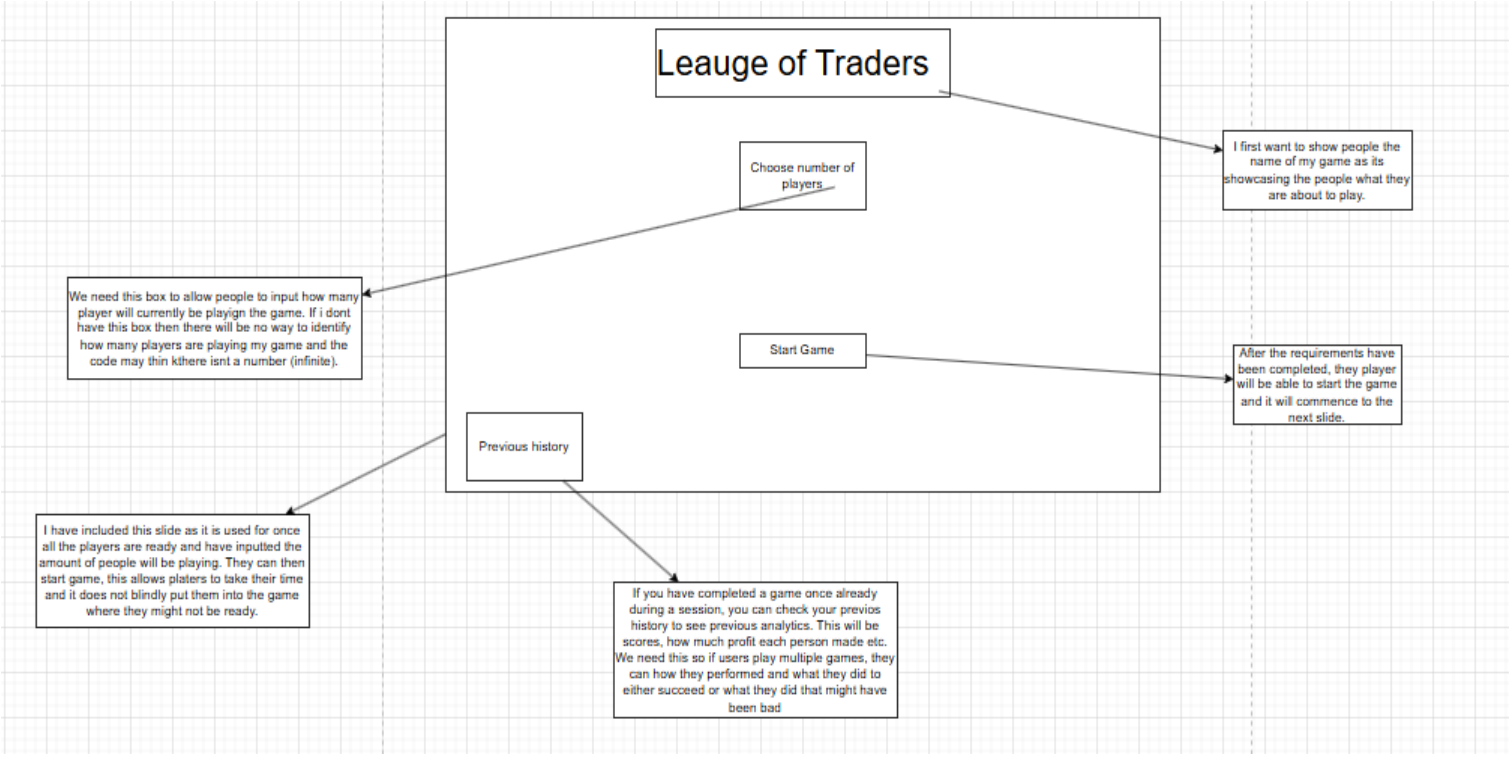
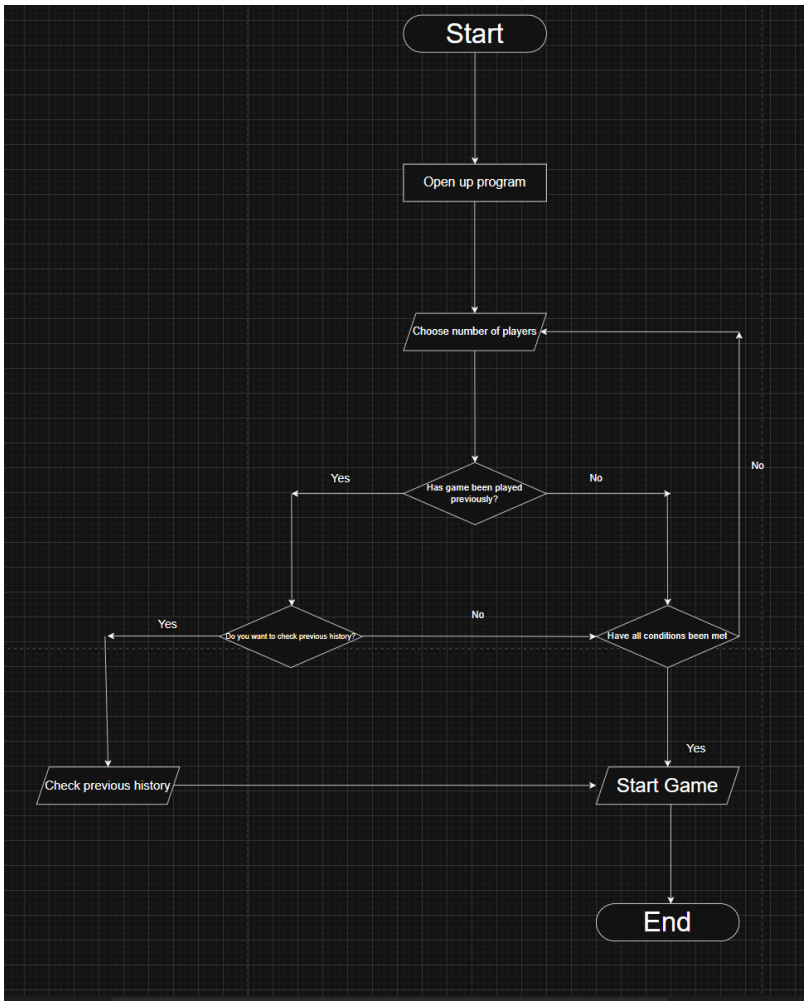
Iterative Development

Stage 1:



ALGORITHM:

Flow chart:



Iterative Development

The flow chart is used to show how exactly my menu will run. The start initialises the start of my program. Once that begins and you open the program, the first thing you should do is choose between the set number of players on the game, which is between 2 and 8. After that is done you then get sent to the next section which is if, if you have played a game before you will have the option to check your previous history. However, if its not done then you can start the game. Once you check history you can leave that section and get straight into the game.

Pseudocode:

CREATE window screen #Gievs a blank window pop up

SET windowTitle = “League of Traders”

SET windowIcon = “app.png”

SET windowBackground = “Trading.png” #Theses add details the window such as a background, logo and title.

DISPLAY label "League of Traders" #Will display the title of my game

DISPLAY label “Choose number of players” #Another label which is stating what a button should do

CREATE dropBox “Choose number of players” #This button is tied to the label above, telling the user, the number of players they should chose to play with.

CREATE push button "Start Game" #Once clicked, you will move to the next slide

IF previous history EXIST:

    CREATE pushbutton “Check History” #Another push able button to check history

Iterative Development

IF Start Game button CLICKED THEN

Print(“Game is starting with”, playerCount, “Players”) #Python that is outputted if someone clicked start game

IF Previous History button CLICKED THEN

RETRIEVE previous game data

DISPLAY previous scores and analytics #This will be outputted if you clicked the previous history button, on the basis that you have played this game before

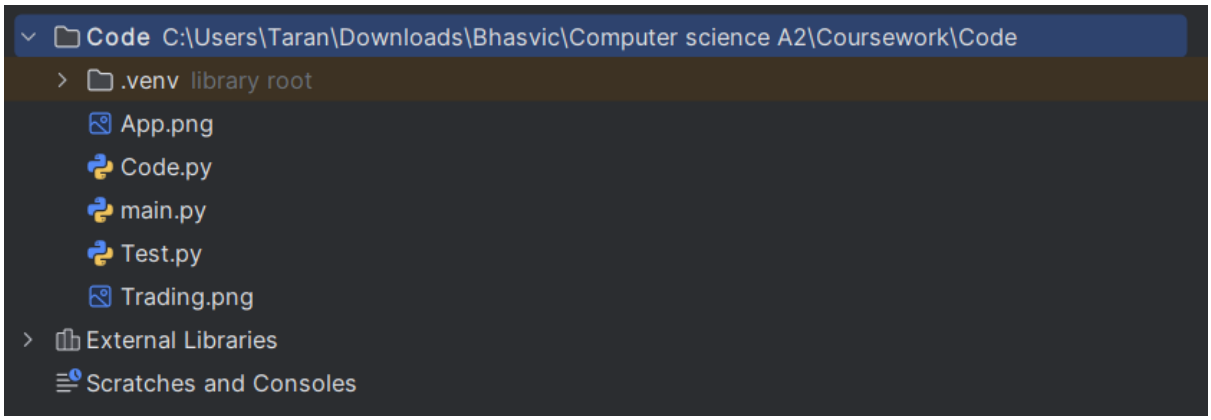
Iterative Development

Filed Name	Data Type	Data Format	Field Size	Description	Example
Field Name	**Data Type**	**Data Format**	**Field Size**	**Description**	**Example**
app	QApplication	Object instance	None	Initalises the start of your gui (game can start now)	app = QApplication(sys.argv)
window	MainWindow	Object instance	None	This the window that pops up in which the game is now starting	window = MainWindow()
MainWindow	class	Class Definition	None	This allows the tab to contain buttons, titles, inputs etc	class MainWindow(QMainWindow)
label_title	QLabel	String	1 - 20 characters	Piece of string that displays the title of my game	"League of Traders"
label_background	QLabel	QPixmap object	1 - 20 characters	This is the widget that gives my screen a dynamic background for my users to see	QPixmap("Trading.png")
button_start	QPushButton	String	1–20 chars	These are interactable buttons that perform an action, this one will allow you to go the next stage of the game	"Start Game"
button_history	QPushButton	String	1–20 chars	These are interactable buttons that perform an action, this one will allow you to check the previous histroy of your game, that is if you have played before	"View History"
combo_players	QComboBox	Integer or String options	1–8 options	This will act like a dropdown bar that gives you the ability to choose between 1 and 8 players	"4 Players"
icon_app	QIcon	File path	None	This is just a png that I can place for specific parts of my app such as its taskbar icon	"App.png"
background_image	QPixmap	PNG file	None	This will act as the background of my game	"Trading.png"
history_exists	Boolean	True / False	1 bit	This will basically be used to track wheather a previous game data has been there, so users can see how previous games went	FALSE
setGeometry()	Method	Integers (x, y, width, height)	4 parameters	Its indiacates the size of the window and where it is placed on your screen,	(700, 300, 1000, 1000)
setWindowTitle()	Method	String	None	Will display the title of the game	"League of Traders"
setWindowIcon()	Method	QIcon(path)	None	Gives the game an icon that it will be presented by instead of something that is not linked to the game	QIcon("App.png")
setPixmap()	Method	QPixmap object	None	Loads and applies an image to a QLabel.	label_background.setPixmap(QPixmap("Trading.png"))
setScaledContents()	Method	Boolean	1	Allows images to be resized to fit the screen of the game so they are always 'true to size'	TRUE
clicked.connect()	Method	Function reference	None	Connects a button to a function that will happen once you click it	button_start.clicked.connect(start_game)
start_game()	Function	User-defined	None	Launches the main game and takes you to the next slide	def start_game(): ...
view_history()	Function	User-defined	None	If game data has been stored, it will show previous analytical data	def view_history(): ...

Iterative Development

Code

I have first set up my PyCharm program, using pyqt5, which will contain all the needed files to be able to code this project.



Each of the 3 python files have a crucial function and needs to be there. The main file is where I will hold the code of how I make a single window, I have done this as I believe it will be a recurring function that appears, and having a file showing that code, in my opinion, is handy.

I then have the code file which will be used to show all the main code of my game, this being the first stage which is just implementing a menu which will be expanded on in the second stage.

Finally, I have a test file which will individually test a concept I want to implement such as a push button so it can work.

Iterative Development

```python
import sys #Python module that provides access to specific parameters
from PyQt5.QtWidgets import QApplication, QMainWindow #Imports from PyQt5 so I can develop my GUI

class MainWindow(QMainWindow): #Code used to make the window pop up  1 usage
    def __init__(self):
        super().__init__()
        self.setGeometry(700, 300, 1000, 1000) #Sets the size of the window

def main(): #Subroutine that allows the tab to open and be in that state until users closes it  1 usage
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

To begin with my code, I have imported the python system module, which will provide me access to parameters that are, system specific. This has been shown in my code by me passing command line arguments into the application. This is needed so sys.argv will be running with my PyQt5 application. This allows the program handle system level arguments and once at app is exited, it is closed without errors.

Then I now imported 2 key classes from PyQt5 library:

QApplication which manages my Gui

QMainWindow which will provide us the window and features that come with it such as widgets and menu bars

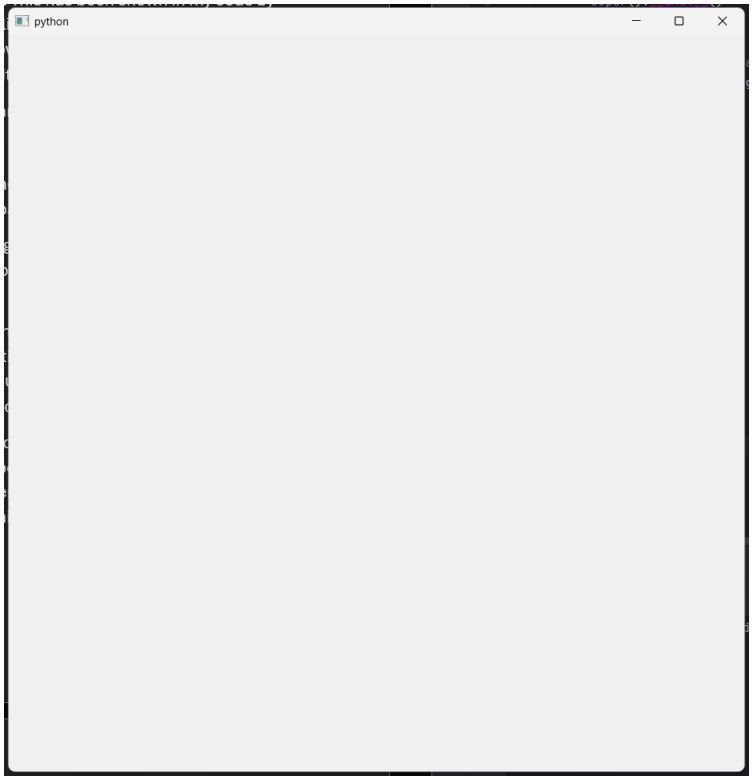
The reason behind using this is because I want my game to be developed with using PyQt5, I came to wanting to use this, because of how intricate you can make your tabs and its ease of use.

class MainWindow, inherits from QMainWindow. This will create my blueprint for my main window so it can allow me to start implementing other sections into my code. I have done this so the window can be shown to the user as it is a necessity. As I will be testing my code, I have set the window to be a size of 700 by 300 by 1000 by 1000

Finally, I created the main subroutine which will allow the program to ran. The code basically allows me to, once my main window is open, create a event loop so the actual GUI can stay active until it is eventually closed when the users want the program to end. Having this main function keeps the program organised and easy to reuse as it is a subroutine, for later parts of the code.

Iterative Development

With all this being programmed you will then be shown this once the program starts:



Next, after setting up my window, I wanted to start to create widgets for my game such as labels or buttons to interact with that have a purpose, however, before I could implement any of these interactive and dynamic elements, I first wanted and felt I needed to create a central widget and layout. This will be used to organise them in a proper manner. Furthermore, it ensures everything inside the window has a structured position that keeps it integrity:

```
import sys #Python module that provides access to specific parameters
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout #Imports from PyQt5 so I can develop my GUI
from PyQt5.QtGui import QFont


class MainWindow(QMainWindow): #Code used to make the window pop up  1 usage
    def __init__(self):
        super().__init__()
        self.setGeometry(700, 300, 1000, 1000) #Sets the size of the window

        central_widget = QWidget() #Creates a way to contain all widgets in window
        self.setCentralWidget(central_widget) #Tells the program all widgets must go inside main area of window
        layout = QVBoxLayout(central_widget) #Makes layout automatically vertical so it appears top to bottem
        layout.setSpacing(25) #It will automatically have my widgets be spaced out by at least 25
        central_widget.setFont(QFont("Arial", 16, QFont.Bold)) #Give all my widgets the same font so i don't have to automatically change them
        central_widget.setStyleSheet("""
                    QPushButton,QComboBox, QLabel {
                        border: 2px solid black;
                        border-radius: 5px;
                        padding: 5px;
                    }
                """)
```

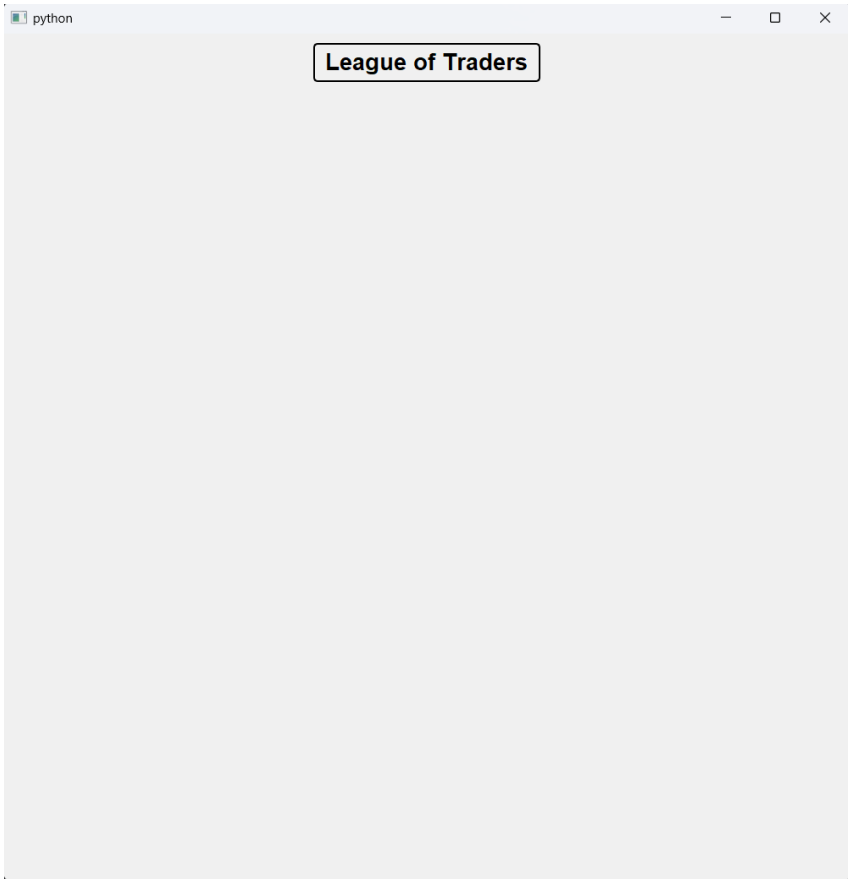
Iterative Development

To begin with, I have new imported another 2 functions from PyQt5.QtWidgets, being QWidget and QVBoxLayout. These functions will be used to later implement my widgets and for my widgets to automatically be assorted in an order such as vertical or horizontal.

I then added the Qfont function from PyQt5.QtGui which is a module that main aspect is to contain functions that assist in the design for a GUI such as a widget's font.

Then I begin to code the central widget. The code in that box is used to make somewhat of a container for all my widgets to be in. Then the latter part of a code is used to automatically give properties to these widgets so that I don't have to do them individually for each one. Such as their font, all being bold and all my widgets having a box around them. Also the spacing between each one being set to a minimum of 25 which can be changed if I want it to

Due to me not having anything to test it on, now I will be placing this code in the test section and creating a label to test it with. After putting in the needed detail I was able to then input a label shown here:



Iterative Development

After finding out that the code was a success, I am now able to start adding my other widgets, such as labels, push buttons, drop down buttons etc:

```
QLabel, QComboBox, QPushButton
```

I first added the 3 necessary functions to produce these widgets, which are labels, combo box (dropdown box) and pushbuttons.

```
from PyQt5.QtCore import Qt
```

Then I added the module PyQt5.QtCore which is currently holding Qt, which will help with being able to use widgets,

```
    title_label = QLabel("League of Traders") #Titles game league of traders
    layout.addWidget(title_label, alignment=Qt.AlignTop | Qt.AlignHCenter) # Align game to fit where i want it to be

    player_label = QLabel("Choose Number of Players:") #Some string to tell the player how many other members are in the game
    layout.addWidget(player_label, alignment=Qt.AlignHCenter) #Alligns it with where i want it to be

    self.player_dropdown = QComboBox() #Inital code for dropdown box
    for i in range(2,9): #For statement to give the ran of vales
        if i == 2: #If it's the smallest value
            text = "2 Player" # Output 2 players
        else:
            text = str(i) + " Players" #Otherewise do the output someone chose plus the string 'Players'
        self.player_dropdown.addItem(text) #Adds each item to the dropdown
    layout.addWidget(self.player_dropdown, alignment=Qt.AlignHCenter) #Alligns the dropdown box just under its string


    start_button = QPushButton("Start Game") #Startgame button
    layout.addWidget(start_button, alignment=Qt.AlignHCenter) #Allign start game

    history_button = QPushButton("View History") #View History button
    layout.addWidget(history_button, alignment=Qt.AlignHCenter) #Allign history button

    start_button.clicked.connect(self.start_game) #Allows the button to be push
    history_button.clicked.connect(self.view_history) #Allowes the button to be pushed

def start_game(self):  1 usage
    print("Game started with", self.player_dropdown.currentText()) #Text once you click the button

def view_history(self):  1 usage
    print("Viewing game history...") #Text once you click the button
```

Here comes the code of my section which is used to be able to be able to access these types of widgets. I first wanted to implement the easiest widget, which is label. It's the easiest as it does not need you to give it any other function but to just stay on the screen. The label was just going to show the game, league of traders, as I need to show the title of my game. This also goes for the next label which was a choose number of player's label.

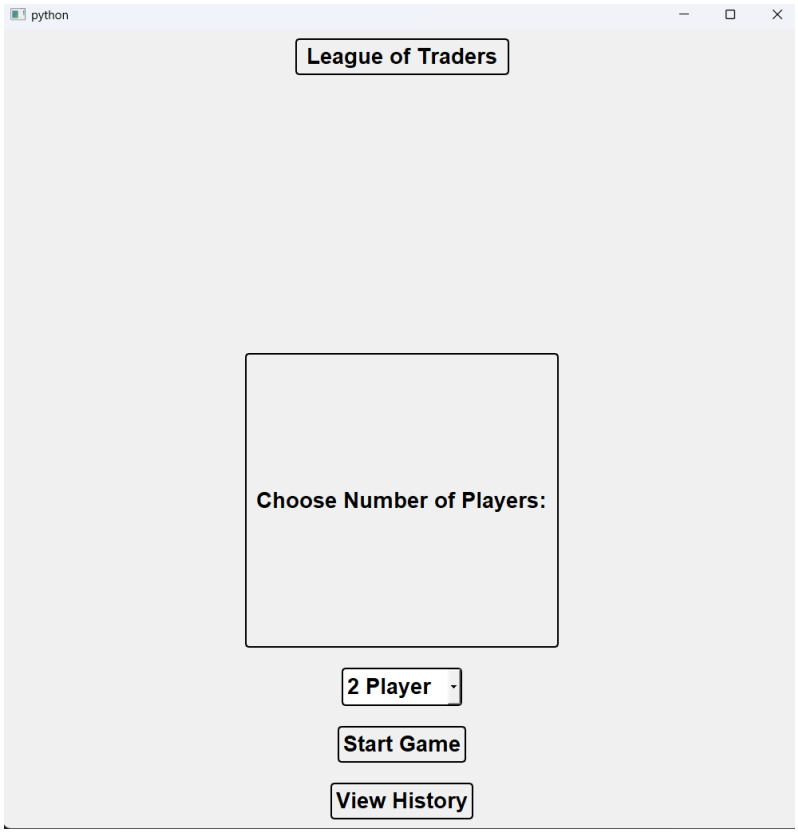
The next widget I added was my combo as it is slightly harder as now, I have to add some python. To add a dropdown box, I first initiated player dropdown making QComboBox now be known as that. After doing so I then went to create a for loop to state the range of players ill be choosing to be presented within the button. I made the range between 2 and 9 and made a if statement staying if it was 2 then you would see 2

Iterative Development

players else present the other players then the final piece of code for that section would add each item to the list and align the box to fit in the window.

Finally, I then added 2 push buttons, start game and history. Once creating them if you clicked start game it would out put game starting then with how many players are playing and if you pushed view history, it would allow you to view the history and say history is being viewed.

Once I tested this code, while it did work, I came across a couple issues. The choose player label box was very big and took over the space. Furthermore, I felt as if the title was not as 'out there' as I would like and wanted it to be slightly bolder than the rest. Finally, it was hard to tell whether my push buttons have been pushed unless you looked at the code, which most people would not do. This is what it currently looks like:

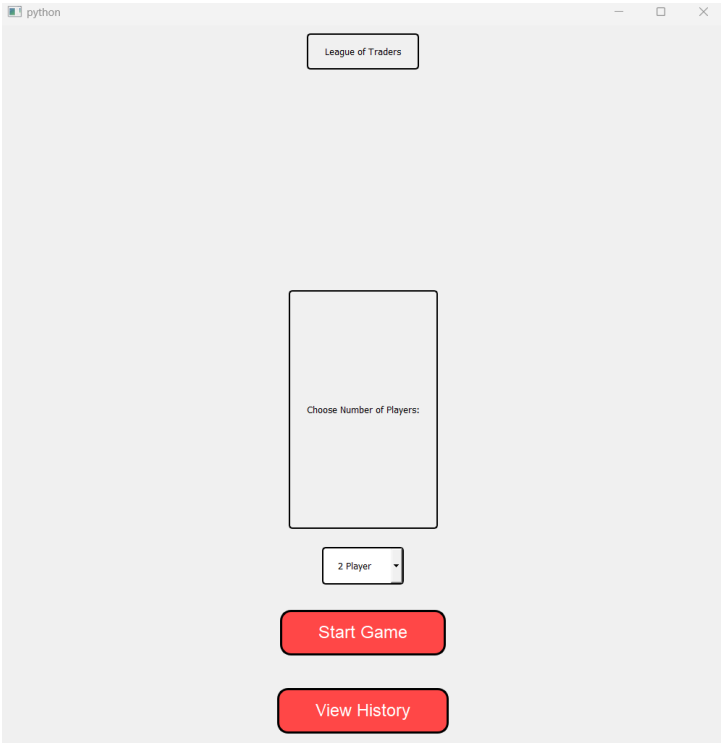


I will now be fixing these mistakes and showing the code, upon completion:

Iterative Development

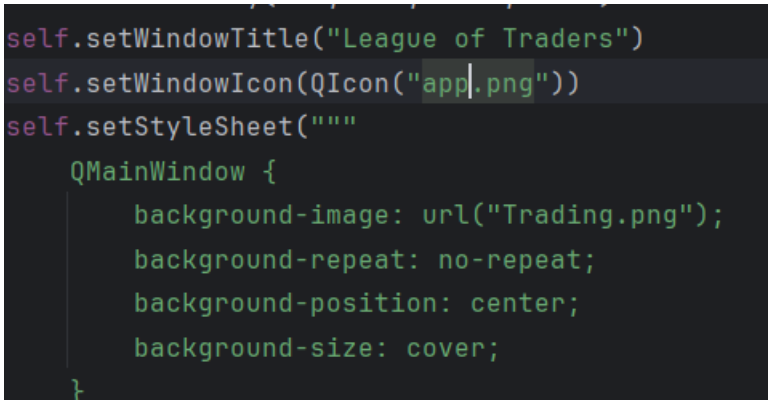
```
startButton = QPushButton("Start Game") #Startgame button
startButton.setObjectName("gameButton") #Assign object name for styling
layout.addWidget(startButton, alignment=Qt.AlignHCenter) #Allign start game

historyButton = QPushButton("View History") #View History button
historyButton.setObjectName("gameButton") #Assign object name for styling
layout.addWidget(historyButton, alignment=Qt.AlignHCenter) #Allign history button

# Apply custom style for all buttons with objectName "gameButton"
self.setStyleSheet(self.styleSheet() + """
    QPushButton#gameButton {
        font-size: 24px;
        font-family: Arial;
        padding: 15px 50px;
        margin: 10px;
        border: 3px solid black;
        border-radius: 15px;
        background-color: hsl(0, 100%, 64%);
        color: white;
    }
    QPushButton#gameButton:hover {
        background-color: hsl(0, 100%, 84%);
    }
""")
```

To begin with, I first wanted to assign both my pushbuttons as gamebutton so once I have to assign them in their style sheet I would not have to do it twice. Then I started to style them in their style sheet and when ever you hover on it, it will change different colours, so the user knows when they are on the button. With all this, my window finally looks like this:



Iterative Development

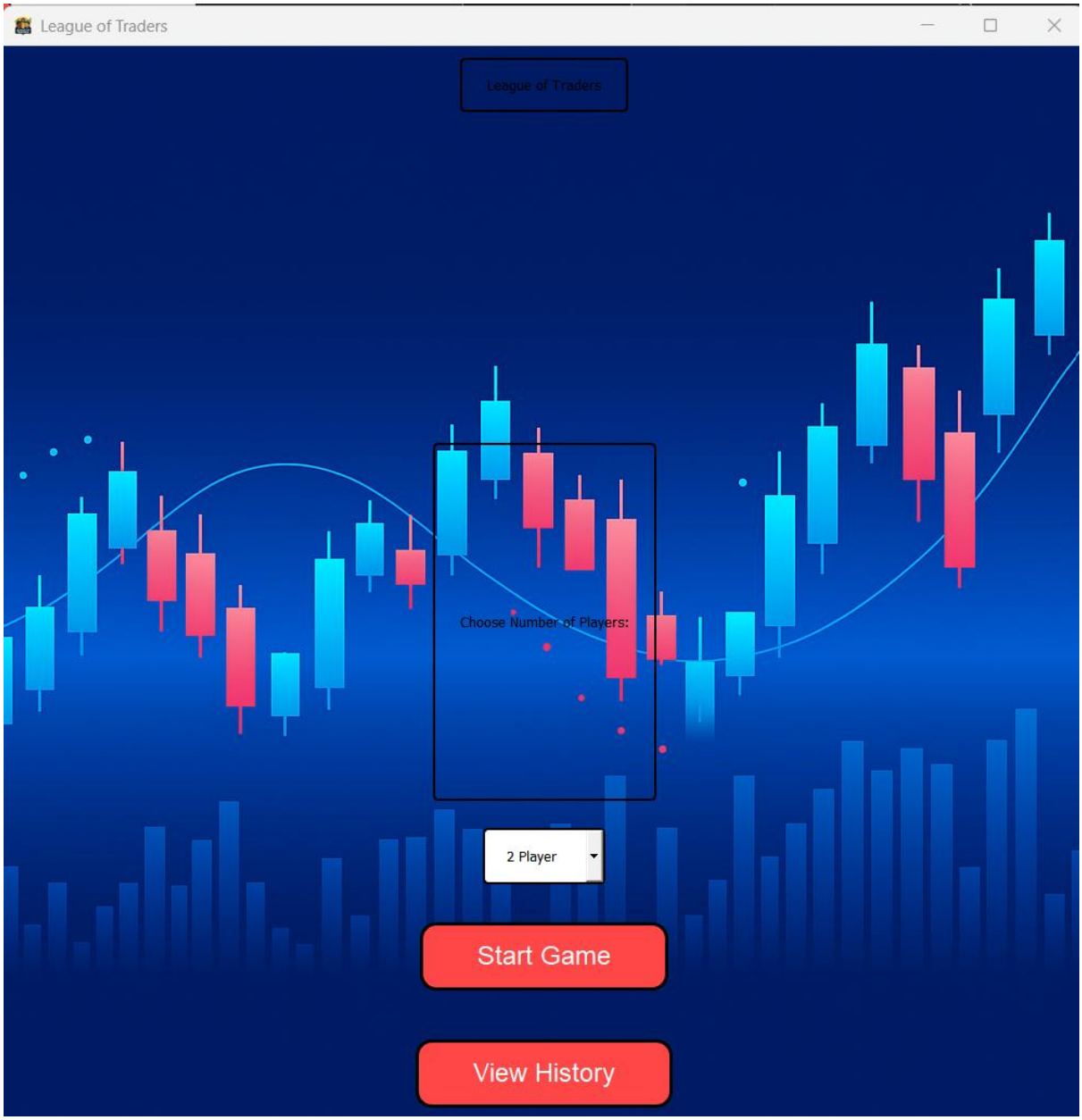
Finally, to make the game look slightly better, I am now going to be adding a background and icon for my stage one.



With this code im able to change the name of the app to the name of my game, the icon for the game is no one that I like and now I have a dynamic background behind the game.

This is the finished product of the first stage of the gamne:

Iterative Development



Through the second stage I will be adding a multitude of different things and remaking so of my concepts in the first stage to make them look better, such as some of the sections of this being unreadable, having the icon be shown on the taskbar, the input buttons doing something etc.

With the main aspects of my main menu completed, I wanted to additionally add a couple features that would make the game slightly more interactive, easier to use and just a lot better than it is right now.

Iterative Development

First to make the words on the game screen easer to read o decoded to fill in the 2 label boxes instead of leaving them clear:

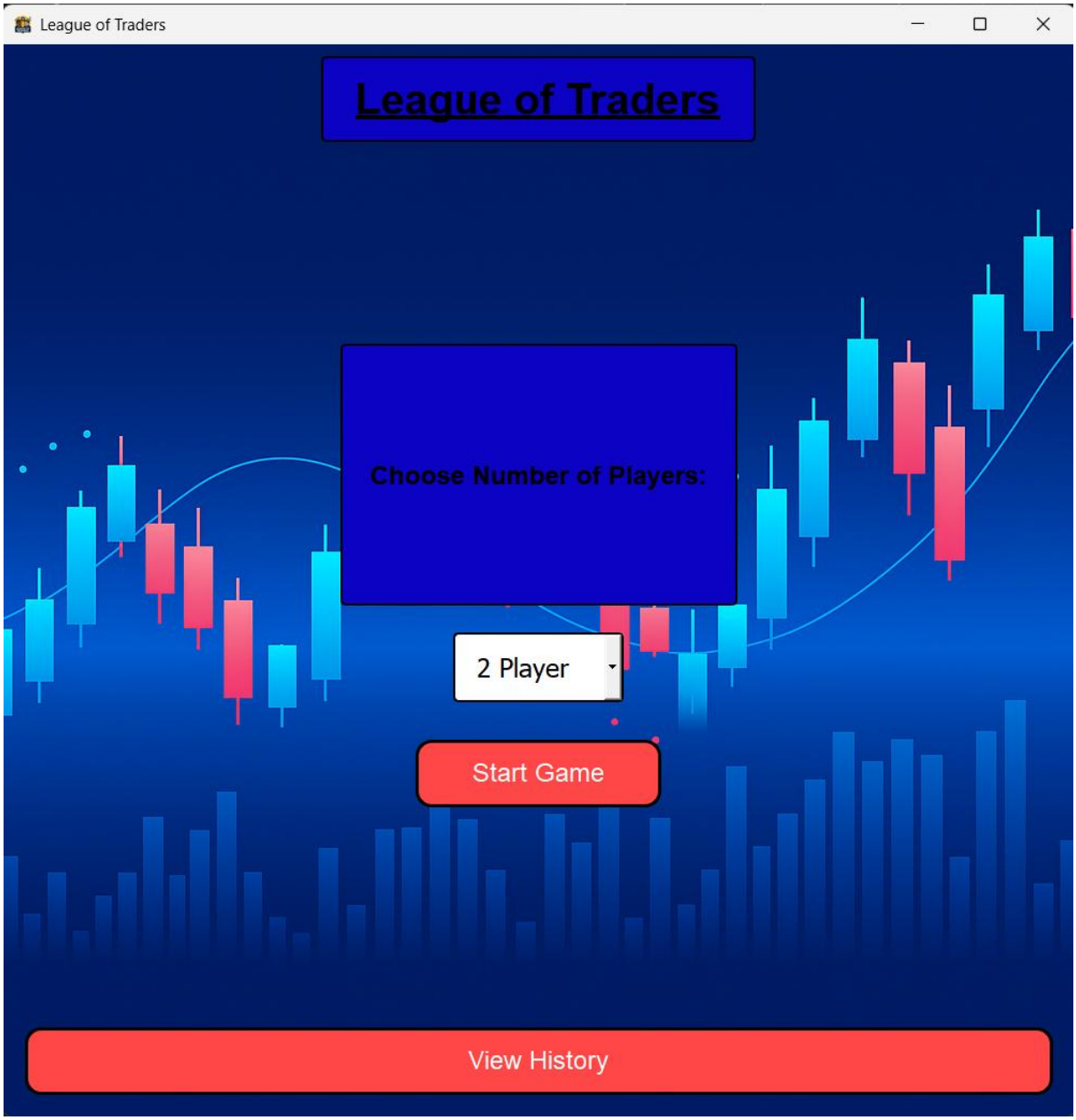
```
titleLabel = QLabel("League of Traders") #Titles game league of traders
layout.addWidget(titleLabel, alignment=Qt.AlignTop | Qt.AlignHCenter) # Align game to fit where I want it to be
titleLabel.setFont(QFont("Arial", 14, QFont.Bold))
titleLabel.setStyleSheet("background-color: #0c02c4;" #Blue fill on textbook to read title
                         "font-size: 40px;"
                         "text-decoration: underline;"
                         ) #Allows my title to be alot more bold compared to other sections of my screen



playerLabel = QLabel("Choose Number of Players:") #Some string to tell the player how many other members are in the game
layout.addWidget(playerLabel, alignment=Qt.AlignHCenter)#Alligns it with where I want it to be
playerLabel.setFont(QFont("Arial", 14, QFont.Bold))
playerLabel.setStyleSheet("background-color: #0c02c4;" #Blue fill on textbox to read label
                          )
```

An issue I came around after finishing the main aspects of my stage one was how unreadable some of the text was on my screen and how the title label looked very lack lustre. To counter this issue, while I still have the main features of my widgets still under the centralWidget object. For some labels that needed more features compared to others, I added the necessary details:

This was evident on both labels, such as adding a fill on my text boxes so they have a deeper shade of blue, this would allow my text to contrast well with the boxes, hence, making it easier to read.

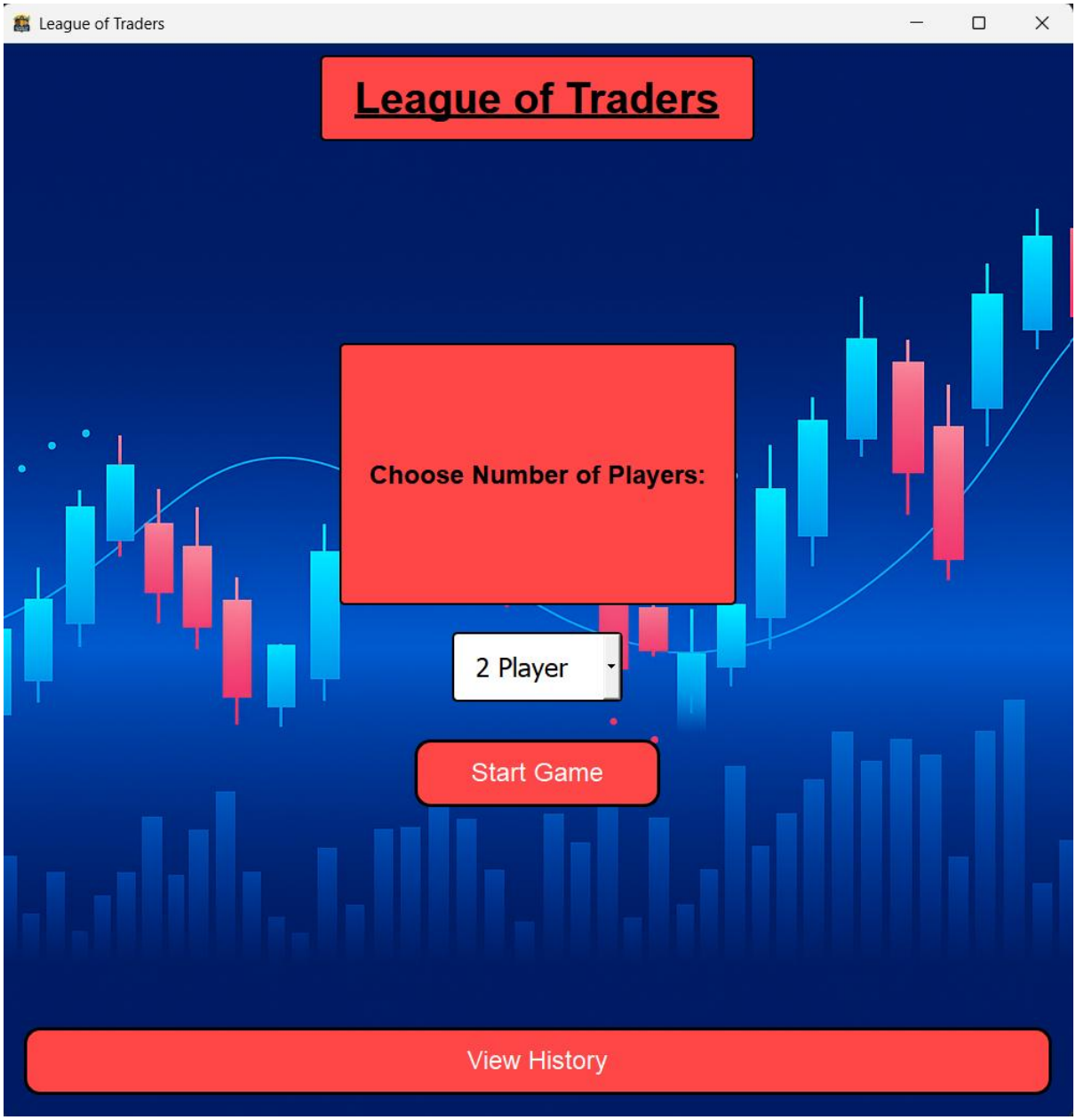
Then for my title, I had quite a lot of things to edit to make it stand out more compared to other labels and buttons. One major thing I wanted to implement was increasing the font size to 40px, which I did. This it to make it bigger than other labels. Furthermore, I underlined the code to make sure it would further stand out. With all this added, the code now would look like this:

Iterative Development



After looking over how the colours contrasted with each individual box and label, I came to the decision to change my labels boxes to be the same colour as the push buttons, after doing this the menu screen now looked like this:

Iterative Development



In my opinion, I believe this was an amazing idea as I feel it is even easier now to read the text on my screen compared to how it looked last time, even though it was still somewhat readable.

However, after looking at my label, I thought they were still pretty lacklustre, and looked dull compared to what you see in professional games. This was also evident as I compared the look of my work to my peers, and the design of their work was exceptional and more professional than mine. When I say professional, I mean in the sense that their work visually was comparable to that of publicised games. This is where io first decided to first change the title of my game to look better:

https://www.fontspace.com/new/fonts

Iterative Development

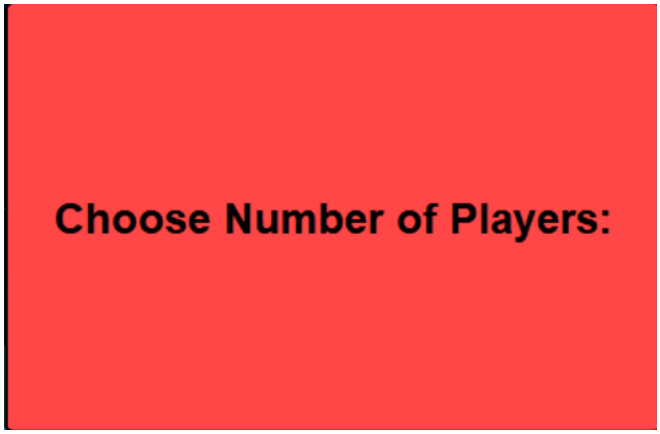
After further research on the website, I stumbled upon the perfect font, in my opinion, to use for my game:

https://www.fontspace.com/modern-cosmo-font-f144664

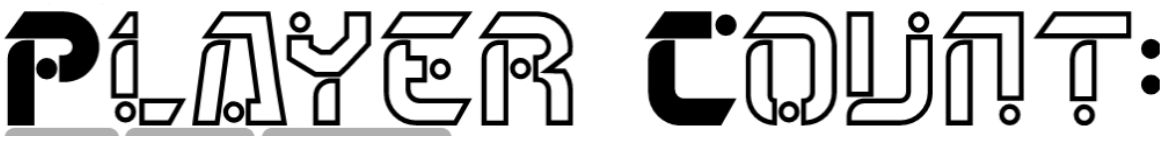
This is what the fonts looks like with my title:

What I like about this font is that I blends both a modern and dynamic look to my game, which is in my opinion also quite a modern game. This is because it involves blending both trading and competitive gaming into a educational and entertaining experience. This is perfectly captured by the font.

To extend on this, I am now also able to solve the issue of my other label (choose number of players) which box looked very big in comparison to the words inside it:



I can change this by now making my label into a png image like that of the other font:



I had to change the name as there were to many words with the last name, it would not be able to capture all the letters. Now with finding these to new labels, I am now going to show you the code I used to implement these 2 cool designs:

Iterative Development

After this, I now wanted to fix the scaling issue of my boxes. Once I tried to go full screen with my window, the window would be shown as this:

This is a major issue as I need to scale my labels, buttons and backgrounds to fit the actual screen. This is to make my game look a lot more professional and less clunky.

This how I fixed my issue:

Stage 1 review:

Each part of this stage felt like it was building into more things. What I mean by this is when I first started with creating my flowchart, I purely wanted to show the process of how people should navigate through my menu screen. This was simple and gives a solid outline, of the process people should take.

To build upon this I then made simple pseudocode, this gave the outline of what my code should look like without creating actual code. This was a way to show testers and people looking at my code, what are the main aspects this piece of code will do, without looking at possibly hundreds of lines of code that will get increasingly more confusing. Furthermore, it condenses large pieces of code into readable one line piece. For example, to create a window in PyQt5, you would need to do this:

```
import sys
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QApplication, QMainWindow



class MainWindow(QMainWindow):
  def __init__(self):
    super().__init__()
```

Iterative Development

```
        self.setGeometry(700, 300, 1000, 1000)


def main():
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())


if __name__ == '__main__':
    main()
```

However, what I did in pseudocode is tis:

CREATE window screen

This is much more readable and gives a clear idea on what I will be doing, without producing 10+ lines of pseudocode that will look very similar to my actual code. This makes it obvious that I have not reverse engineered my code and shows I'm doing my stages in a clear step by step format.

Then I have my code which is the main part of each stage. This takes my ideas form my flowchart and pseudocode and outputs them into

My aims for this stage where:

1) Create a menu screen
2) Crete widgets such as buttons and labels
3) Let you have a previous history screen that can only be clicked if data is found
4) Have the widow data to be readable
5) Once clicking on start game, a new, blank, window will replace the existing one

Score – 4/10

Iterative Development

Comments: This stage is seriously underdeveloped. As I will be trying to complete more of it while I go to stage 2, this stage was more of a learning curve for me to overcome as this was the first real experience of using PyQt5 in a large product. So while I am very happy with the results I was able to output while getting over this curve and going on with this experience, I do believe this stage, I should have been able to do more.

Score – 10/10

Comments -

Stage 2: Basics

Stage 2 will be expanding on this work and going onto the next slide (GUI design framework) of my game. I will most likely during this stage, cover the next 2 slides of my game as they will be quick to do and they both are linked to each other until the slide 4 (stage 3) which will be a very large stage and take me the longest, other than implementing the game.

Iterative Development

NOTE:

.Talk about extra features added not in pseudocode

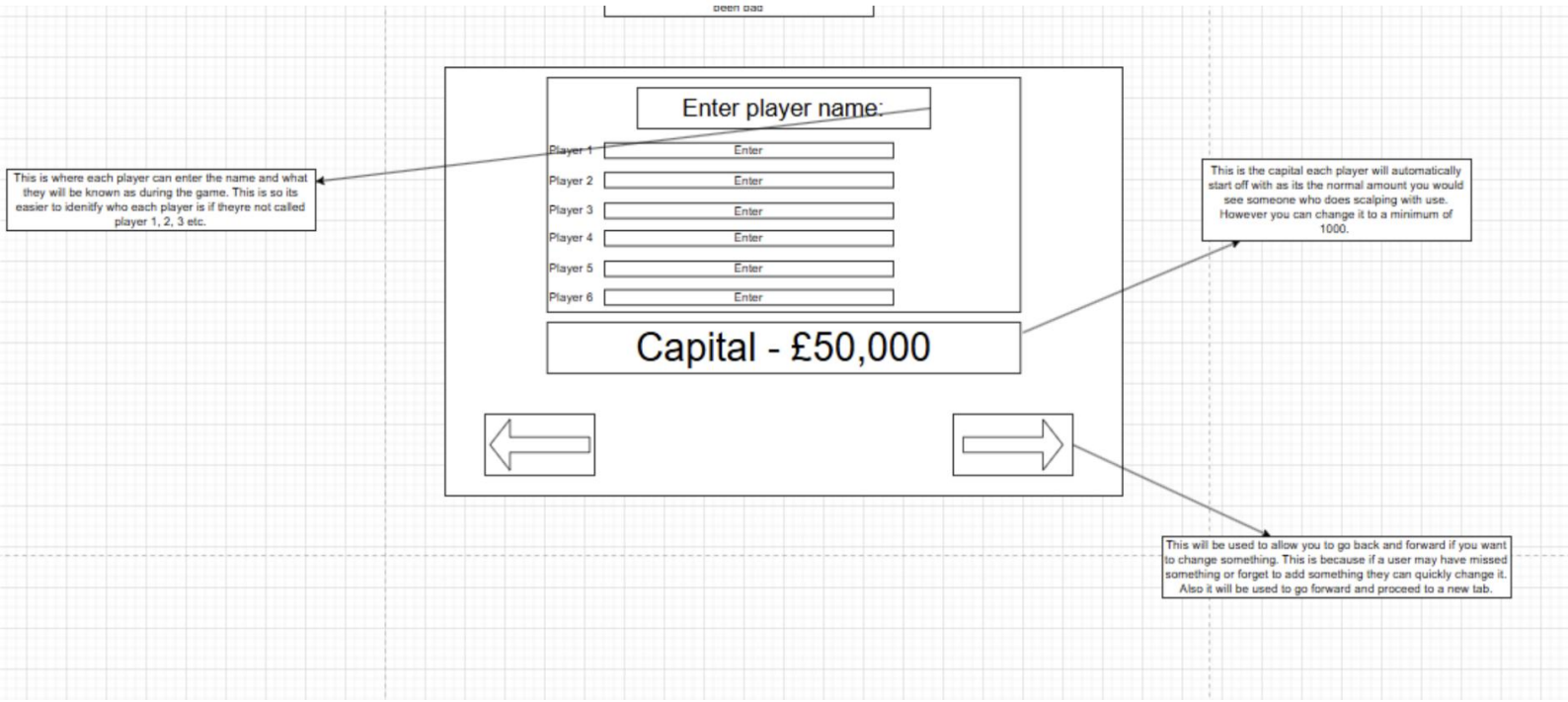
.Talk about features I have added in python and would like to implement into pyqt5 via labels, pushbuttons etc;

.Talk about features such as my test and how I was able to overcome not understanding how to do something

.Talk about each minute detail such as pushbuttons changing colours once hovering over them

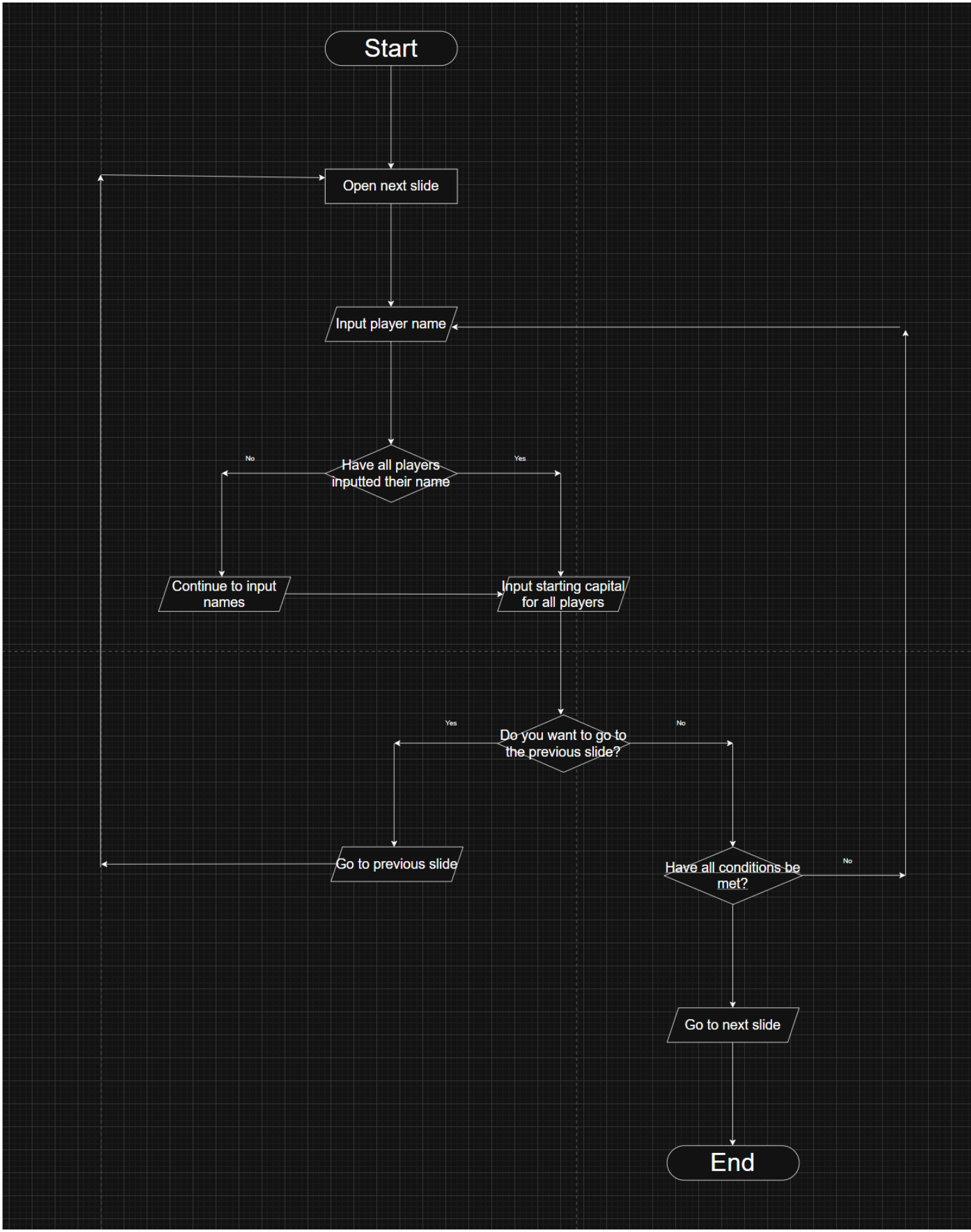
.Talk about how the test plan showed things that worked / did not work during coding the game

Stage 2:



Iterative Development

Flowchart:



Iterative Development

My flowchart for this stage is showcasing what my player's name section will look like. We first start with opinion this new 'slide' of my game as that will be the first thing ever player will have to do. Then, we will have each player input their name, it does not matter what order they do it as stage 3 will take care of these types of aspects. Next, we have a decision box asking whether they have all input their name. If they haven't, they will need to continue inputting names until they're done. If they have done/ fixed their mistakes, then they can go onto inputting their capital (How much each player will be starting with). This is very important as they will have to have a certain amount of capital to actual play may game. To make my game possible I forced players to have at least a minimum of 1000 as it will all my players to at the bare minimum, be able to trade. Furthermore, just in case if they want to go to the previous slide, they can make the decision to do so or not. I did this as I believe it may be important for latter stages of my game so I should integrate it early as well. If they don't want to, it will then check whether all conditions are met, if so they can go to the next slide, if not they have to complete the requirements.

Iterative Development

Field Name	Data Type	Data Format	Field Size	Description	Example
app	QApplication	Object instance	None	Initializes the start of your GUI (game can start now)	app = QApplication(sys.argv)
window	MainWindow	Object instance	None	The window that pops up in which the game is now starting	window = MainWindow()
centralWidget	QWidget	Object instance	None	This widget contains buttons, labels, inputs, etc.	centralWidget = QWidget()
layout	QVBoxLayout	Object instance	None	The layout for organizing the widgets vertically	layout = QVBoxLayout(centralWidget)
player_inputs	List[QLineEdit]	List of objects	6 items	Holds references to all 6 player input fields	self.player_inputs.append(entry)
capital_input	QSpinBox	Object instance	1	Input spinner for setting starting capital	self.capital_input = QSpinBox()
back_button	QPushButton	Object instance	1	Back button for navigating to the previous screen	back_button = QPushButton("← Back")
forward_button	QPushButton	Object instance	1	Start game button to initiate the game	forward_button = QPushButton("→ Start Game")
button_layout	QHBoxLayout	Object instance	None	Horizontal layout to align back and start buttons left and right	button_layout = QHBoxLayout()
go_back	Function	None	None	Function to go back to the previous screen (action for back button)	def go_back(self):
start_game	Function	None	None	Function to start the game (action for start button)	def start_game(self):
setGeometry	Method	Integer	4 parameters	Sets the position and size of the window	self.setGeometry(700, 300, 1000, 1000)
setWindowTitle	Method	String	None	Sets the window's title	self.setWindowTitle("League of Traders")
setWindowIcon	Method	QIcon	None	Sets the window's icon	self.setWindowIcon(QIcon("app.png"))
setStyleSheet	Method	String	None	Sets the stylesheet for the window and widgets	self.setStyleSheet("...")
showMaximized	Method	None	None	Maximizes the window when the program starts	self.showMaximized()

Iterative Development

Pseudocode

```
CREATE window screen  # Opens a blank PyQt5 window for player setup

SET windowTitle = "League of Traders - Player Setup"
SET windowIcon = "app.png"
SET windowBackground = "setup_background.png"

DISPLAY label "Enter player name:"  # Title label at the top of the window

# Create name entry boxes for up to 6 players
FOR i IN range(6):
   DISPLAY label "Player " + (i + 1)
   CREATE textBox "Enter"  # Lets each player enter their name
   STORE input in playerName[i]

# "This section allows each player to enter their name for easier identification in the game."
```

Iterative Development

```
SET startingCapital = 50000  # Default starting amount

DISPLAY label "Capital - £" + startingCapital


# "This is the default starting capital for each player. It can be changed to a minimum of £1000."


CREATE inputField "Change Capital"  # Optional field to edit capital

CREATE button "←"  # Back navigation

CREATE button "→"  # Forward navigation


# Tooltip or note

# "The arrows let you go back to a previous screen or forward to confirm entries."


# Logic for changing capital

IF inputField value >= 1000 THEN

   UPDATE startingCapital = inputField value

   UPDATE label to "Capital - £" + startingCapital

ELSE

   DISPLAY message "Minimum capital is £1000"


# Navigation logic

IF Back button CLICKED THEN

   GO TO previousScreen


IF Forward button CLICKED THEN

   FOR i IN range(6):

      IF playerName[i] is EMPTY THEN

         SET playerName[i] = "Player " + (i + 1)
```

Iterative Development

SAVE playerName[] and startingCapital

GO TO nextScreen

Code:

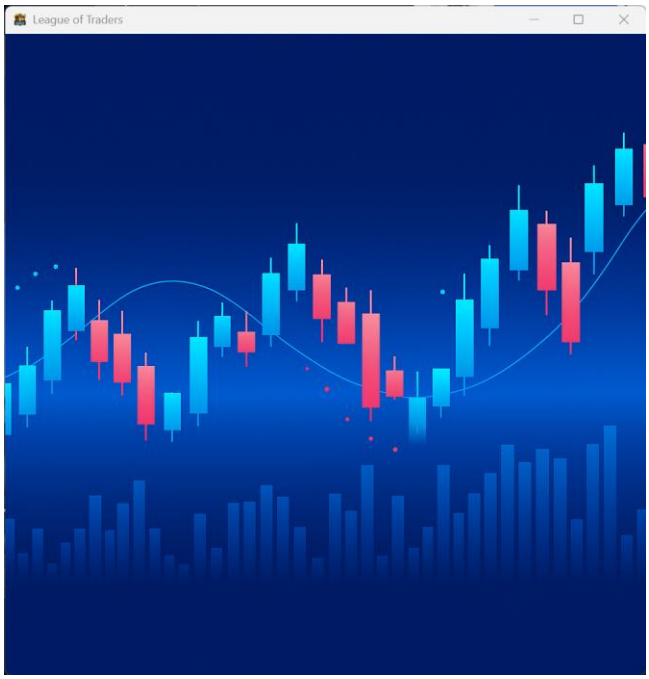
```
import sys  # Needed to access system parameters                                                    ⚠ 5
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QLabel, QLineEdit, QSpinBox, QPushButton  # PyQt5 widgets
from PyQt5.QtGui import QFont, QIcon  # Fonts and window icon
from PyQt5.QtCore import Qt  # Alignment constants
import time  # For countdown delays
```

I first inputted all the built in functions I would need to allow parts of my code to work. This is very important because without it, you are unable to actually use pyqt5 to its full capabilities.

Iterative Development

```python
class MainWindow(QMainWindow):  1 usage
    def __init__(self):
        super().__init__()
        self.setGeometry(700, 300, 1000, 1000)  # Window size and position
        self.setWindowTitle("League of Traders")  # Top bar title
        self.setWindowIcon(QIcon("app.png"))  # Icon on the top bar

        # Background for the window
        self.setStyleSheet("""
            QMainWindow {
                background-image: url("Trading.png");
                background-repeat: no-repeat;
                background-position: center;
                background-size: cover;
            }
        """)

def main():  1 usage
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

I then added the other key features of my window, such as the background, icon and title:



Iterative Development

```
centralWidget = QWidget()  # Container for widgets
self.setCentralWidget(centralWidget)  # Set as main area

layout = QVBoxLayout(centralWidget)  # Stack widgets vertically
layout.setSpacing(20)  # Space between widgets

centralWidget.setFont(QFont("Arial", 16, QFont.Bold))  # Font for all widgets
centralWidget.setStyleSheet("""
      QPushButton, QLineEdit, QLabel, QSpinBox {
          font-size: 20px;
          border: 2px solid black;
          border-radius: 5px;
          padding: 10px 15px;
      }
  """)
```

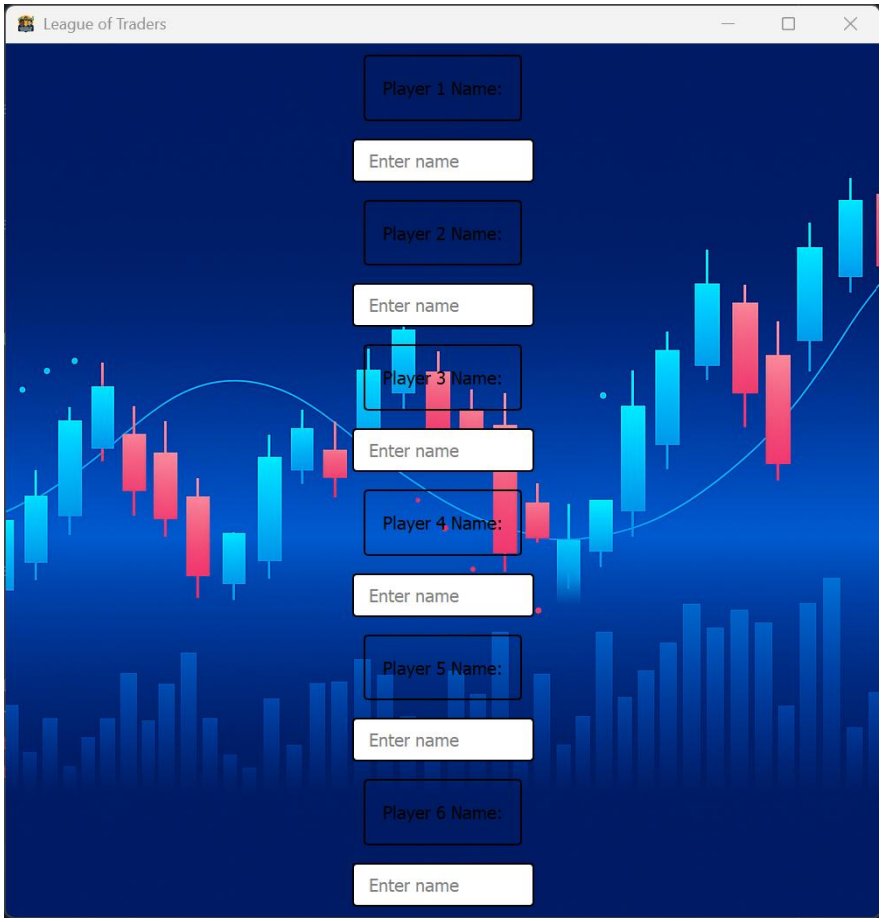
I then added my central widget function to create te outline of my widgets. I did this again as it allows me to have a foundation for what each widget will be like, and if I want to change it, I then can change it under that specific widgets code.

```
self.player_inputs = []  # Store references to inputs

for i in range(6):
    label = QLabel("Player " + str(i + 1) + " Name:")  # Player label
    entry = QLineEdit()  # Input field
    entry.setPlaceholderText("Enter name")  # Placeholder text
    layout.addWidget(label, alignment=Qt.AlignHCenter)  # Center label
    layout.addWidget(entry, alignment=Qt.AlignHCenter)  # Center input
    self.player_inputs.append(entry)  # Save input for later
```

I then created a self.player_input list, like an array, to store all my inputs for QlineEdit. Using a list will allow me to effortlessly access each player's name later, once the actual game starts. Furthermore, I then added a loop to create six players. During this stage im currently just trying to add the core features of each slide. In the last stage I will then tie each slide together, hence it will only show a certain amount of player inputs depending on how many players you decided to host for the game during what you clicked in the menu slide. Then, I added QLabel which is a widget showing player 1 from 6. To make my slide look better, Ihade the alignment set to centre as, in my opinion, looks cool.

I structured it this way because it gives me a solid foundation to expand on, in later stages, to tie everything together in an amazing, playable game.

Iterative Development




```
capital_label = QLabel("Starting Capital (£):")
self.capital_input = QSpinBox()
self.capital_input.setRange( min: 1000,  max: 50000)  # Allowed range
self.capital_input.setValue(25000)  # Default value
layout.addWidget(capital_label, alignment=Qt.AlignHCenter)
layout.addWidget(self.capital_input, alignment=Qt.AlignHCenter)
```

Next I then added to code to make my QSpinBox for my starting capital, this is because it allows the user to puick a specific number within the range I stated, 1000 is the minimum and 50000 is the maximum. Using this input ensure that the user cant enter an invalid text, such as string or negative numbers. I also set the starting value as the median as it seems like a reasonable number to put.

I added this after my player for loop as it makes sense for each player to first input their names then next decide the capital they should all hold.

Iterative Development

```
    button_layout = QHBoxLayout()  # Create horizontal layout for buttons
    back_button = QPushButton("← Back")  # Create back button
    forward_button = QPushButton("→ Start Game")  # Create start game button
    button_layout.addWidget(back_button, alignment=Qt.AlignLeft)  # Align back button left
    button_layout.addWidget(forward_button, alignment=Qt.AlignRight)  # Align start button right

    layout.addStretch()  # Push all widgets above upwards
    layout.addLayout(button_layout)  # Add button layout at bottom of window

    back_button.clicked.connect(self.go_back)  # Connect back button to go_back function
    forward_button.clicked.connect(self.start_game)  # Connect start button to start_game function

def go_back(self):  1 usage
    print("Going back...")  # Placeholder action for back button

def start_game(self):  1 usage
    print("Starting game...")  # Placeholder action for start button
```

Next I wanted to integrate me previous slide and next slide buttons which act as a way to transition to previous and next slide. First I wanted to instead of integrating vertical buttons, integrate horizontal, as it will allow me to place buttons in a more viable spot, like bottom left and bottom right. Next I created wqhat the buttons would look like. Then I made these push buttons have a function, once you click theback button it goes back and once you click the forward button it starts game. However, the functions has not been integrate into pyqt5 gui but python so it only says go back and go forward. In my last stage I will be integrating this feature.

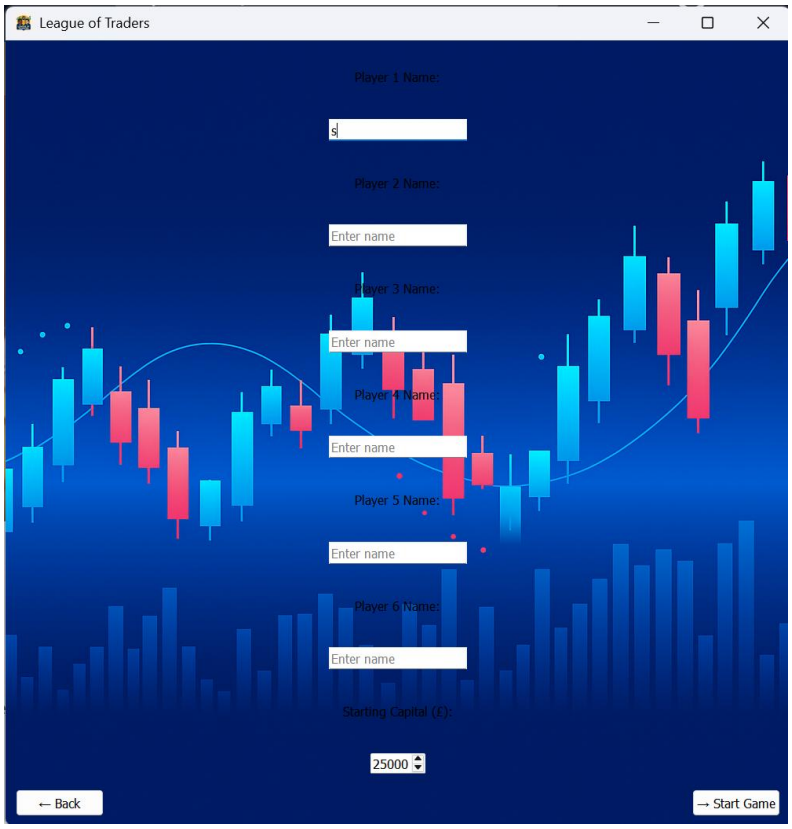
Iterative Development

An issue I have now is that my background is not showing. While I didn't really understand how to, I triend many different methods via trial and error. However what worked was me removing the comment out of my style sheet:

```
self.setStyleSheet("""  # Add background image to main window
    QMainWindow {
        background-image: url("Trading.png");
        background-repeat: no-repeat;
        background-position: center;
        background-size: cover;
    }
""")
```

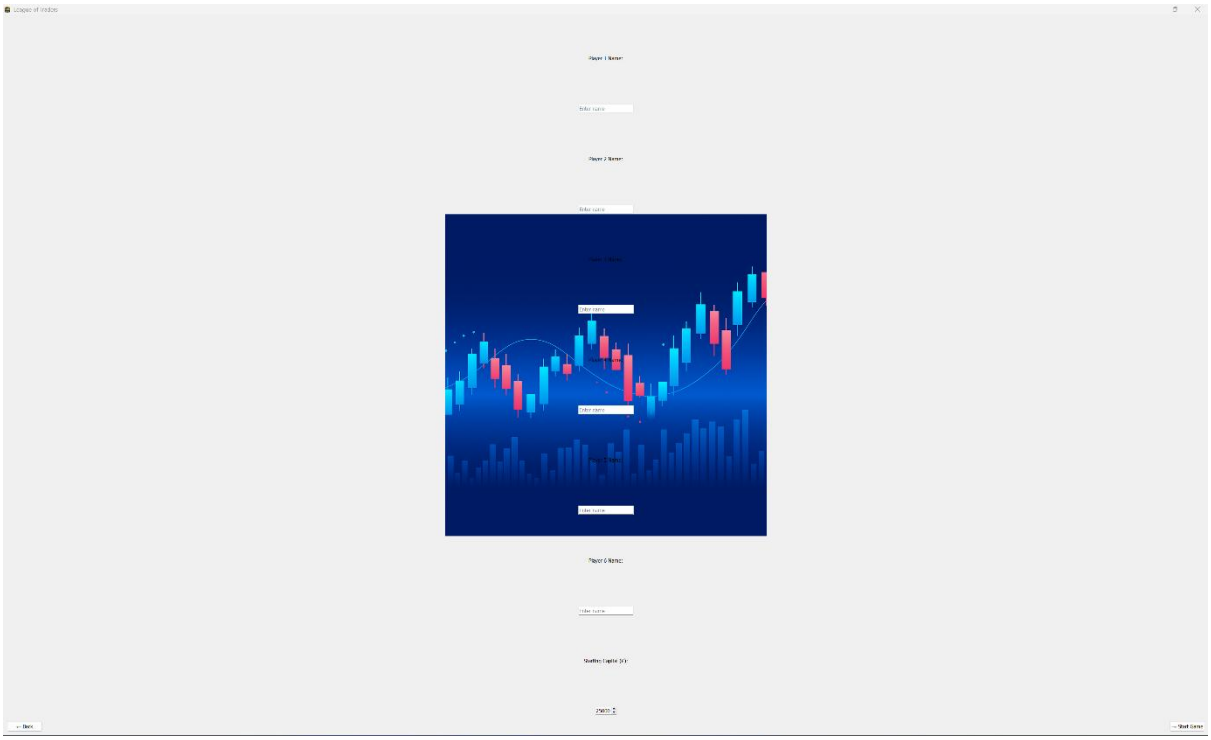
Iterative Development

By removing the # comment, it managed to fix my background. While I don't understand why it fixed it.




```
self.showMaximized()
```

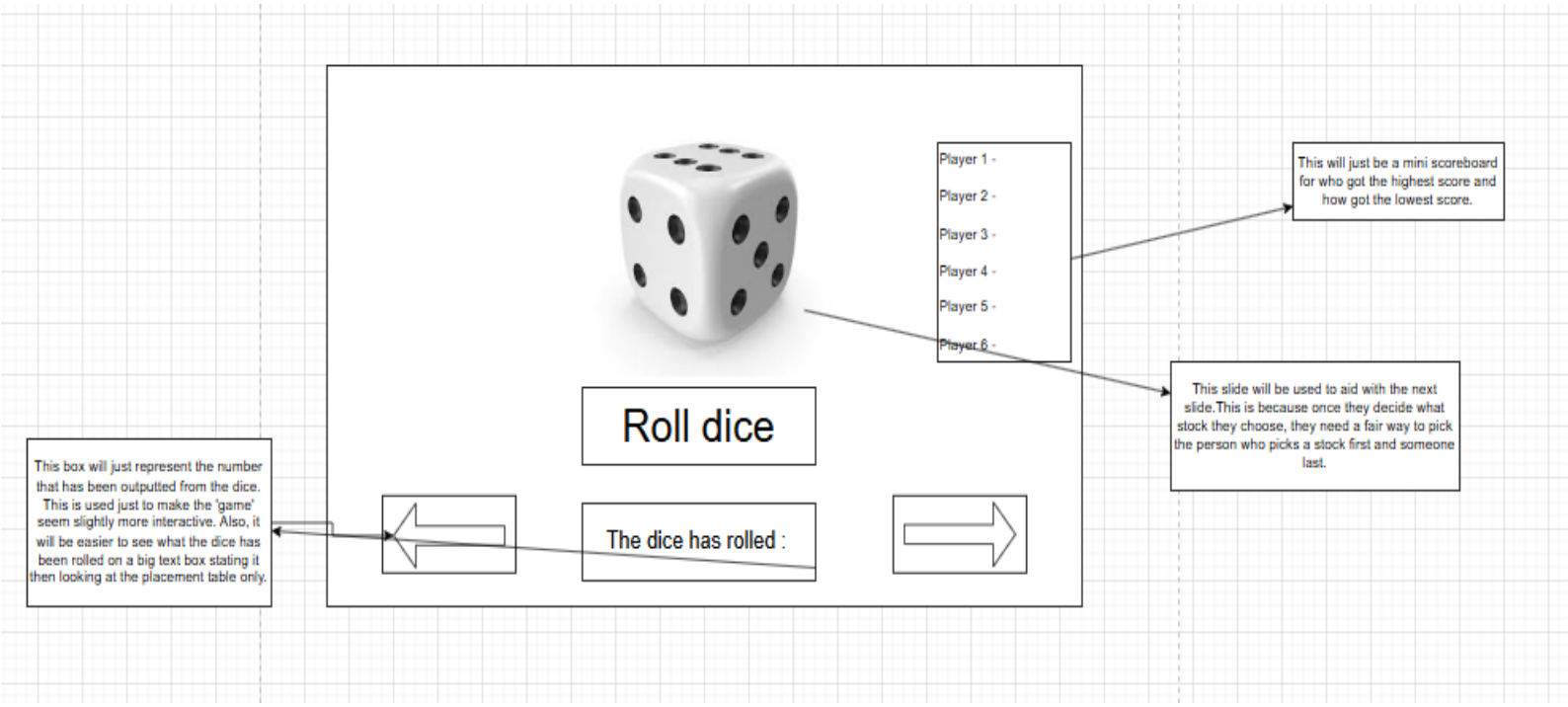
Finally so my screen will be maximised once I run the program, I added this piece of code to maximise my tab instantly.



Iterative Development

During my final stage I also will be fixing this issue of my background and widgets not scaling to fit for how large the screen is. This will be done in the final stage as I consider things like these, quality of life, so they improve user experience.

Stage 3:



Iterative Development

Flowchart: