Criterion B: Design

Activity A (File for scanning)

Input data

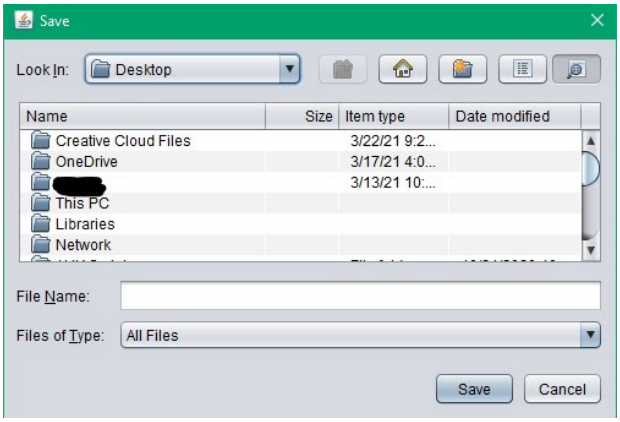
Data format: Timestamp,Email Address,Date Attending,Last Name,First Name,Sport,Q1,Q2,Q3 (Q1, Q2, and Q3 are Yes/No answers)

Q1: Within the last 14 days, have you been diagnosed with COVID-19 by a medical professional or had a test confirming you have the virus?
Q2: In the last three (3) days, have you had or developed one or more of these symptoms: fever of 100˚F or greater, fatigue, body aches, chills, night sweats, cough, congestion, runny nose, shortness of breath, sore throat, headache, nausea or vomiting, diarrhea, a new loss of taste or smell
Q3: Have you been in close contact (within six (6) feet for 15 or more minutes) in the last 14 days with a confirmed positive COVID-19 person?

Example: 2/10/2021 16:10:55,gmail@gmail.com,2/10/2023,Doe,Jane,Swim,No,No,No
(the first entry in the data set is disregarded in the file searching code because it is irrelevant)

Location: .csv file stored on clients computer (appendix 1)

Activity B (Client's input)



Input/output data (File)

File input: User finds the file they want to scan via the file explorer utility and selects the one they want
File output: Tells the user they have selected the correct file format or if they have not selected the correct file format

Input data (Searching)

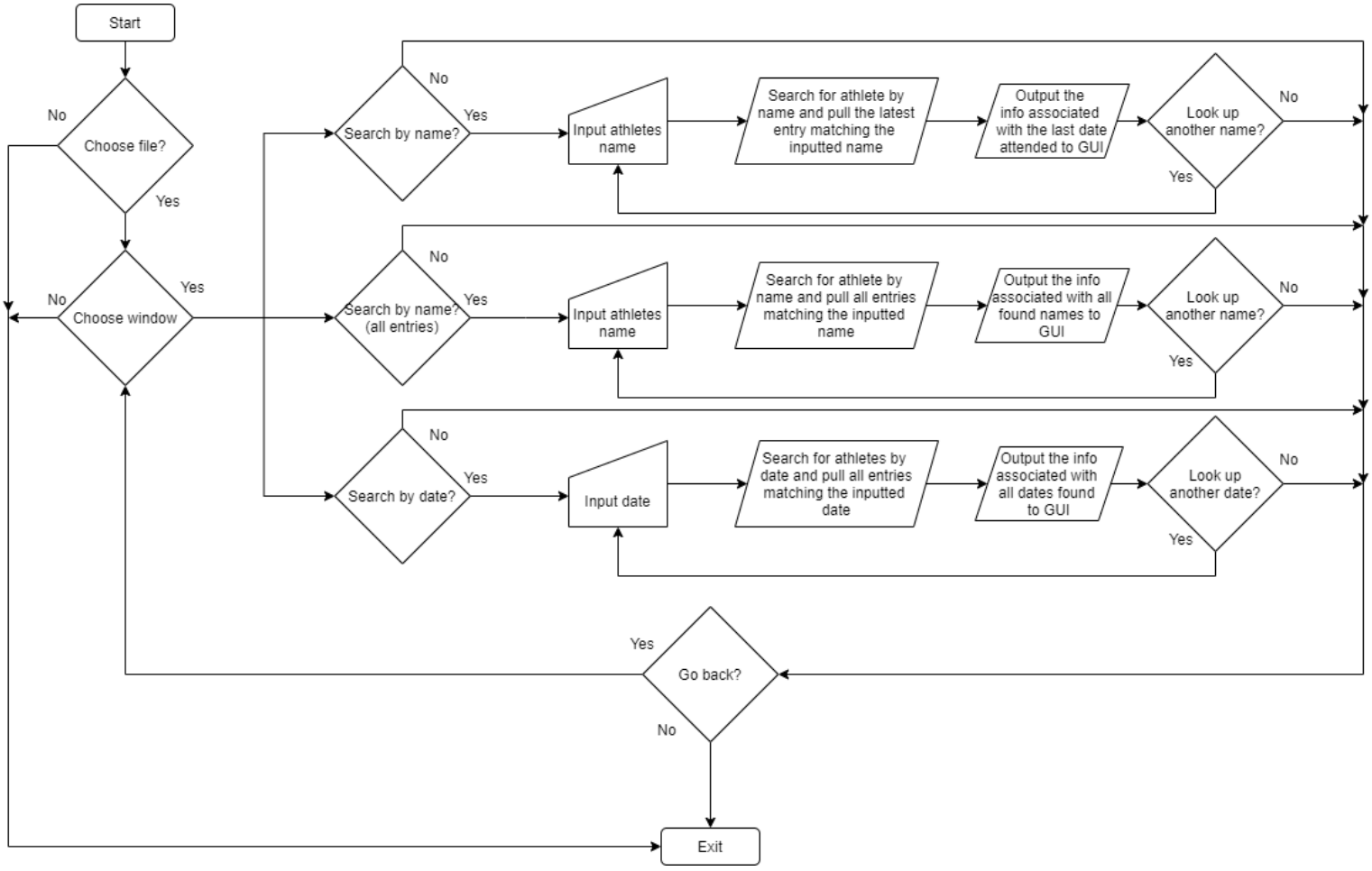
Search by name input:	**Search by name list input:**	**Search by date input:**

First Last or Last Example: Jane Doe or Doe	First Last or Last Example: Jane Doe or Doe	MM/dd/yyyy Example: 1/10/2021 or 01/10/2021

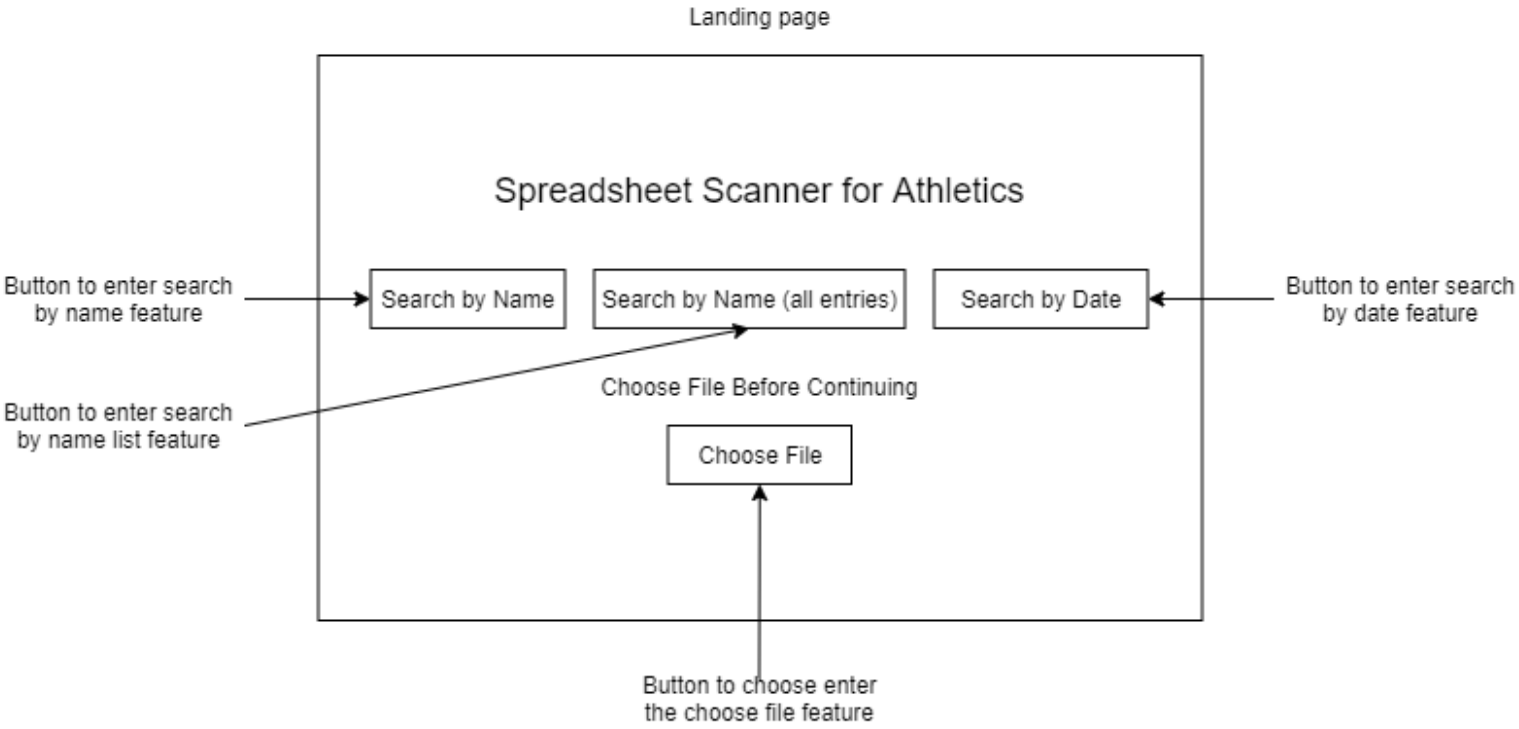
Output data (Searching)

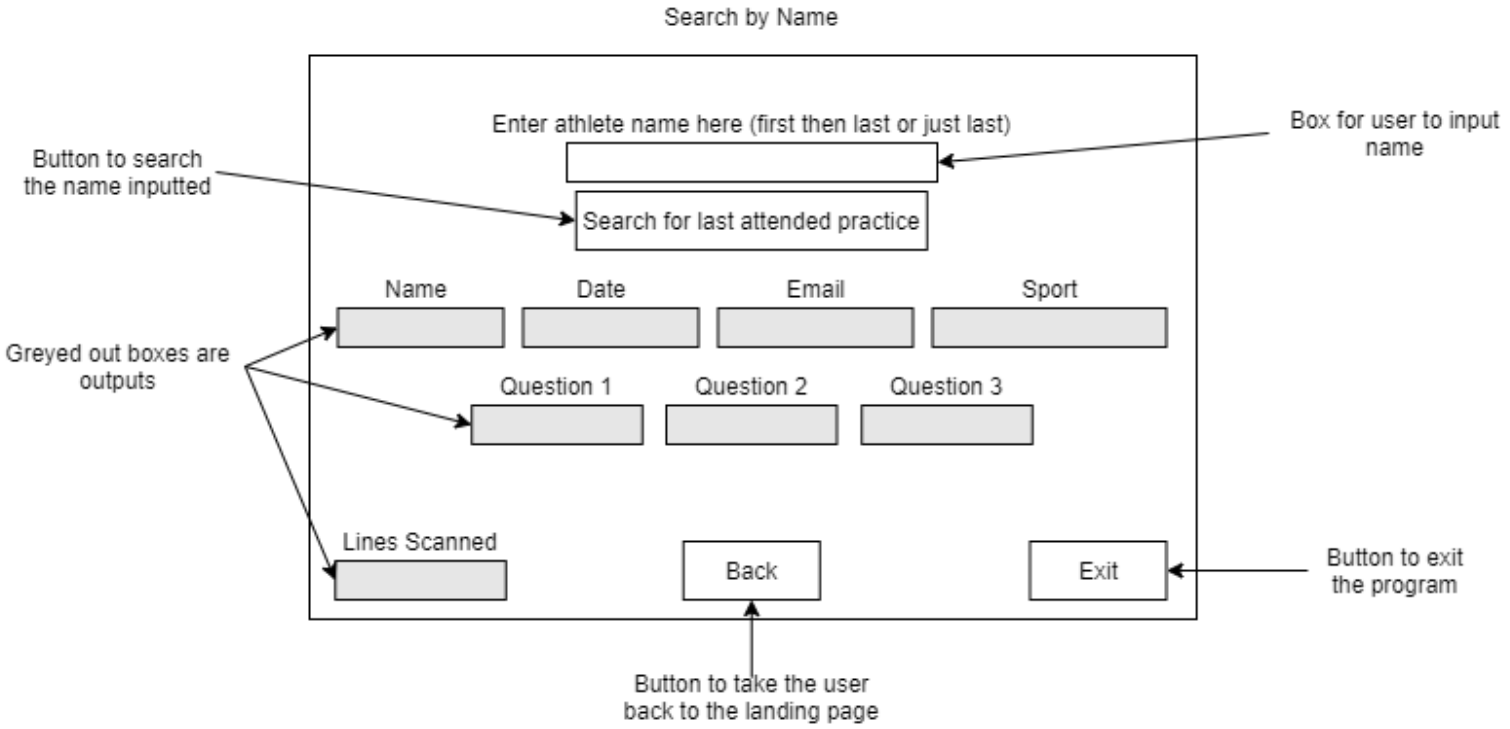
Search by name output: One athlete in the data format as shown in the example file	**Search by name list output:** Every entry that has a date matching the inputted one, outputted in the format shown in the example file	**Search by date output:** Every entry that has a date matching the inputted one, outputted in the format shown in the example file

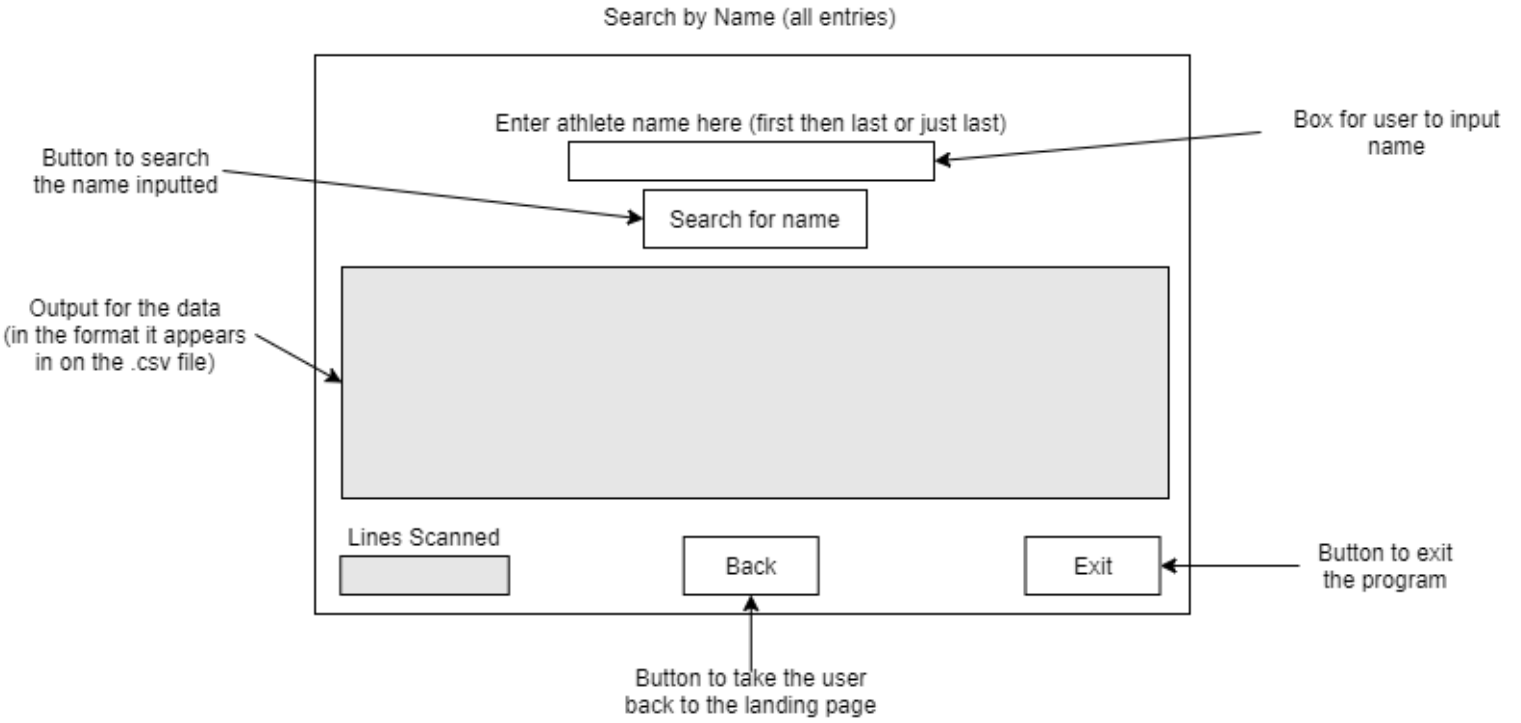
Process description

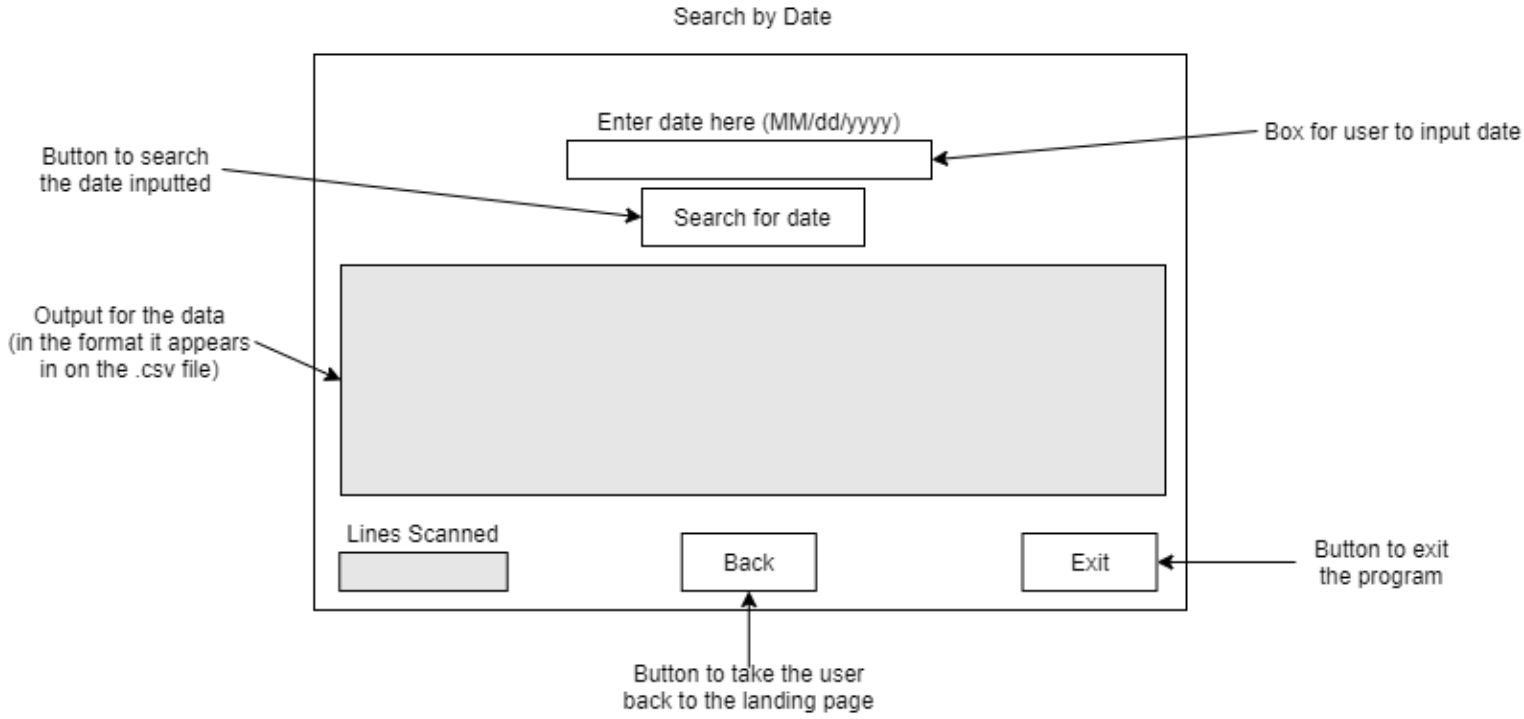


Graphical user interface storyboard (Updated to final product)






Search by Name (all entries)
Enter athlete name here (first then last or just last)
Box for user to input name
Button to search the name inputted
Search for name
Output for the data (in the format it appears in on the .csv file)
Lines Scanned
Back
Exit
Button to exit the program
Button to take the user back to the landing page


Search by Date
Enter date here (MM/dd/yyyy)
Box for user to input date
Button to search the date inputted
Search for date
Output for the data (in the format it appears in on the .csv file)
Lines Scanned
Back
Exit
Button to exit the program
Button to take the user back to the landing page

Risk mitigation

I mitigated my risks by developing the code in the building block model I did. This allowed me to implement a feature, immediately test it to be sure it works, then move on to the next one. A big reason this helped so much is if an error occurred when adding a new feature, I knew that it would be the new feature that was causing the error and not past code. I also developed the code in a way that did the most basic functions up to the more complex ones, which normally would be a bad thing, but in this case I needed to do it this way to prevent multi level errors.

Client requested changes

After the prototype was delivered to the client he requested that some changes be made. The major ones were that he wanted a feature to search by name and get a list of all dates that athlete attended practice and he wanted a feature to search by date instead of name. In addition, he wanted to be able to search not just by first and last name, but also by only last name and to be able to search the date by "02/1/2021" as well as "2/1/2021." These changes were suggested in a google form I constructed for the client to fill out when I delivered him the prototype (appendix 4) as well as a meeting in person after this (appendix transcript 1).

Schedule for developing the program

I originally divided the program into four building blocks based on the code needed to create the search by name feature, but after delivering my client the program he requested that I create the other two features as well (search by name list and search by date). For this reason, I will be dividing my program into three features with four building blocks that are used in all of the three features. In total, I spent 7 + 5 + 9 + 7 = 28 days on the building blocks and 6 days on developing the extra two features. The way I constructed my building blocks for the first feature allowed for easy adaptation for what I needed for the second two features, allowing me to spend much less time developing the last 2 features than the first one.

1. Building blocks
    a. Athlete object (7 days)
        i. Write code for an athlete object
        ii. Construct array list that can store all of the necessary values and this athlete object in an arraylist while still being able to access it's methods
    b. File scanner (5 days)
        i. Write code to make a file scanner for the data my client needs scanned
        ii. Get the scanner to store the data in the athlete object
    c. Searching and sorting algorithm (9 days)
        i. Write code to scan a test document with multiple entries
        ii. Write code to pull the lines I need from it based on the input as well as store it in the athlete object via the file scanner. (appendix 2)

iii. Additionally, write code to sort this information in order of most recent to least recent date of attendance (appendix 3)

d. GUI (7 days)
   i. Write code for a functioning GUI that allows for the user to select the file, input the search value, and get an output of the most recent date an athlete attended practice according to the file they scanned
   ii. Create a file explorer to select the correct file to scan
   iii. Create a landing page for the GUI
   iv. Create a video quick start guide for delivery with the prototype to the client

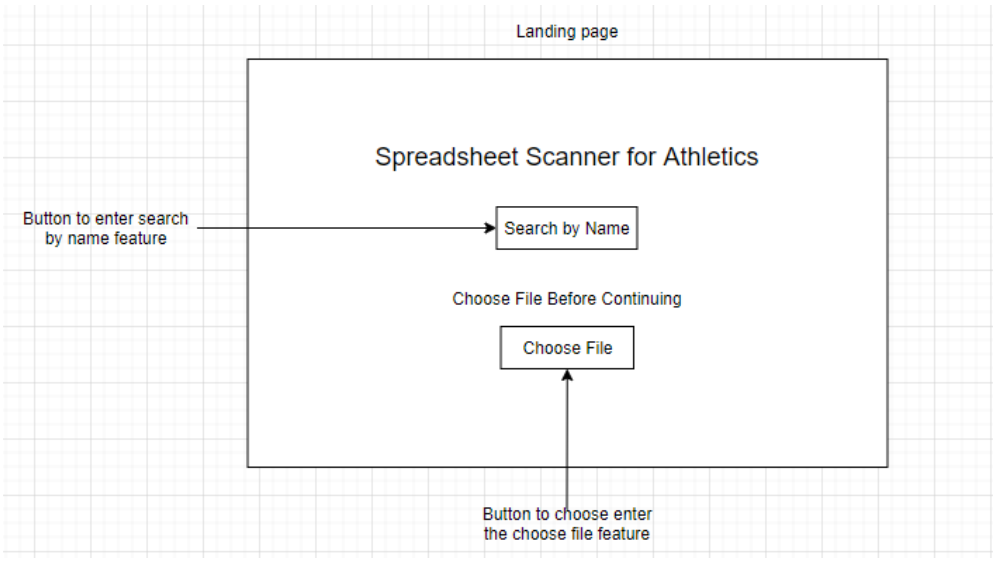
2. Search by name
   a. This feature was the original one and included all of the building blocks, but limited the searching and sorting algorithm to one output, just the most recent date the athlete attended practice
3. Search by name list
   a. This feature was essentially the same as the first feature, all I had to do was make program output the entire list of dates the athlete attended practice since it was already sorted from most recent to least recent
4. Search by date
   a. This feature was an adaptation of what I did for the search by name list except it did not include the sorting feature. What I did was change the search algorithm to search by date instead of name, which was a very simple change, and got rid of the sorting algorithm because since this was already searching by only one date. This meant that sorting by the date was completely unnecessary, so I just outputted the list of athletes as it appeared because my client did not need it sorted.

Testing plan

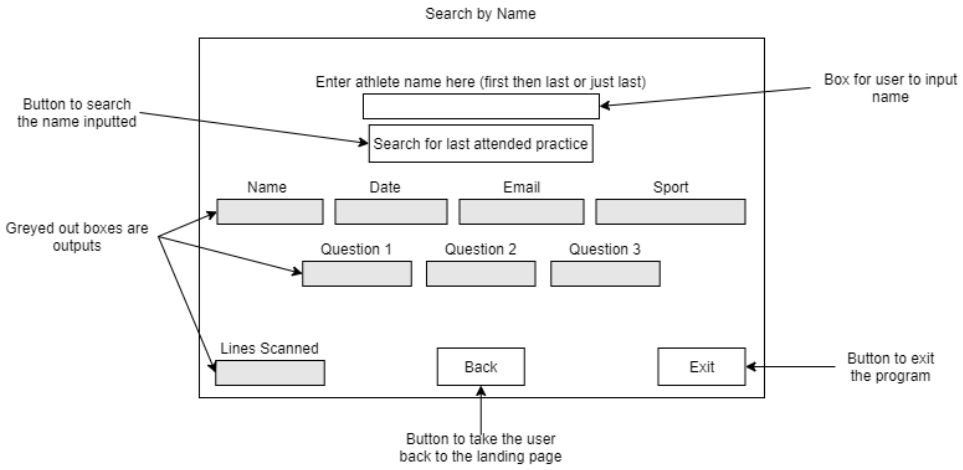
Examples of what the program should be able to do

Unit testing 1 (building block 1)	Object athlete should have a value and return method for: Email Address, Date Attending, Last Name, First Name, Sport, Q1, Q2, Q3 Example: athlete Jane Doe who plays women's soccer has values: janedoe@gmail.com, 1/1/2021, Doe, Jane, Women’s Soccer, No, No, No This athlete should be able to be stored in an arraylist athletes and all of it's values should be able to be called via the return method

	Example: athlete Jane Doe listed above is in arraylist athletes Athletes.add(Jane Doe) athletes(get(1).getSport()) Should return Women's Soccer
Unit testing 2 (building block 2)	File scanner should be able to take information from a sample file and store it in the athlete object Example: athlete Jane Doe above has her information stored on a .csv file fileScanner scan (.csv file) athletes(get(1).getDate()) Should return 1/1/2021
Unit testing 3 (building block 3)	The file scanner should be able to now search the .csv file for a specified name Example: the .csv file should look something like this janedoe@gmail.com, 1/1/2021, Doe, Jane, Women's Soccer, No, No, No johndoe@gmail.com, 1/2/2021, Doe, John, Boy's Soccer, No, Yes, No jimdoe@gmail.com, 1/10/2021, Doe, Jim, Boy's Swim,Yes, No, Yes And if I search for John Doe athletes(get(1).getQ1()) Should return Yes
Unit testing 4 (building block 4)	Graphical user interface should be able to house this function and give a place to input and output in a user friendly way The landing page should look like this:



The search by name feature GUI should look like this:



Action Test	Method of testing & results
Test if the athlete object works and is stored in an array list while still able to call methods	Set up a test athlete, store it in an arraylist, and call return methods from inside the arraylist to check if I get the correct values back
Test if the program is able to scan a .csv file and return the values from it	Set up a test .csv file and have the file scanner scan that file and return the values on it
Test if the program can search the file for a specified value in multiple lines of correctly formatted data and store it in the athlete object	Set up another test .csv file with correctly formatted data, store it inside the athlete object, and output it's values from inside the arraylist
Test if the program can sort those	Take the values that have previously been stored from

found values in the .csv file by date that have been stored in the athlete object	the .csv file and sort them in order of most recent to least recent and test the output to make sure that it is correctly sorted
Test if all of the GUI buttons function	Go through the GUI and perform every available task to check to make sure they all function as they should
Error handling	Go through the program and identify all sources of error, go back through the code and add handling for all of these