Final Project

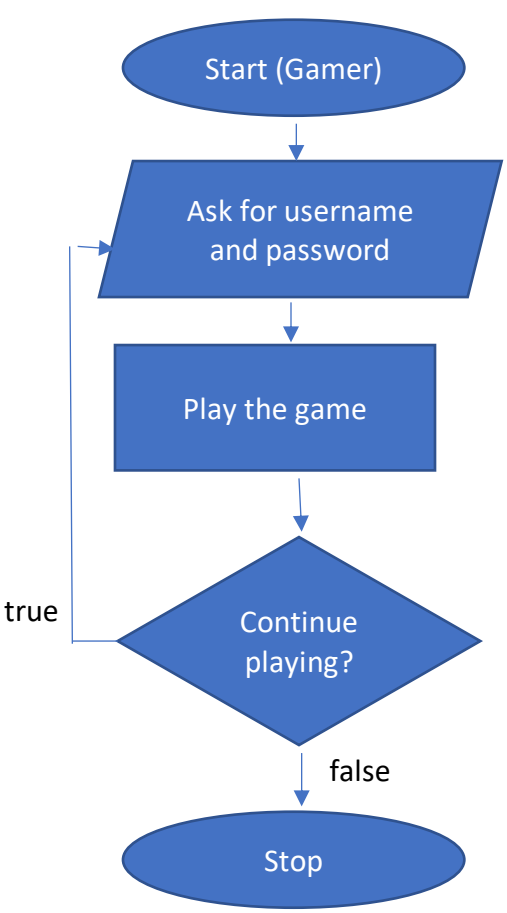
Objective 1:
Gamer:

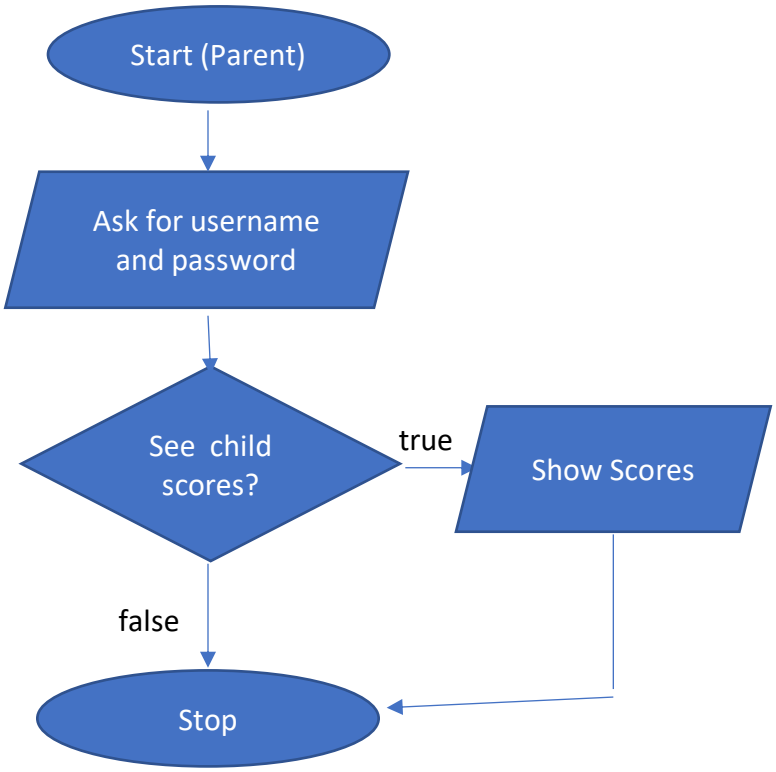
1. User –
    a. The program must allow the user to enter their username and password.
    b. The first screen should allow the user to either create a new account, or log into an existing account.
    c. The user must create an account to play the game.
    d. The account must be created by the parent.
    e. These requirements belong in the User category because it has to do with how the user can play the game and how to set up the game.
2. Functional –
    a. When the program first starts, there will be no users accounts created at all.
    b. The parent must create the gamer's account to start playing.
    c. Gamers can then play the game and see their high scores Parent accounts can create a child account and change the time limit to how long their child can have to solve an anagram.
    d. These requirements belong in the Functional category because it describes how the program functions and the sequence of actions.
3. Non-Functional – There are no non-functional requirements because there are no constraints to how long a gamer can play the game.
4. System –
    a. The program can only run on laptops, and not mobile phones.
    b. It requires a file called AnagramData.txt to be in the same folder as the source code. This allows the program to read the words in the text file correctly.
    c. These requirements belong in the System category because they describe where you can play this game.
5. Domain – There are no domain requirements because this game can be played by anybody, anywhere, at any time.
6. Adding Value – If the gamer knows many vocabulary words, this will be a fun game. If not, put easier words into AnagramData.txt and try again.

Parent:

1. User –
    a. The program must allow the user to enter their username and password.
    b. The first screen should allow the user to either create a new account, or log into an existing account.
    c. The parent must create a child account after creating their own account.
    d. These requirements belong in the User category because it has to do with how the user can set up the game for the gamer.
2. Functional –
    a. When the program first starts, there will be no users accounts created at all.

   b. User can create a child account and change the time limit to how long their child can have to solve an anagram.
   c. These requirements belong in the Functional Category because it describes how the program functions.
3. Non-Functional – There are no non-functional requirements because there are no ways to constraint how long a gamer can play the game.
4. System –
   a. The program can only run on laptops, and not mobile phones.
   b. It also requires a file called AnagramData.txt to be in the same folder as the source code. This allows the program to read the words in the text file correctly.
   c. These requirements belong in the System category because they describe where you can run this game.
5. Domain – There are no domain requirements because this game can be played by anybody, anywhere, at any time.
6. Adding Value – If the gamer knows many vocabulary words, this will be a fun game. If not, put easier words into AnagramData.txt and try again.





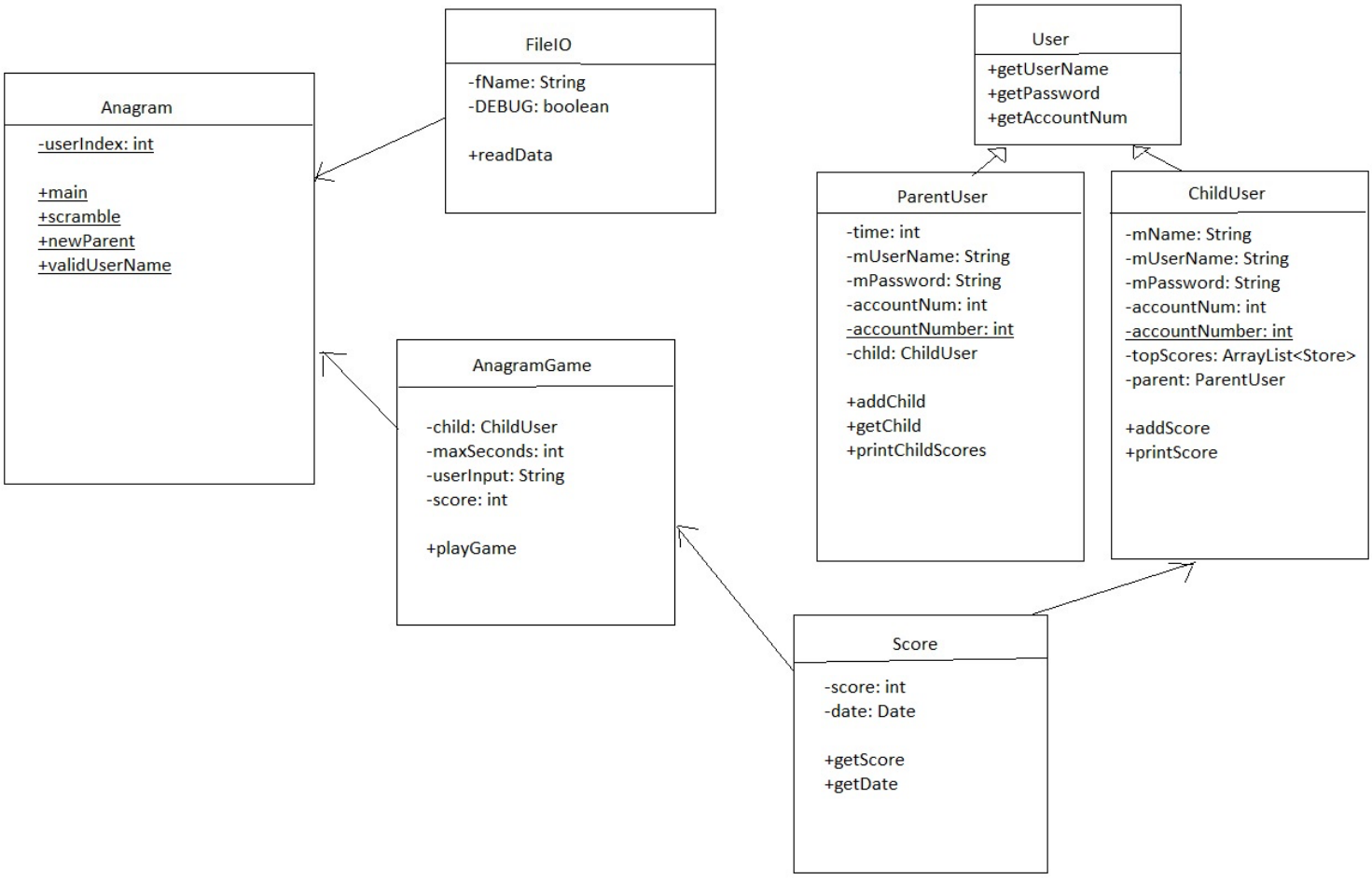
This is the gamer flow chart diagram. It shows how the gamer will be asked to enter their username and password in the beginning of the program, then it will play the game if their username and password is correct. When they finish playing the game, they will be given a choice to play again or to stop. If the gamer chooses to continue, then they will need to log in again and play the game.

This is the parent flow char diagram. It shows how the parent will be asked to enter their username and password. If the information is correct, they can view their child's scores. If they choose to show the scores, then it will finish the process and stop the game. If they do not, then the game will stop.

Objective 2:

1. First list of methods and classes: Anagram (main, scramble, newParent), ParentUser(addChild, printChildScores), ChildUser(addScore), Score(getScore), FileIO(readData)
2. Final list of methods and classes: Anagram(main, scramble, newParent), AnagramGame(playGame), ChildUser(addScore, printScore), ParentUser(addChild, printChildScores), Score(getScore, getDate), FileIO(printData), User(interface)

Objective 3:
UML Diagram:



Objective 4:

They are in the github link under the folder doc

Objective 5:
The code is in the github link. AnagramText.txt is also already in the correct place for relative path.