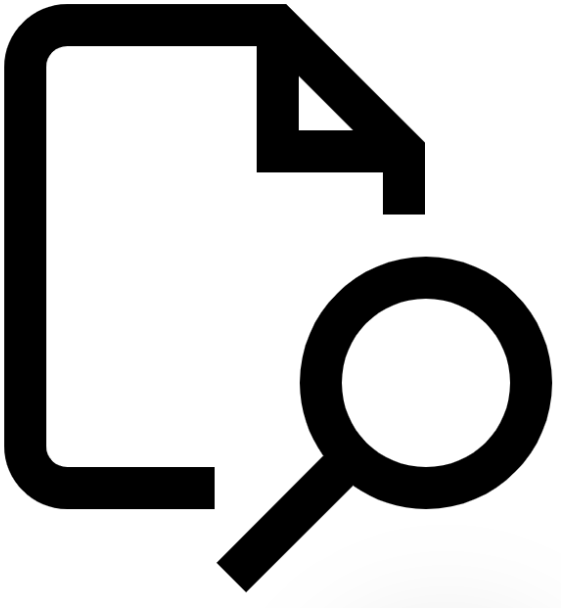
Project Report

Data Viewer



SDV602

By Travis Byrman

Contents

Milestone One

Brief

Overview

I am going to create an application that enables users to view graphs based on a data source. When a user adds a data source to the application, the user will be given graphing options on how they would like to view the data. This application will be a useful tool for data analysts, allowing them to have a better understanding of the data. The data source that this application will be using is sourced from Geonet about earthquakes.

Geonet: www.geonet.org.nz

For the client-side user interface I will be using Python as the programming language, tkinter as the interface library and matplotlib as the graphing library. For the back-end server I will be using a MySQL database to store all the data. To retrieve and send the data I will be using a JavaScript web API.

Purpose

The purpose of this application is to provide users with an efficient way of viewing and analysing data. This application will achieve this by providing users with a variety of ways to graph the data. Instead of manually creating graphs in excel, users can quickly and effectively create a graph using this application. As well as this, my application will provide additional tools that users can use to interact with the graphs, such as zoom and panning. Users will be given the ability to quickly change datasets. The application with have buttons that allows users to view the different datasets. Users can view the data on a graph, by selecting a graph type from a list of available types, as long as the data supports the graph type. Multiple users will have the ability to view the same dataset. A chat is available for users that want to communicate with other users on the application.

Requirements

Login – My application must have a way for users to register and account and login. Only users that have an account can use this application. To login, the user must enter their username and password. Every user will have a unique username, no account will have the same username or email address. This is to ensure that there are no duplicate accounts, created by the same person. A username will uniquely represent a user.

Registration – As my application has login functionality users will have the ability to register/create an account. To create an account users will need to enter a username, password, and email address. The username and password will be how the user logs into their account. The email address will be used by administrators if they need to contact the account holder.

Data Explorer Screens (DES) – These screens are what the user will see when they are viewing data. For my application there will be three data explorer screens that users can alternate between and view. Users can view these screens as different forms of graphs. Also on the DES, users will have a summary of the data and a chat will be available for users to chat with other users.

Default Data Explorer Screen - If a data source or graph is not selected, then the graph will be blank. A summary will only be visible if a data source is selected.

Data source – The application will show information/data from a source. The data source will be kept in the database. The data used is decided by the user.

Data Summary – Each DES will have a summary box that will show key information about the data source. Information such as axis values will be shown here.

Graphs – Users can create graphs from the data source that they provide. Users will have the ability to choose the type of graph that they want to create.

Graph toolbar – Users are provided with graphing tools that they can use to interact with the graph. The toolbar has 7 features:

- **Home button** – This resets the graph view to default.
- **Previous display button** – This shows the previous view of the graph.
- **Next display button** – This shows the next graph view. (after changes)
- **Panning button** – This enables the user to interactively pan/move the graph.
- **Zoom button** – This allows the user to interactively zoom in and out of the graph.
- **Settings button** – This enables the user to view a settings panel that changes the graph layout.
- **Save button** – This allows the user to save the current graph view as an image.

Multiuser – This application will be accessible by multiple users. Multiple users can use this application at one time and can view the same DES.

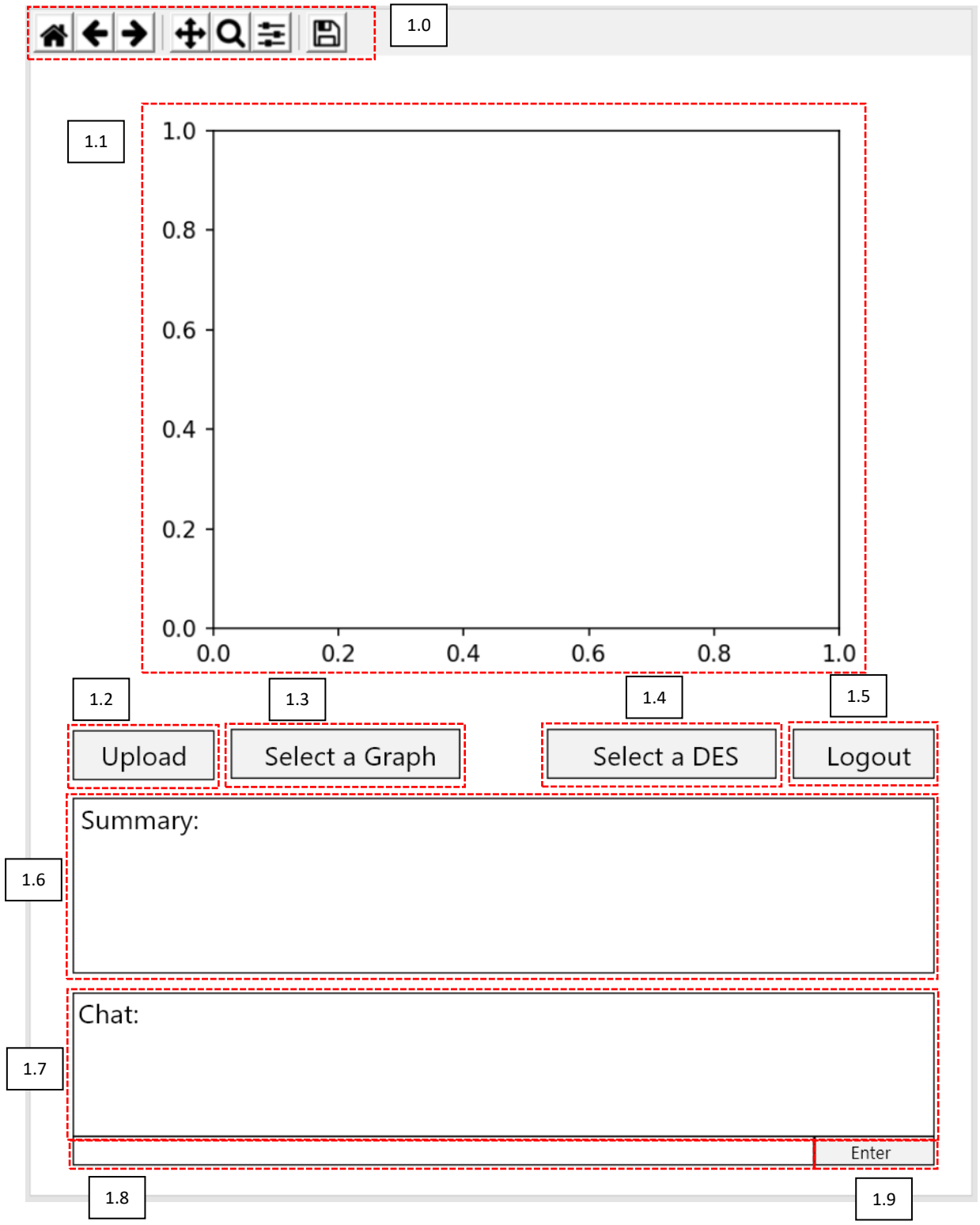
Chat – This gives the user the ability to talk with over users that are using the application. This chat is a global chat that every user can see and have access to regardless of what DES they are viewing.

Navigation – To navigate between that data explorer screens and the graph types, the application will have a navigation specifically for these actions. The application will have two combo boxes. One combo box will have the selection of data explorer screens, while the other will have the selection of graph types. Users can select the DES and graph that they want to view by click the corresponding combo box and selecting the item. If the graph or data explorer screen is invalid the request will not happen, and the graph will be blank.

In addition to the navigation there will be two other buttons, a logout and upload button. The logout button will log out the user from their account, and the upload button will provide the user a way of uploading a dataset. I will add an upload button as I am considering this as a possible way to allow the user to add their own data set. Although depending on the future of this application, I may remove this feature and add a more appropriate feature in its place.

Storyboard

Storyboard 1: Default DES view



1.0 **Toolbar** – Click to interact with the graph.

1.1 **Graph** – Displays the data as a graph.

1.2 **Upload button** – Click this to upload a data source.

1.3 **Graph combo box** – Click this to change the graph type.

1.4 **DES combo box** – Click this to change the DES/data set.

1.5 **Logout button** – Click this to exit/logout of the application.

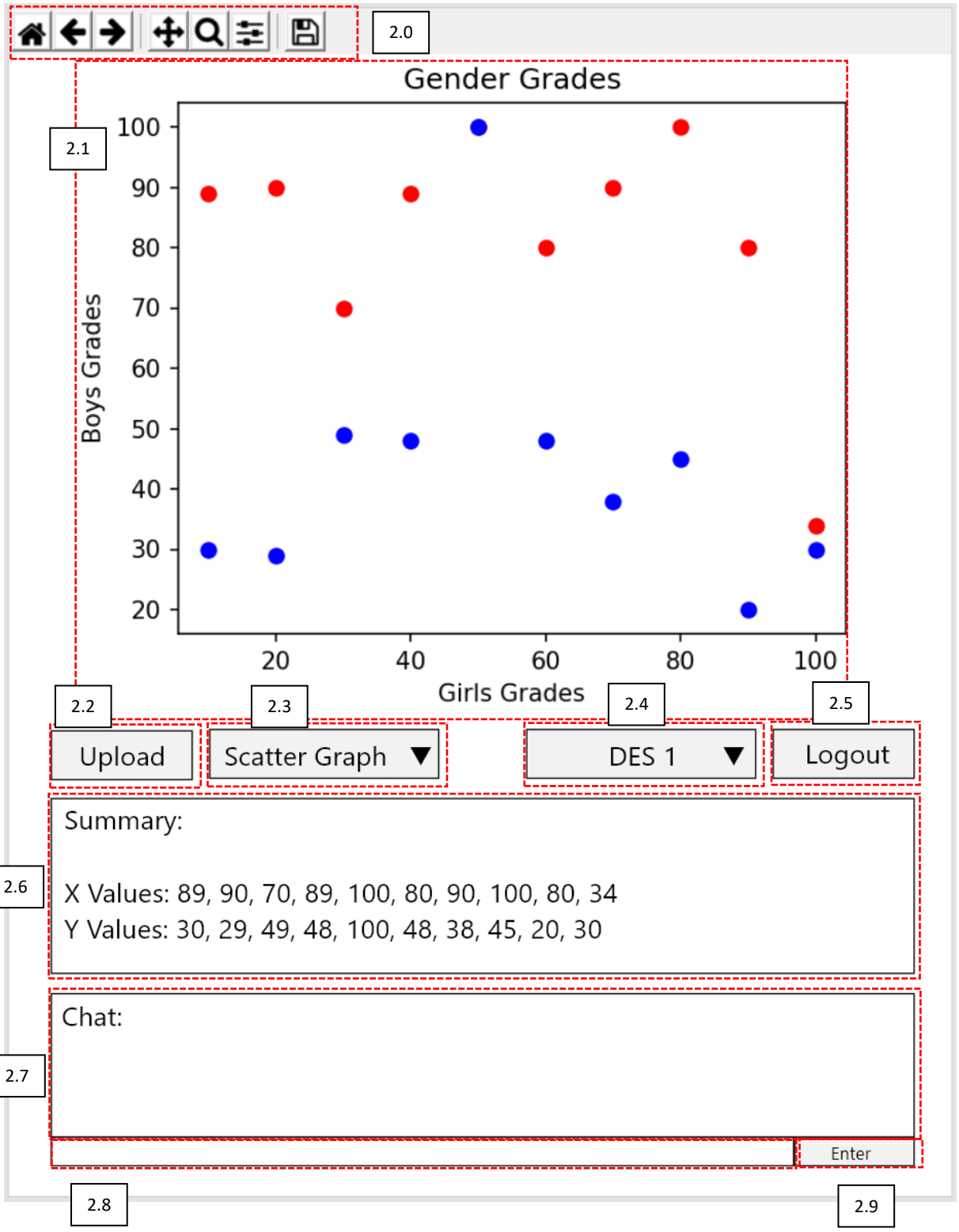
1.6 **Summary text box** – Shows the data summary.

1.7 **Chat text box** – Shows the chat entries.

1.8 **Chat entry box** – Input field to type chat messages.

1.9 **Send chat entry** – Click to send chat entry.

Storyboard 2: Data viewer DES 1



2.0 **Toolbar** – Click to interact with the graph.

2.1 **Graph** – Displays the data as a graph. Currently a scatter graph

2.2 **Upload button** – Click this to upload a data source.

2.3 **Graph combo box** – Click this to change the graph type.

2.4 **DES combo box** – Click this to change the DES/data set.

2.5 **Logout button** – Click this to exit/logout of the application.

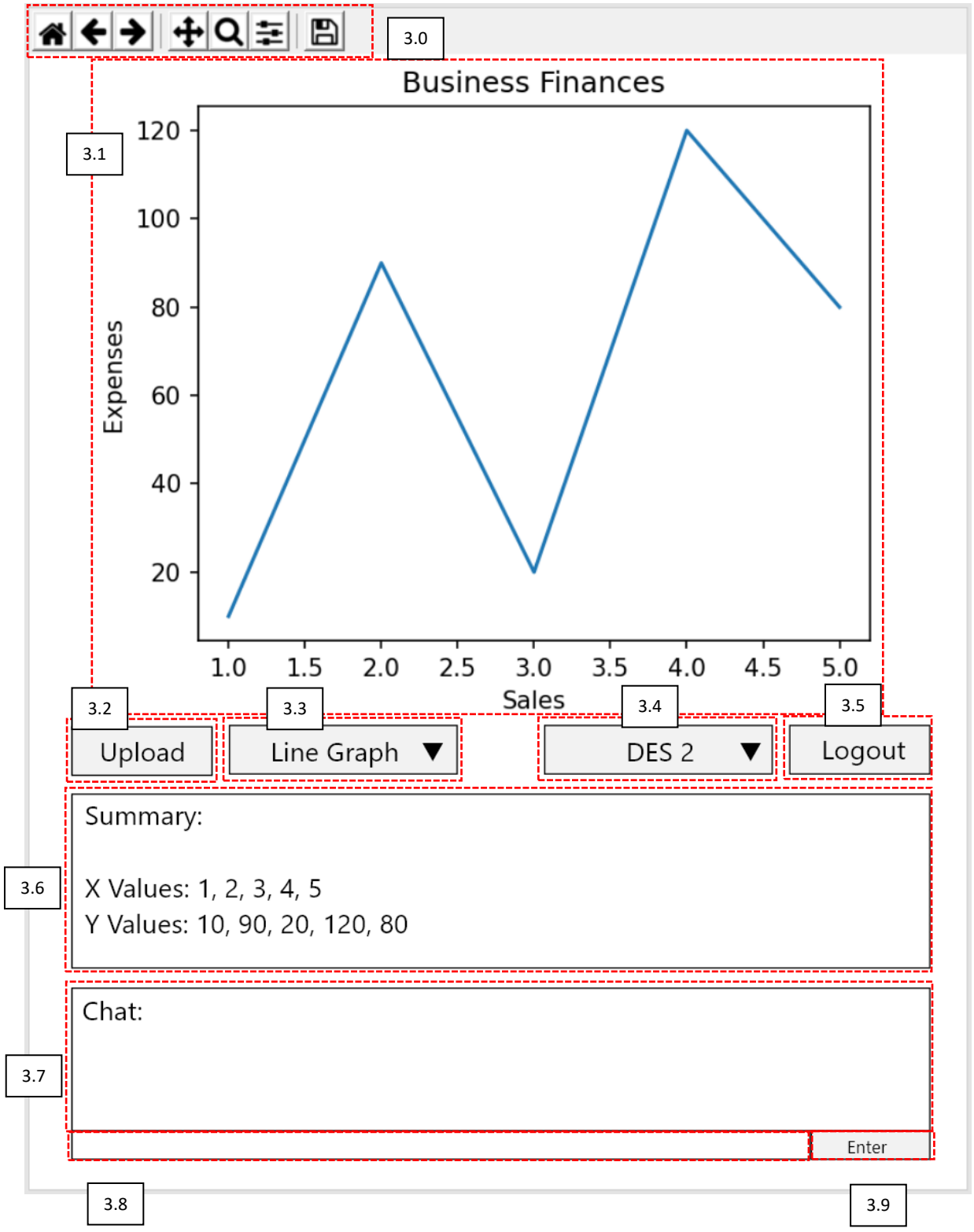
2.6 **Summary text box** – Shows the data summary.

2.7 **Chat text box** – Shows chat entries.

2.8 **Chat entry box** – Place to type chat messages.

2.9 **Send chat entry** – Click to send chat entry.

Storyboard 3: Data viewer DES 2



3.0 **Toolbar** – Click to interact with the graph.

3.1 **Graph** – Displays the data as a graph. Currently a line graph

3.2 **Upload button** – Click this to upload a data source.

3.3 **Graph combo box** – Click this to change the graph type.

3.4 **DES combo box** – Click this to change the DES/data set.

3.5 **Logout button** – Click this to exit/logout of the application.

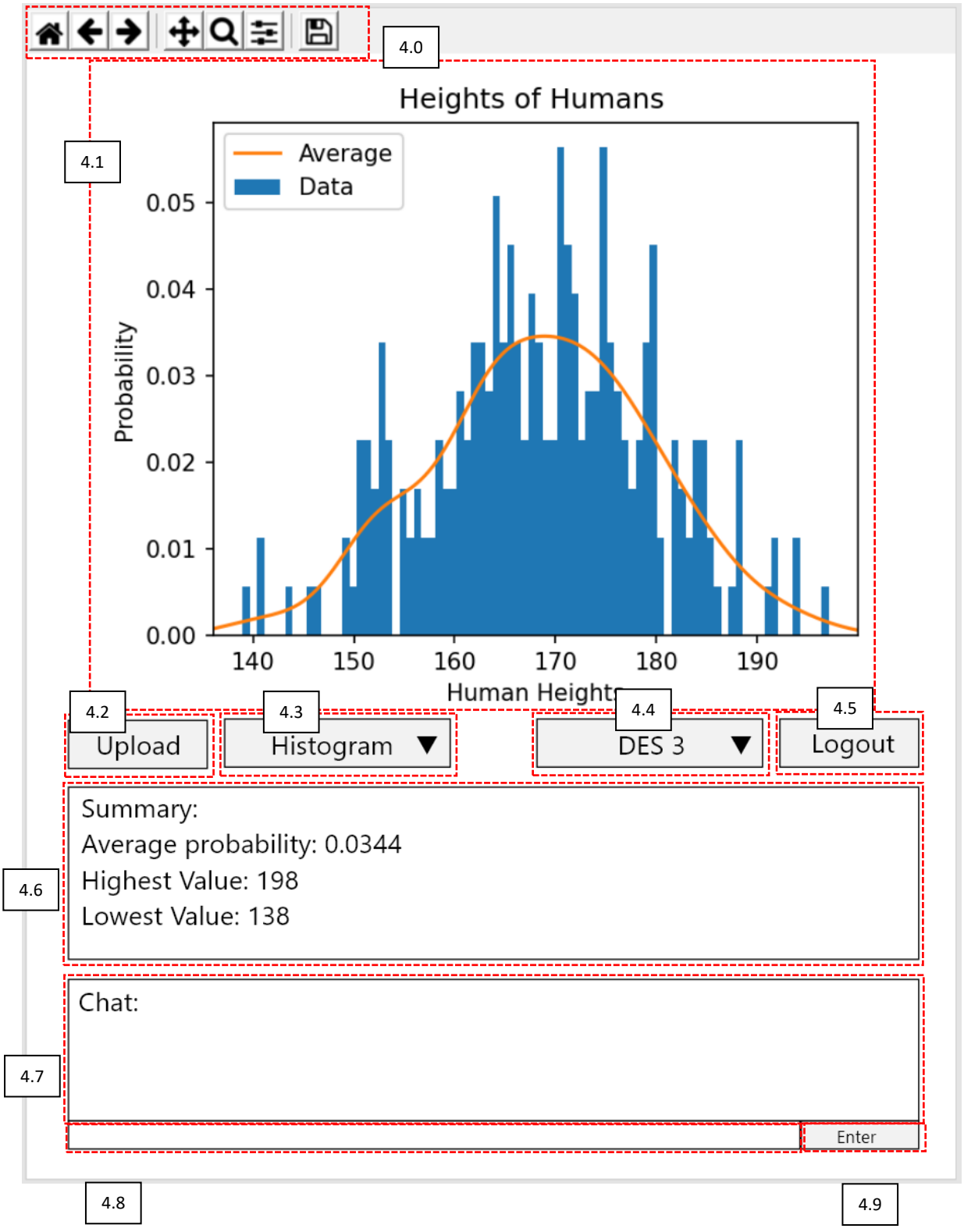
3.6 **Summary text box** – Shows data summary.

3.7 **Chat text box** – Shows chat entries.

3.8 **Chat entry box** – Place to type chat messages.

3.9 **Send chat entry** – Click to send chat entry.

Storyboard 4: Data viewer DES 3



4.0 **Toolbar** – Click to interact with the graph.

4.1 **Graph** – Displays the data as a graph. Currently a histogram.

4.2 **Upload button** – Click this to upload a data source.

4.3 **Graph combo box** – Click this to change the graph type.

4.4 **DES combo box** – Click this to change the DES/data set.

4.5 **Logout button** – Click this to exit/logout of the application.

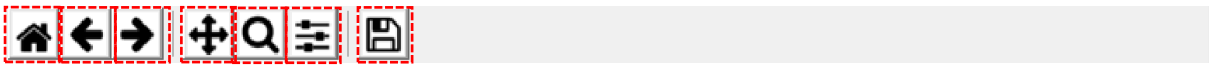
4.6 **Summary text box** – Shows the summary of data.

4.7 **Chat text box** – Shows chat entries.

4.8 **Chat entry box** – Place to type chat messages.

4.9 **Send chat entry** – Click to send chat entry.

Storyboard 5: Data viewer toolbar



(From left to right)

5.0 **Home button** – Resets the graph view to default.

5.1 **Previous display button** – Shows the previous view of graph.

5.2 **Next display button** – Shows the next graph view. (after changes)

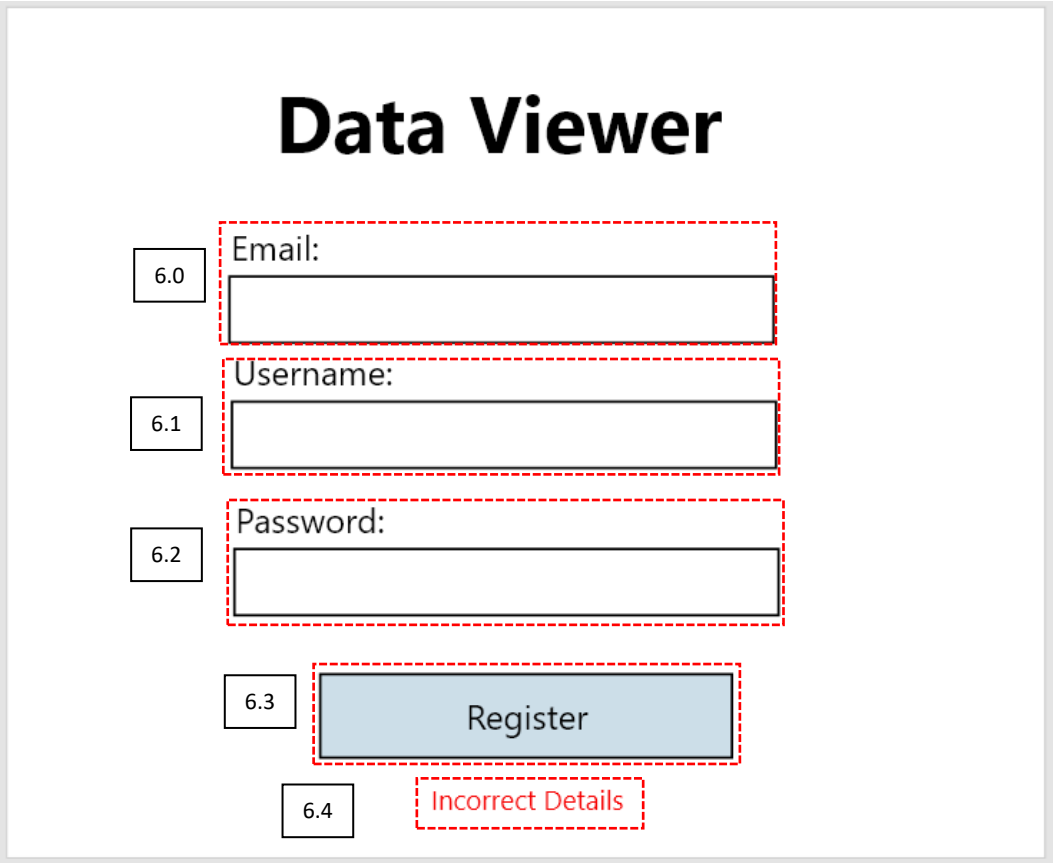
5.3 **Panning button** – Click to interactively pan the graph.

5.5 **Zoom button** – Click this to interactively zoom in and out of the graph.

5.6 **Settings button** – Click this to view a settings panel that can change the graph layout

5.7 **Save button** – Click this to save the current graph view as an image.

Storyboard 6: Account Registration Screen



6.0 **Email field** – Entry field where a user enters their account email address.

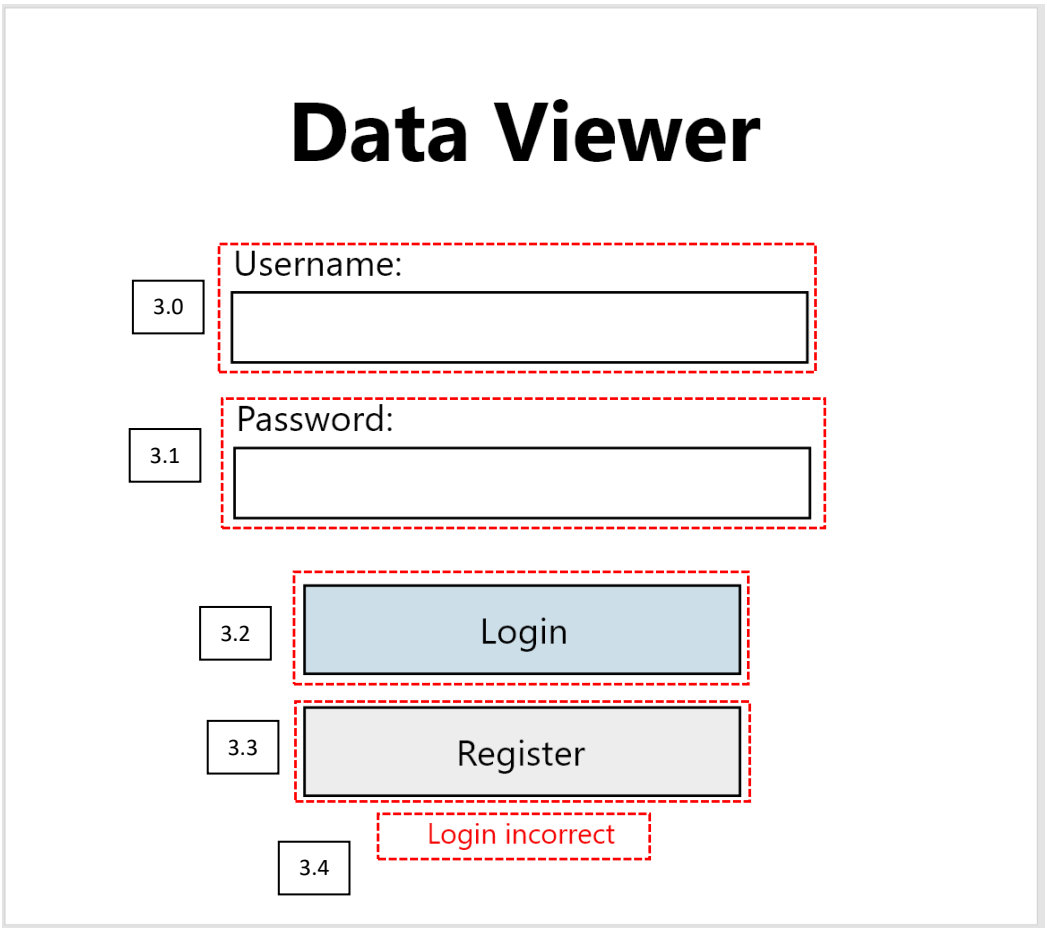
6.1 **Username field** – Entry field where a user enters their account username.

6.2 **Password field** – Entry field where a user enters their account password.

6.3 **Register button** – User clicks this button to register an account. If the username or password is invalid, then the user will be notified.

6.4 **Error message** – Text message appears if registration details are incorrect.

Storyboard 7: Account Registration Screen Unsuccessful



7.0 **Username field** – Entry field where a user enters their player account username.

7.1 **Password field** – Entry field where a user enters their player account password.

7.2 **Login button** – User clicks this button to login. If a username or password is invalid, then the user will not be logged in.

7.3 **Register button** – User clicks this button to create an account. Storyboard 2 will be shown.

7.4 **Error message** – This message appears when the login details are incorrect. This will be shown if registration details are incorrect.

Prototype and Test scripts

In addition to this documentation, I have attached a prototype of the user interface that this application will use. This prototype has a working chat, graph, graph toolbar and navigation for the data explorer screens. I have also created functions that work and are attached to the user interface elements.

Currently the prototype has already set data sets for the data explorer screens and the feature that allows users to add/upload their own data set is not set up.

The prototype currently uses the following libraries:

- **Tkinter** – I am using this library to create the user interface. With Tkinter I am able to use a range of widgets that help me to achieve the requirements. Alongside this I am using the Tkinter place layout manager to position my elements. I chose this layout manager as it allows me to place widgets using an x and y coordinate position. I am also using Tkinter for its choice of fonts.
- **Matplotlib** – I am using this library to create the graphs from the dataset. With this library I am able to add a graph toolbar to the user interface. Matplotlib also allows users to interact with the graphs which is a nice and useful feature to have.
- **Numpy** – Numpy is a mathematical library that allows Python to use a wider range of mathematical functions. I am using this to help create my graphs.
- **Scipy** – Scipy is another mathematical library that allows python to use a greater range of mathematical equations.

Milestone Two

For this milestone I implemented CSV file merging and the ability to add local CSV files to the GUI for graph data. I have also changed the GUI so that you can view multiple data explorer screens with different data sets. I also corrected my file/folder structure.

The dataset requirements are as follows:

- The document needs to be in a .csv format.
- The first row of the CSV file must contain a header
- Two or three data columns can be used with this application.
- When merging, select the first file, second file and a save location. This feature merges two files together and creates a csv file containing both sets of data. The headers from the first csv file chosen will be used as the headers for the merged data.

Additional packages used:

- Python inbuilt csv package.

To run the application, please run main.py

I have also made altercations to better suit my application, the largest addition is another textbox that will show the file location of the dataset.

In the future when I add a database to it I can use this to provide a graph title or a link to the dataset in the database.

Milestone Three

For this milestone, I implemented a remote database that would retrieve and save data.

When a user uploaded a CSV file from their desktop, the data would be stored onto a remote database. When a user selects a data source from the application, the data source would be retrieved from this data source.

I also introduced a login and registration system where users are required to create an account and login before they gain access to the application.

I also implemented a chat system to the application where users can chat to other users on the application. All messages were saved and retrieved from the same remote data source that the data was being retrieved from.

Below is my tabular comparison of Python.

Introduction

From this course I have learnt how to code in a variety of languages and have become quite familiar with a wide range of them. From my experience, I will be comparing python and its programming environment to JavaScript, a programming language that I been using quite a lot through my studies.

Python

Python is an object-oriented programming language created by Guido van Rossum in 1991. Python was originally a side project for Rossum as he was looking for a project to focus on during Christmas. At the time Rossum was working on Amoeba, a microkernel-based distribution system for system utilities. While working on them he found that programming in C took too much time and was difficult to learn. Rossum created Python based by ABC, as Rossum helped develop ABC earlier in his career, and found the language was easy to use. While Rossum was working on ABC, he didn't like some of the features it had, and saw that there were lots of complaints about bugs and features from users. (Exyte, n.d.)

Seeing this, Rossum developed Python following these 4 goals.

- To be an easy and intuitive language that is as powerful as other major languages.
- Open source, anyone can further develop it.
- Code is understandable in plain English.
- Suitable for everyday tasks.

(PythonInstitute, n.d.)

As Python was a modified version of ABC, Rossum re-used the well-liked syntax of ABC, and just fixed the complaints, bugs and features people didn't like. He also added features that he thought he thought ABC was missing such as lambda, map and filter. (GeeksForGeeks, n.d.)

Rossum created this project using resources provided from Centrum Wiskunde & Informatica (CWI), the institute he was working at, and agreed to have the project published open source. This helped Rossum while developing Python as he was able to gain help from developers from all around. (GeeksForGeeks, n.d.)

Rossum came up with name language ‘Python’, as he was a huge fan of an old BBC comedy sketch called ‘Python’s Flying Circus’. Rossum was the ‘Benevolent dictator for life’ up until 2018 and continues to have a huge impact in the language’s development and releases. (JavaTPoint, n.d.)

Currently Python is being maintained by the Python Software Foundation. This is a non-profit organisation that is responsible for the intellectual property and development of Python. (PSF, n.d.)

C#

C# is an object-orientated component-based programming language developed by Microsoft using the .NET framework architecture. C# was created in 2002 by a development team lead by Anders Hejlsberg. C# is based on C++ and Java but has many extensions which allows it to have a component orientated programming approach. (Medium, 2017)

At the time, Microsoft had the sales strategy to embrace successful technologies, then bundle those technologies with Windows to make Windows better. When Sun, the founding company of Java, released Java, Microsoft saw the potential in the technologies and wanted to implement them. Microsoft then implemented their own modified version of the Java Virtual Machine, one of Java’s technologies, but was sued by Sun for incompletely implementing their technologies. Microsoft then either had to fully implement Sun’s technologies as it is, which would give Sun the advantage of being in the Windows environment or create their own language, which would guarantee them backwards compatibility, and full development control. So, Microsoft formed their own team of developers with a range of experiences and created their own programming language (C#) and environment (.NET Framework). Microsoft then removed all of Java’s technologies from Windows and implemented C# and .NET. (Shappir, 2018)

C# is still being maintained and developed to this date. As C# and .NET is open source, anyone can contribute to the project, and help add features and fix bugs. C# is still owned by Microsoft.

Python vs C# comparison

The main difference between the two languages is that C# has the sole purpose of creating graphical user interfaces due to its component-based approach. C# is best to be used when creating a graphical application as you can implement graphical components to create games and windows-based applications.

Although Python can create graphical applications, it is much easier to use C# and the .NET Framework because of its easy-to-use support for adding and creating components.

Python is also statically typed compared to C# which is dynamically typed. This means at runtime, variables and expression types are already defined in Python, and for C# the types are variable. s

Python: ‘int a;’ – only accepts and integer

C#: ‘var a;’ – accepts any value

Python’s syntax is also a lot easy to learn. Python’s syntax is straightforward and requires much less lines than C# to do basic functionality, like print a line. Although a benefit of C# is that it is very similar to Java and C so if you already are familiar with those languages, you will not have much difficulty learning C#. (Scully, 2020)

Overall, if you are looking to learn programming for the first time, you are better off using Python as it’s easy to use, easy to learn and has a great syntax. Although if you already have basic

programming knowledge and would like to create GUI applications, then C# would be a better choice.

Python Libraries

Python has the ability to import code from a package manager. The code imported is called libraries, and provides Python with more features and abilities, that it doesn't already have. For example, if I wanted to create a GUI using Python, I would need to import a library, like tkinter, to make such a system feasible. Libraries are used to reduce the amount of code that you have to write. Everything that is in a library can be hard coded in Python, but would require a lot of work and time, compared to just importing a library and using its functions. Libraries tend to be open source, meaning that anyone can contribute to or create feedback/suggestion tickets for the library. Once a library has been added you can simply just call the function directly.

There are a couple of ways you can add a library to an application.

'from tkinter import *' – the from clause is where you specify the library, and the import clause allows you to import a single function/variable. Alternatively, you can use * to import all the functions/variables the library has to offer.

'import tkinter as tk' – here the import clause specifies the library, and the as clause specifies how the library should be called. If you are using this method, this means you need to specify the library before you call the function. For example: tk.function(example).

Integrated development environments

An integrated development environment (IDE) is software used for build applications and software. It contains a wide range of developer tools that help the development of the application.

Every IDE has three main tools:

Code Editor – A text editor where a developer writes the code.

Local build automation – The ability to compile, build and run an application locally, in a development environment.

Debugger – A tool used to test applications, notifying the developer if theirs and error or bug and where it is.

IDE's allow developers to quickly create an application. Although you can use just a normal text editor, it would be much more difficult to create an application, if not impossible. For example, to build a C# application, you need an IDE where it supports the .NET framework and C# programming language.

Each IDE has a range of programming languages its tools support. There are some IDEs that support almost all programming languages such as Visual Studio Code, whereas there are some IDEs like Sublime Text that only support, HTML, CSS, and JavaScript.

One of the more favourited IDEs to use is Visual Studio Code due to its ease of uses and support for extensions. This IDE allows developers to download extensions that improve their workflow. Through these extensions you can also download additional syntaxes such as MySQL that provide extra debugging, and support while building an application.

(Red Hat,n.d.).

Strengths and Weakness of the Python Programming Environment

Python has a lot of strengths and weaknesses in relation to the programming environment.

Strengths

Easy to learn – Python is very easy to learn and is a great programming language to begin learning from.

Free to learn – Python does not cost anything and is supported by a variety of IDEs.

Portable – Python is support across many devices, operating systems and environments.

Extendable – Python has a wide range of libraries that enable new functionality and capabilities.

Object-Orientated – Python supports both object-oriented and procedure-orientated approaches.

Database Connectivity – Python supports all types of databases for applications.

GUI Support – Python has the ability to create GUI's.

Weaknesses

Speed – Python is executed line by line, making it slower than other programming languages like C.

Memory Consumption – Python consumes a lot of memory due to its data structure. Applications that are limited by memory may cause an issue.

Mobile Development – Python isn't really good at mobile development and python doesn't support mobile development as much as client and server-side development.

Database Access Issues – Python has issues with database access layers which can be restricting.

(KCP LMS, n.d.).

References

About Python. (n.d.). Python Institute | Python Training and Certification Programs. Retrieved November 16, 2021, from https://pythoninstitute.org/what-is-python/

BillWagner. (n.d.). *A tour of C# - C# guide*. Developer tools, technical documentation and coding examples | Microsoft Docs. Retrieved November 16, 2021, from https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

A brief history of Python. (2020, November 3). Mobile, Blockchain, and AR/VR Development Agency | Exyte. Retrieved November 16, 2021, from https://exyte.com/blog/a-brief-history-of-python

C# history. (n.d.). www.javatpoint.com. Retrieved November 16, 2021, from https://www.javatpoint.com/csharp-history

C# vs Python: What's the difference? (2020, May 22). Career Karma. https://careerkarma.com/blog/csharp-vs-python/

(n.d.). Forbes. Retrieved November 16, 2021, from https://www.forbes.com/sites/quora/2018/03/02/why-did-microsoft-create-c/?sh=1539538170f3

History of C# programming language. (n.d.). C# Corner - Community of Software and Data Developers. Retrieved November 16, 2021, from https://www.c-sharpcorner.com/blogs/history-of-c-sharp-programming-language

History of Python. (2019, May 6). GeeksforGeeks. Retrieved November 16, 2021, from https://www.geeksforgeeks.org/history-of-python/

Mkhitaryan, A. (2017, October 13). *Why is C# among the most popular programming languages in the world?* Medium. Retrieved November 16, 2021, from https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb

Python history - javatpoint. (n.d.). www.javatpoint.com. Retrieved November 16, 2021, from https://www.javatpoint.com/python-history

Python software Foundation. (n.d.). Python.org. Retrieved November 16, 2021, from https://www.python.org/psf/

Strengths and weaknesses of Python. (n.d.). KCP: E-Learning. Retrieved November 16, 2021, from https://kcpelearning.com/showcourse/python3/strengths-and-weaknesses

What is an IDE? (n.d.). Retrieved November 16, 2021, from https://www.redhat.com/en/topics/middleware/what-is-ide