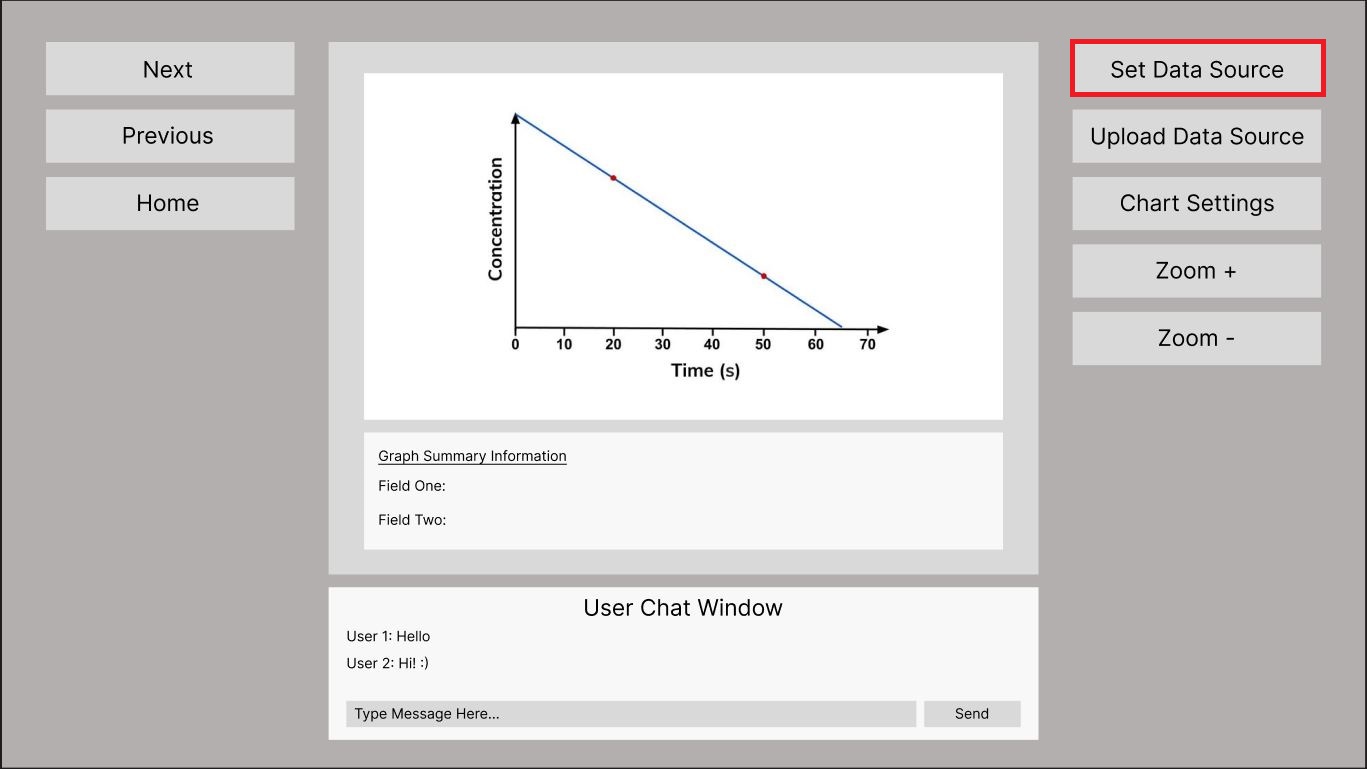
To close the application, the user selects the “Close Application” button, this closes any of the active screens and terminates the application.

### User Story Four – The user uploads a new data source for use by the graph on active screen



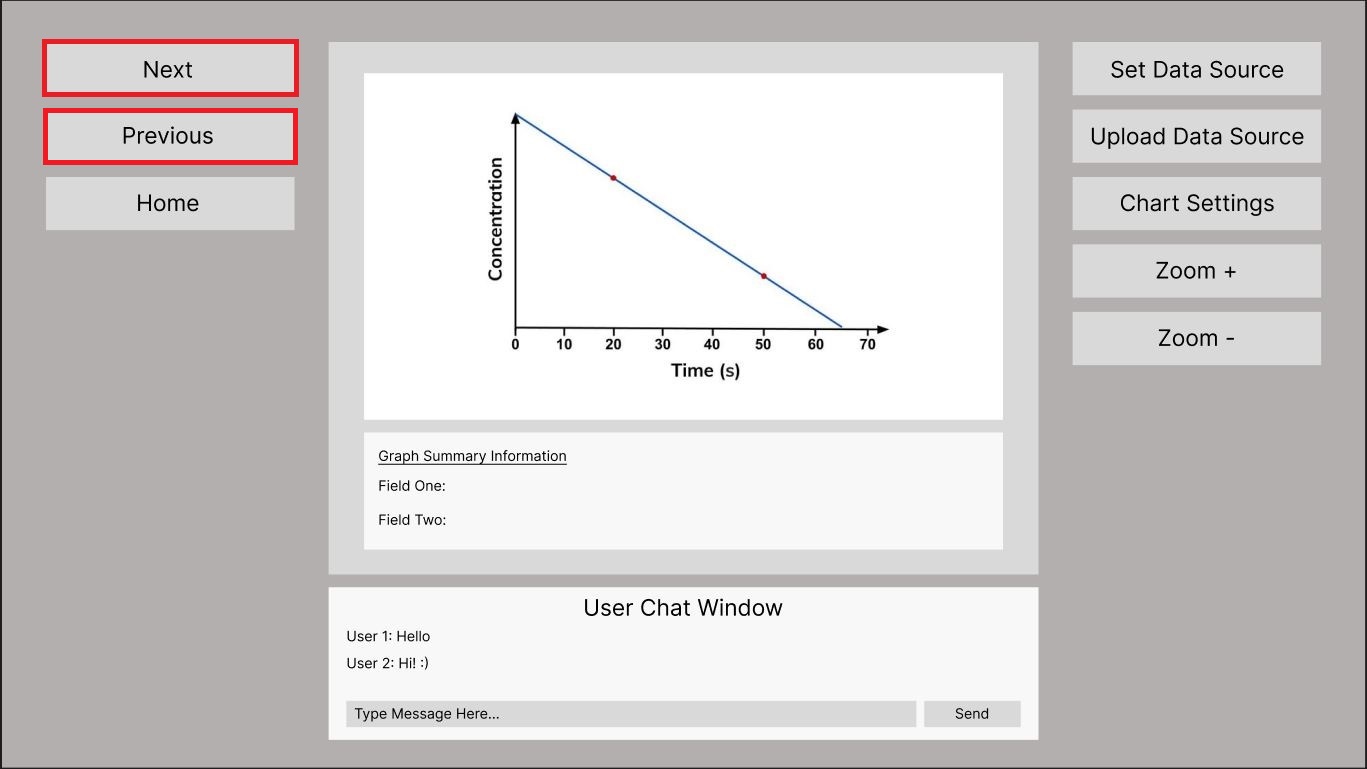
The user selects the “Upload Data Source” button, which then presents a window allowing them to select and upload an appropriate file that can accessed by the graph.

### User Story Five – User set’s the data source to be accessed and displayed by the graph on active screen



Once the user has uploaded a data source to be accessed by the graph on the active DES, the user then selects the “Set Data Source” and selects the uploaded file, the data from this file is then processed and loaded into the graph on the active screen.

### User Story Six – User changes the active screen using the “Next” or “Previous”



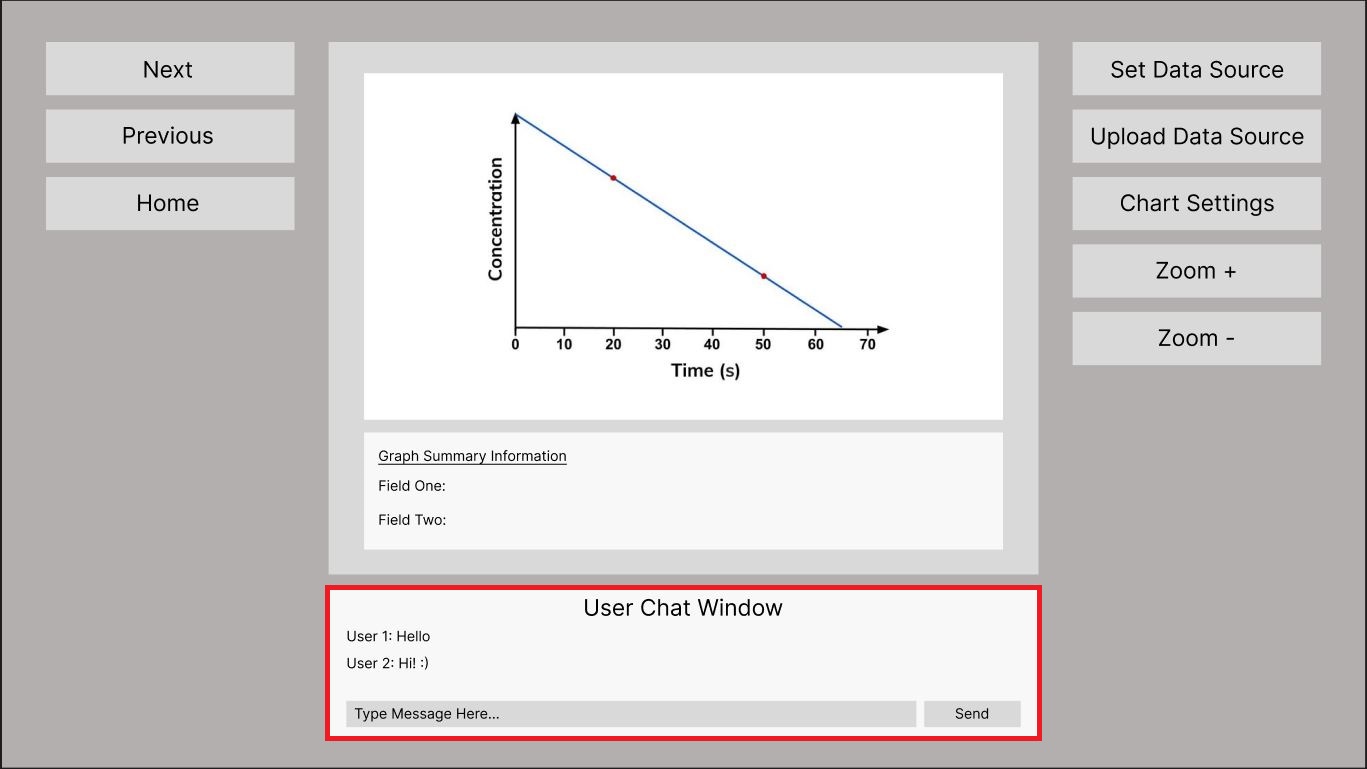
The user cycles through the three Data Explorer Screens by selecting either the “Next” or “Previous” buttons in the application GUI, this loads and displays the DES according to the screen that is active.

### User Story Seven – User reviews the DES graph displayed and graph summary information panel

### 

The user uploads a data set and sets it using the appropriate controls in the application GUI, the data is the visualised and displayed in the graph panel on the active DES to be reviewed by the user.

### User Story Eight – User types a message into the User Chat Window and sends the message to other users logged into the application

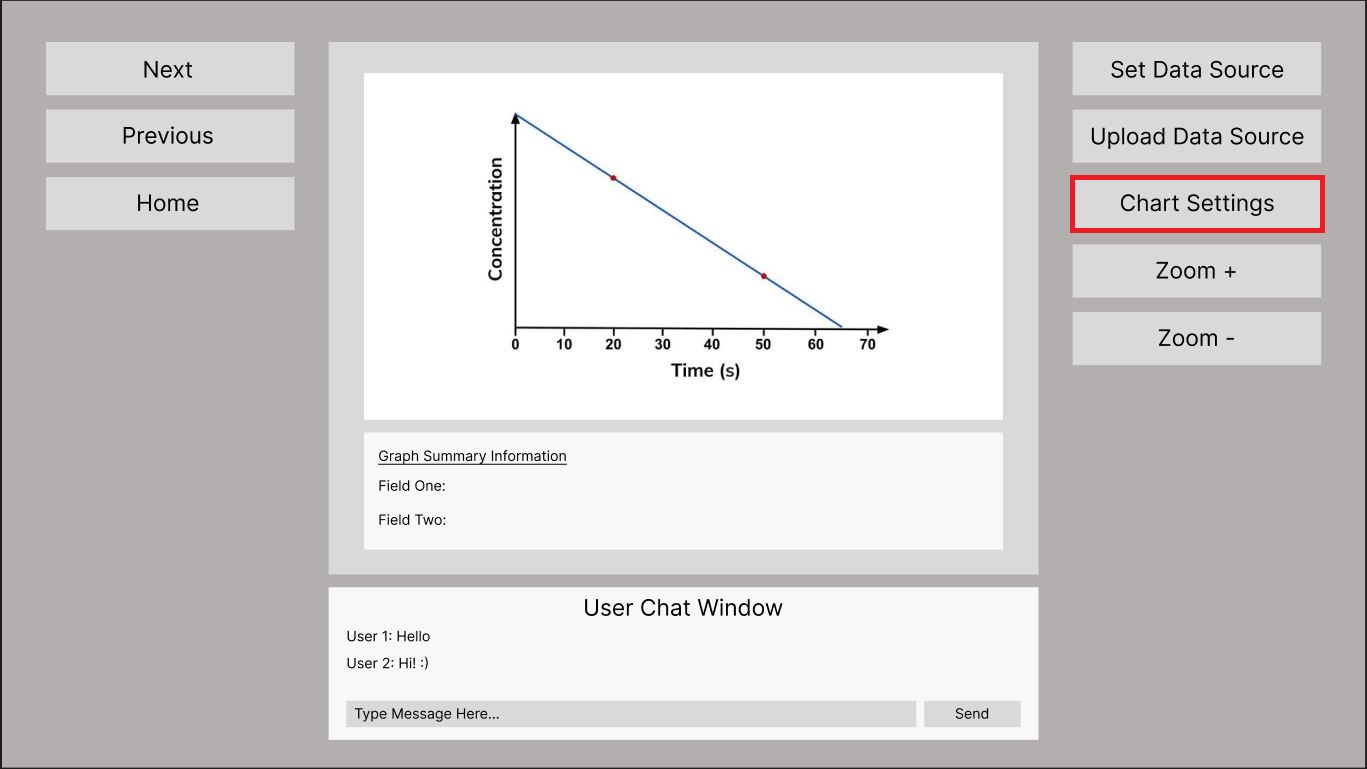


The user is logged into the system using their credentials, whilst interacting with the active Data Explorer Screen the user types a message into the User Chat Window and sends a message. This message is received and displayed to other users that are currently logged into the system.

### User Story Nine – User Zooms in or out of the graph displayed in the DES by using the “Zoom +” or “Zoom -“ buttons in the GUI

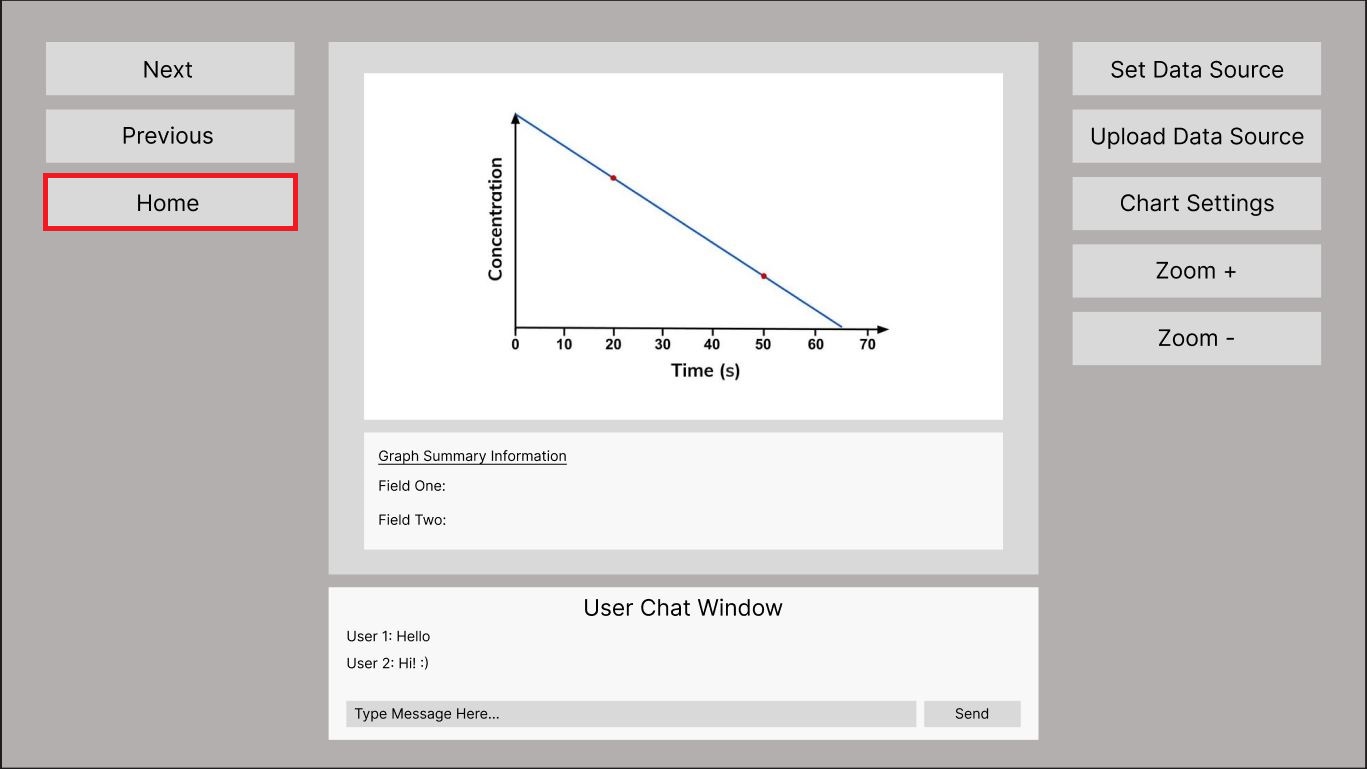
The user wants to increase and decrease the viewing size of the graph displayed in the active Data Explorer Screen, they select either the “Zoom +” or Zoom –“ button which increases or decreases the viewing size of the graph that is displayed.

### User Story Ten – The user changes the settings of the graph by selecting the “Chart Settings” button to change the ranges and values displayed.



The user selects the “Chart Settings” button on the GUI, this then presents the chart settings menu which allows the user to change the values and ranges displayed on the graph.

### User Story Eleven – The user selects the “Home” button on the active DES interface to return to the main menu

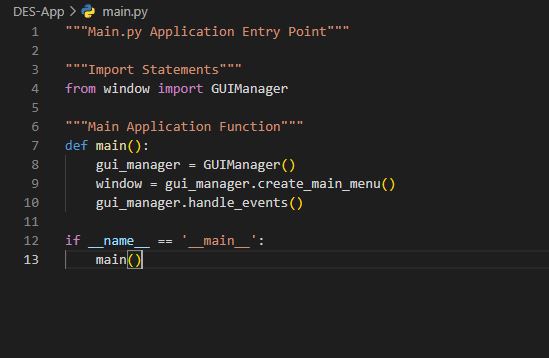


The user selects the “Home” button on the active DES interface they are working on to return to the main menu, this closes all active screens and displays the main menu.

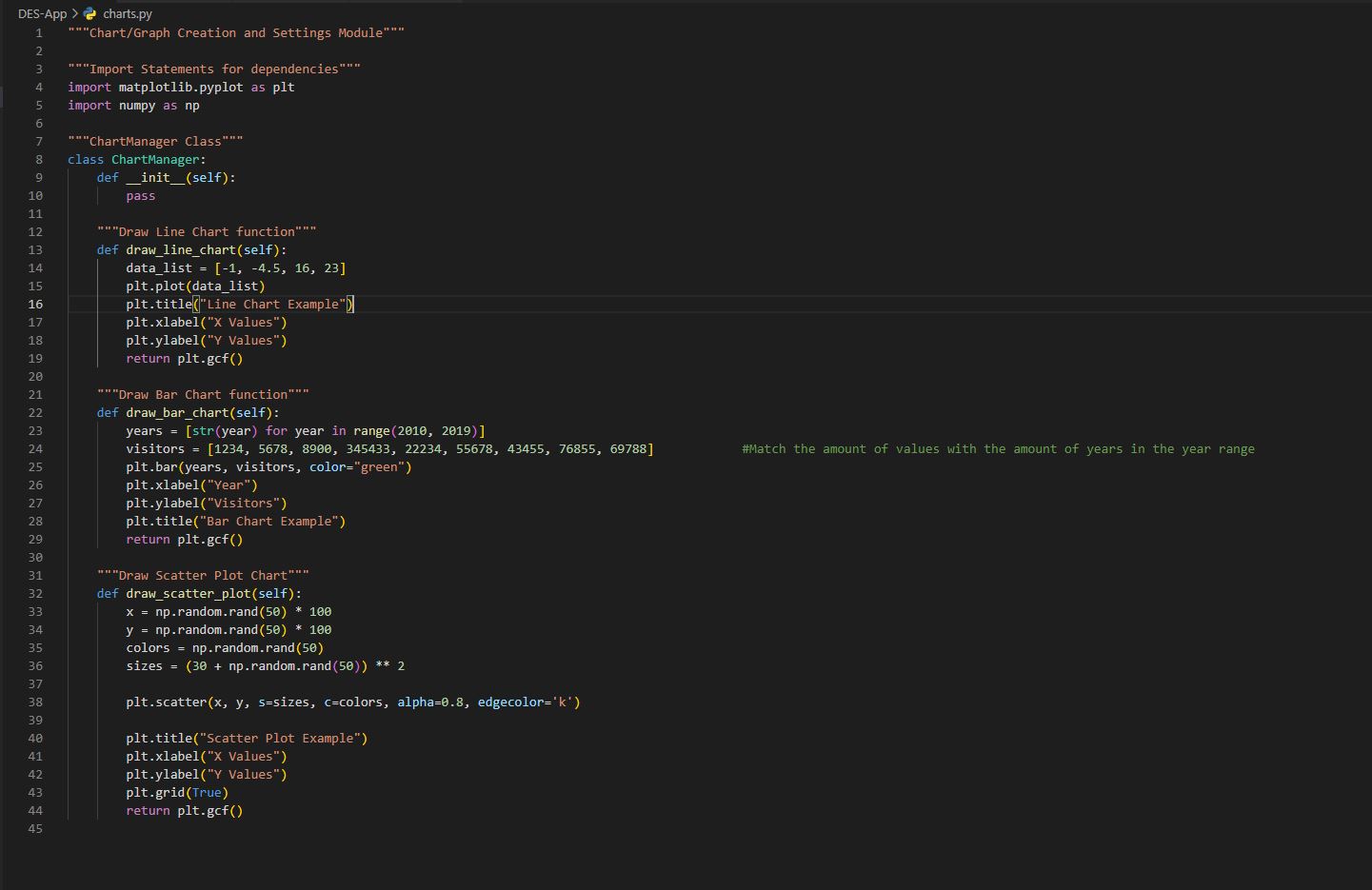
## Test Scripts and Code Implementation

The following scripts and code examples are the initial prototype implementation of the project application, some buttons and functionalities are currently place holders and do not function as they would in the final product. This prototype is to show an example of how this application can be used by data analysts to review an organizations data with graphs and visualizations of the data in a readable and user-friendly interface.

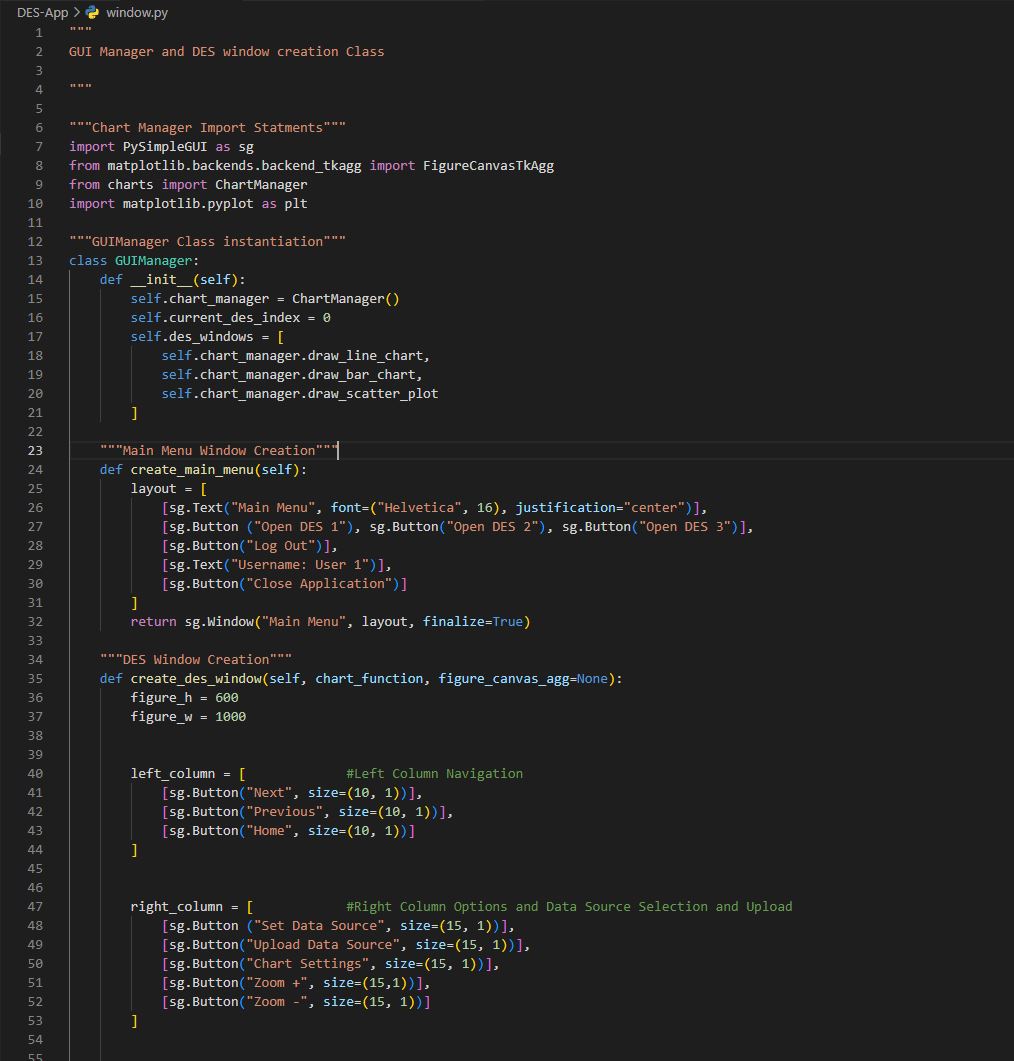
Main.py



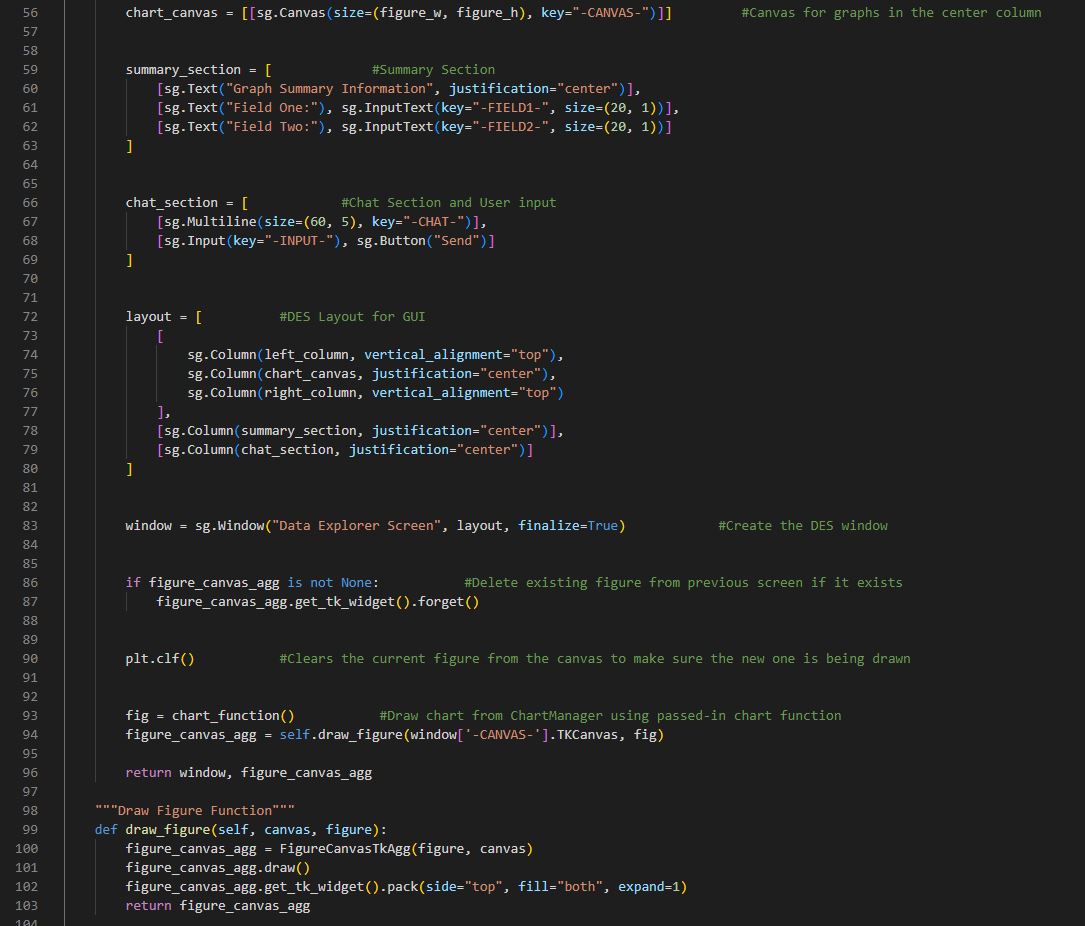
Charts.py



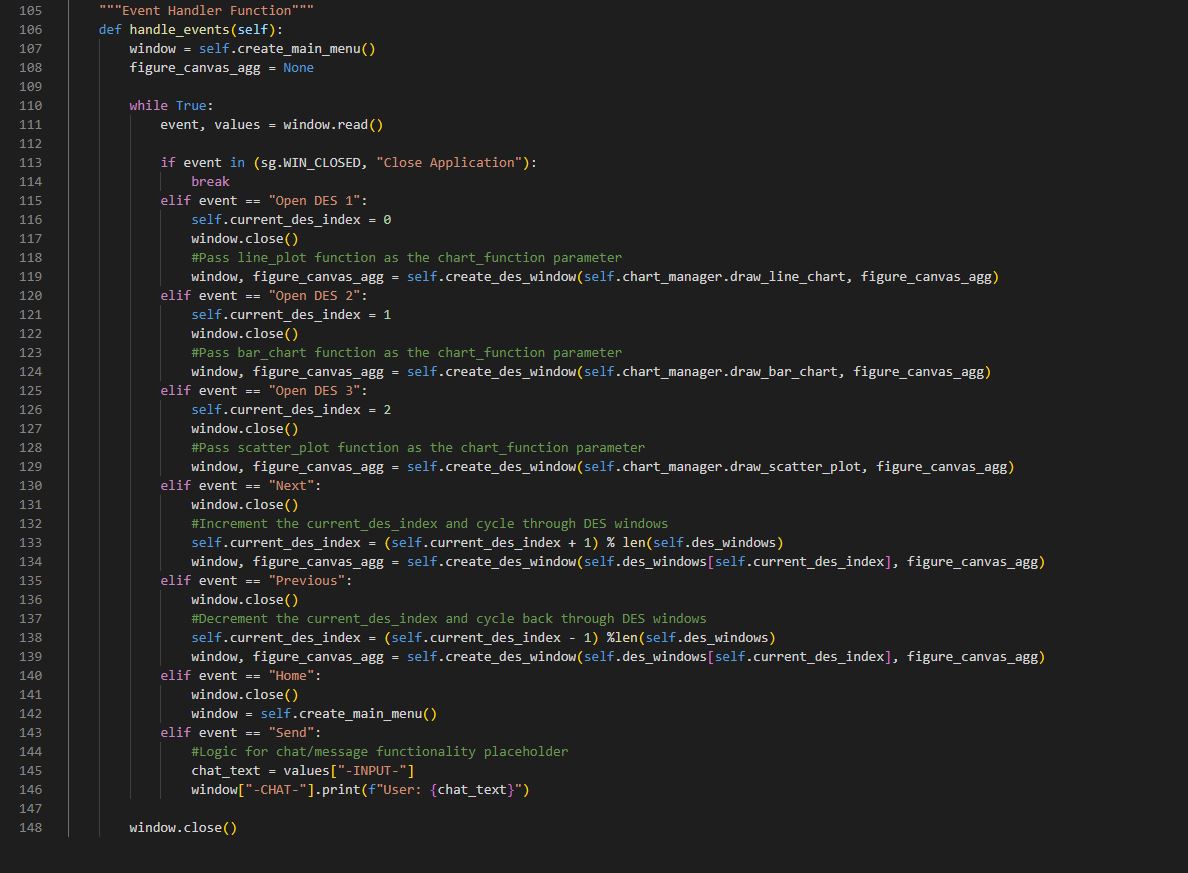
Windows.py Part 1



Windows.py Part 2

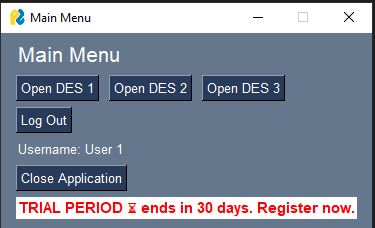


Windows.py Part 3



## Application Prototype Screens

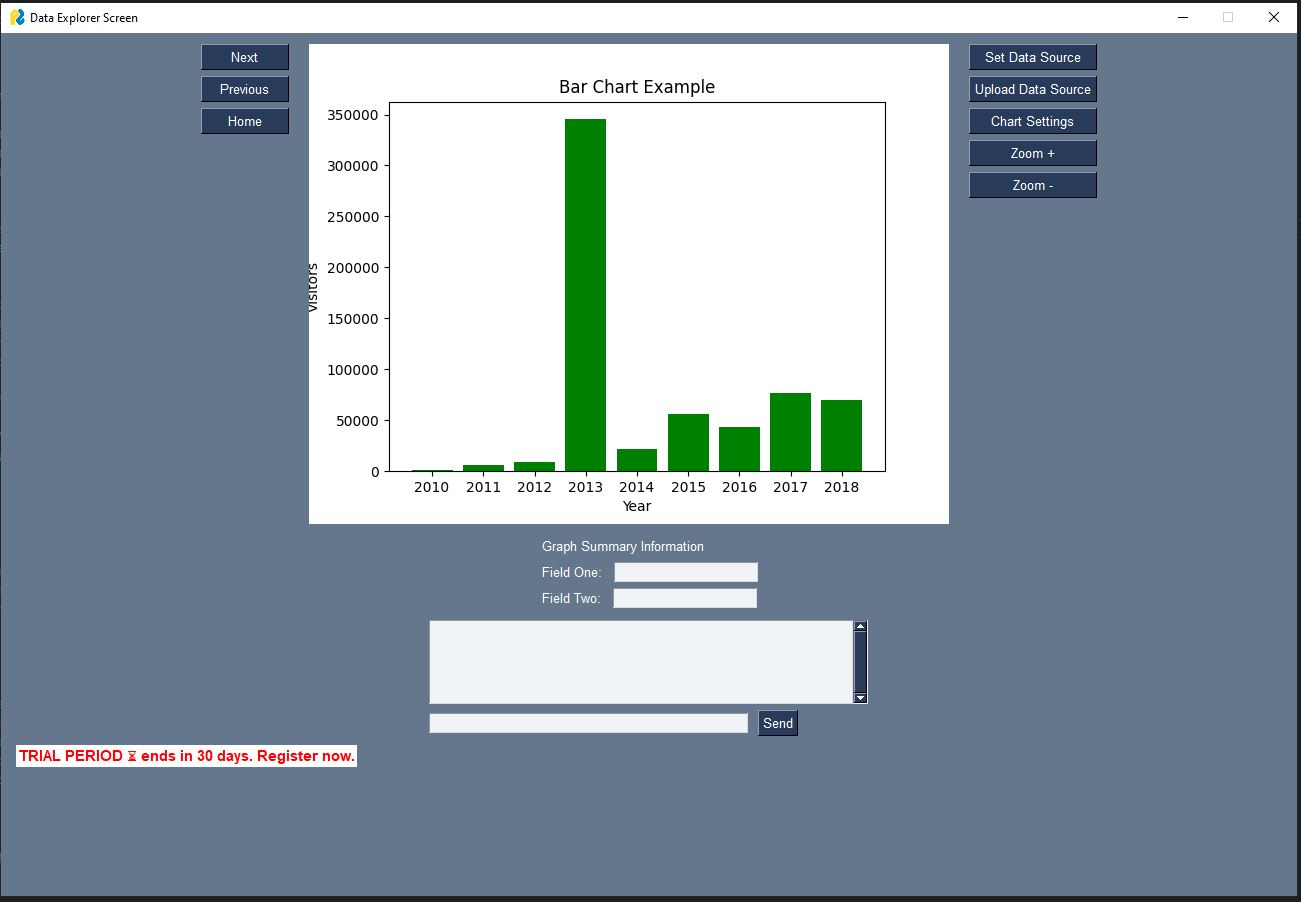
Main Menu



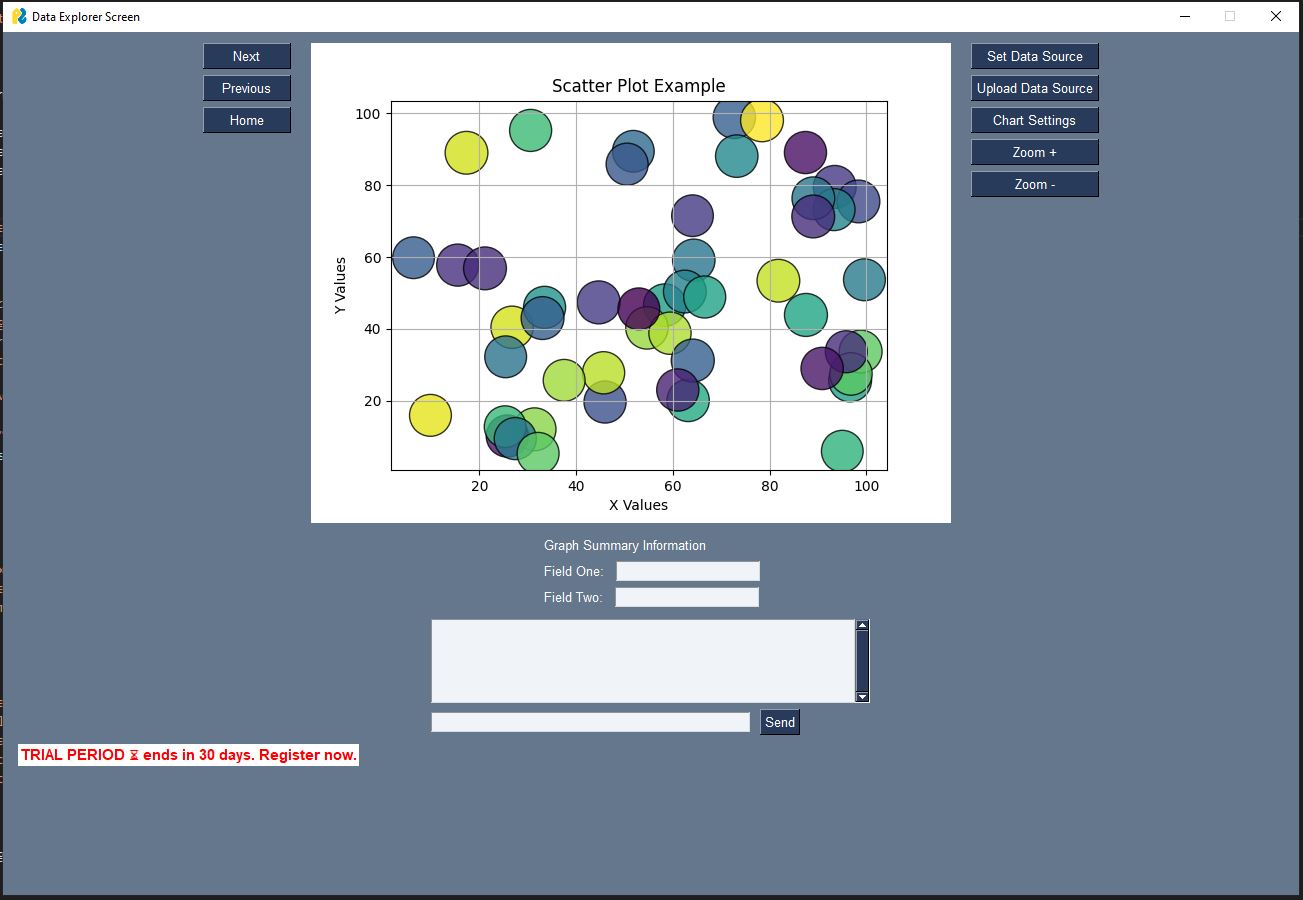
DES Screen 1



DES Screen 2



DES Screen 3



## Final Reflection on the Python Programming Language

#### Heritage and Philosophy of Python

The Python programming language was first conceived in the late 1980’s by Guido van Rossum in the Netherlands, initially it was intended to be a successor to the ABC programming language. The development and implementation of Python began in 1989, Van Rossum took on the responsibility for the project as the lead developer until 2018 and relinquished his role as the “Benevolent dictator for life” that was given to him by the Python community. Python has gone through many versions over the years, adding features such as list comprehensions, cycle-detecting garbage collection, reference counting and Unicode support.   
Python is a multi-paradigm programming language which lends itself to Object Oriented Programming (OOP) and structured programming, providing full support for the different programming approaches, amongst many others. The core philosophy has been summarized as:

* Beautiful is better than ugly.
* Explicit is better than implicit.
* Simple is better than complex.
* Complex is better than complicated.
* Readability counts.

Although these principles have been violated over the years and criticised by the Python community for adding unnecessary language bloat when adding new features and functionality, Python remains as being a simple and readable programming language with a wide range of use cases.

#### Platforms for Developing and Running Python Software

Due to Python’s extensive library and ecosystem of tools, platforms and libraires, it makes Python suitable for developing and running software applications across various use cases and domains. Some of the development platforms for python include:

* **PyCharm** – IDE created by Jetbrains, offering multiple features including intelligent code completion, debugging, and version control functionality.
* **Visual Studio Code (VS Code)** – Created by Microsoft and used in a variety of development environments, VS Code comes with highly customizable features for Python extensions including linting, debugging, and AI powered InteliSense.
* **IDLE** – This is Pythons built in IDE platform, this has been designed for beginners to learn how to code with Python using a simple and user-friendly interface.
* **Text Editors** – Some lightweight text editors like Sublime, Atom and Vim are used for Python scripting.

Additionally, Python comes with a variety of libraries and frameworks, making it ideal for different projects on multiple platforms, some of these include:

* **Web Development** – Frameworks such as Django and Flask are used in the development of web applications.
* **Data Science and Machine Learning** – Python libraries such as pandas, NumPy, and Matplotlib are used to create applications that can be used for advanced data analysis and also AI development projects.
* **Game Development** – Pygame is a library for Python that is used to support game development and other multi-media software applications.

#### Characteristics, Strengths, and Weaknesses

As mentioned earlier in the reflection, Pythons philosophy is based around some core principles and standards to make sure that the programming language is useable in a wide variety of use cases and projects. The following outlines Pythons core characteristics, strengths, and weaknesses.

###### Characteristics

* **Readability** – Pythons philosophy priorities readability and providing a simple syntax so that developers can focus on solving issues and problems without the added complexity of a difficult to read and complicated syntax.
* **Interpreted Language** – Python runs and executed code in a line-by-line approach, which allows for a smoother debugging and testing process during development. Once errors are identified, it makes the isolating the issue a lot easier than other programming languages.
* **Dynamic Declaration of Variables** – Variables in Python don’t require explicit declaration, allowing for a more concise code structure and reducing the number of errors due to variables being declared incorrectly or outside of the scope of a function etc.
* **Multi-Paradigm Support** – Pythons ability to accommodate multiple programming paradigms such as OOP and structured programming paradigms makes Python a very flexible and diverse programming language.

###### Strengths

* **Versatility** – Due to Python having an extensive library of both standard and community packages available for almost any environment or domain, Python is one of the most versatile programming languages in use today.
* **Easy to Learn** – Pythons simple syntax and structure using natural language naming conventions and methods make Python an ideal programming language for beginners as well as experienced developers due to its extensive usability and features.
* **Extensive Support** – The Python community is vast and very active, keeping documentation and tutorials up to date, as well as troubleshooting forums for helping with issues that developers come across and need help with a solution.
* **Cross-Platform** – Applications built using Python can run on various operating systems without the need for huge changes to the code or project.
* **Integration** – Python can be integrated into projects using other programming languages such as C, C++, and Java, increasing its usability and versatility as a diverse and user-friendly programming language.

###### Weaknesses

* Performance Limitations – Due to Python being an interpreted, line-by-line programming language it can run slower than compiled languages like C++, specifically when dealing with intense, computation-heavy tasks. This can hinder the performance of an application and increase the need for optimization improvements.
* Memory Consumption – Python’s ability to allow dynamic typing and garbage collection mechanisms can lead to the program using a higher amount of memory usage, specifically when compared to other statically typed programming languages.
* Limited Development for Mobile – Python is not typically used in mobile application development as it does not provide much support for mobile, as well as performance and memory constraints that are not ideal for mobile devices.

#### Conclusion

The Python development and community adhere to a philosophy and set of principles that emphasise readability, simplicity, and providing extensive tools, frameworks and libraries for use in a variety of projects and applications. Due to Pythons versatility and usability, it has become a widely used programming language that is ideal for both beginners and experienced developers. Whilst Python does have some draw backs and weaknesses, its strengths outweigh the issues with performance and memory usage. Python is used in education, the Software Development industry, and AI development, and due to its ability to be used in cross-platform software applications it can be used in a number of ways to develop powerful solutions to real-world problems.

## References and Citations

Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Alyssa Coghlan <ncoghlan at gmail.com>. August 2013. PEP 8 – Style Guide for Python Code.

<https://peps.python.org/pep-0008/#introduction>

PySimpleGUI Team. 2024. PySimpleGUI Documentation.

<https://docs.pysimplegui.com/en/latest/>

Matplotlib Team. 2024. Matplotlib User Guide.

<https://matplotlib.org/stable/users/index>

Wikipedia. 2024. Python (programming language)

<https://en.wikipedia.org/wiki/Python_(programming_language)>