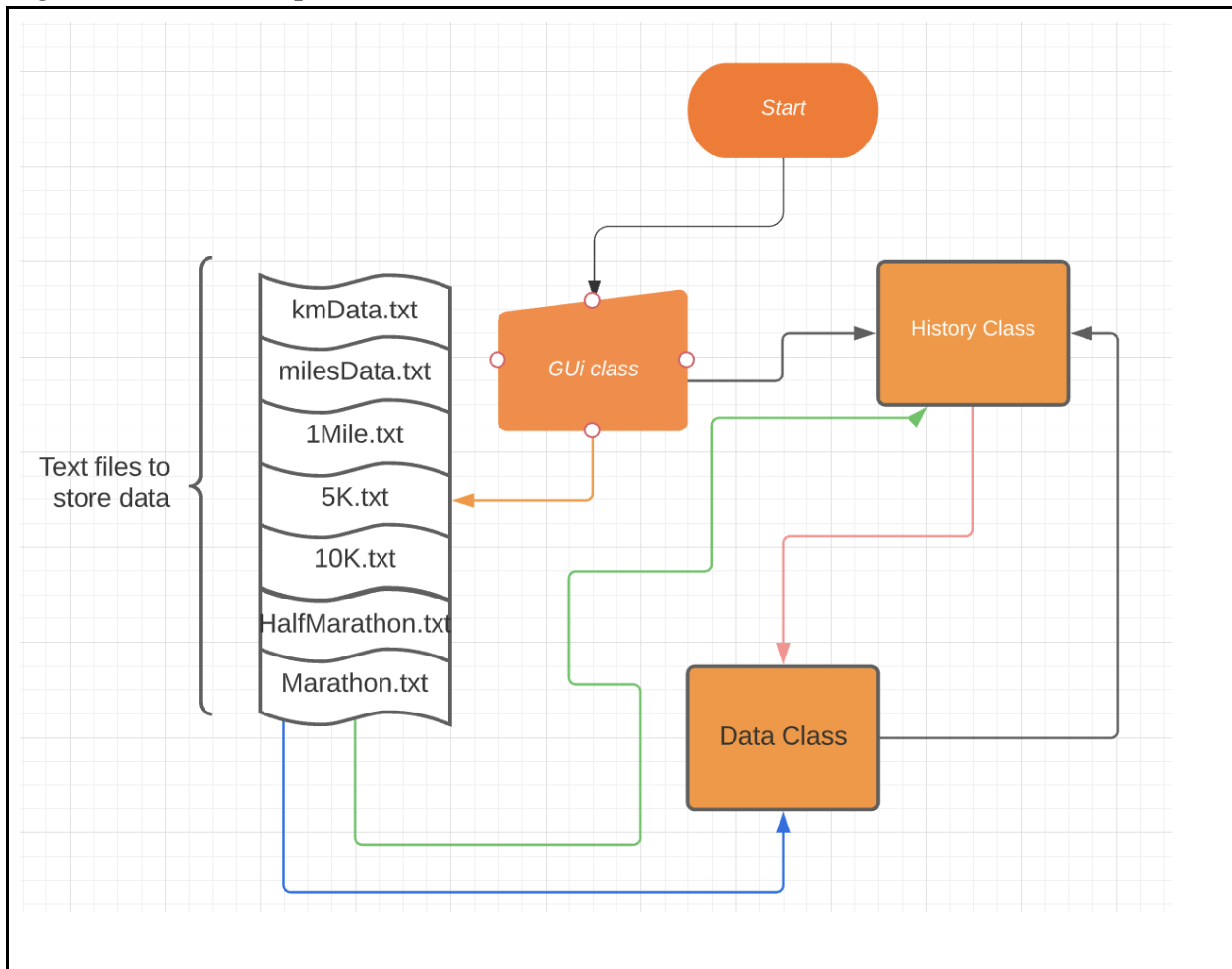# Criterion B: Design

Figure 1: Relationships between the classes and text files
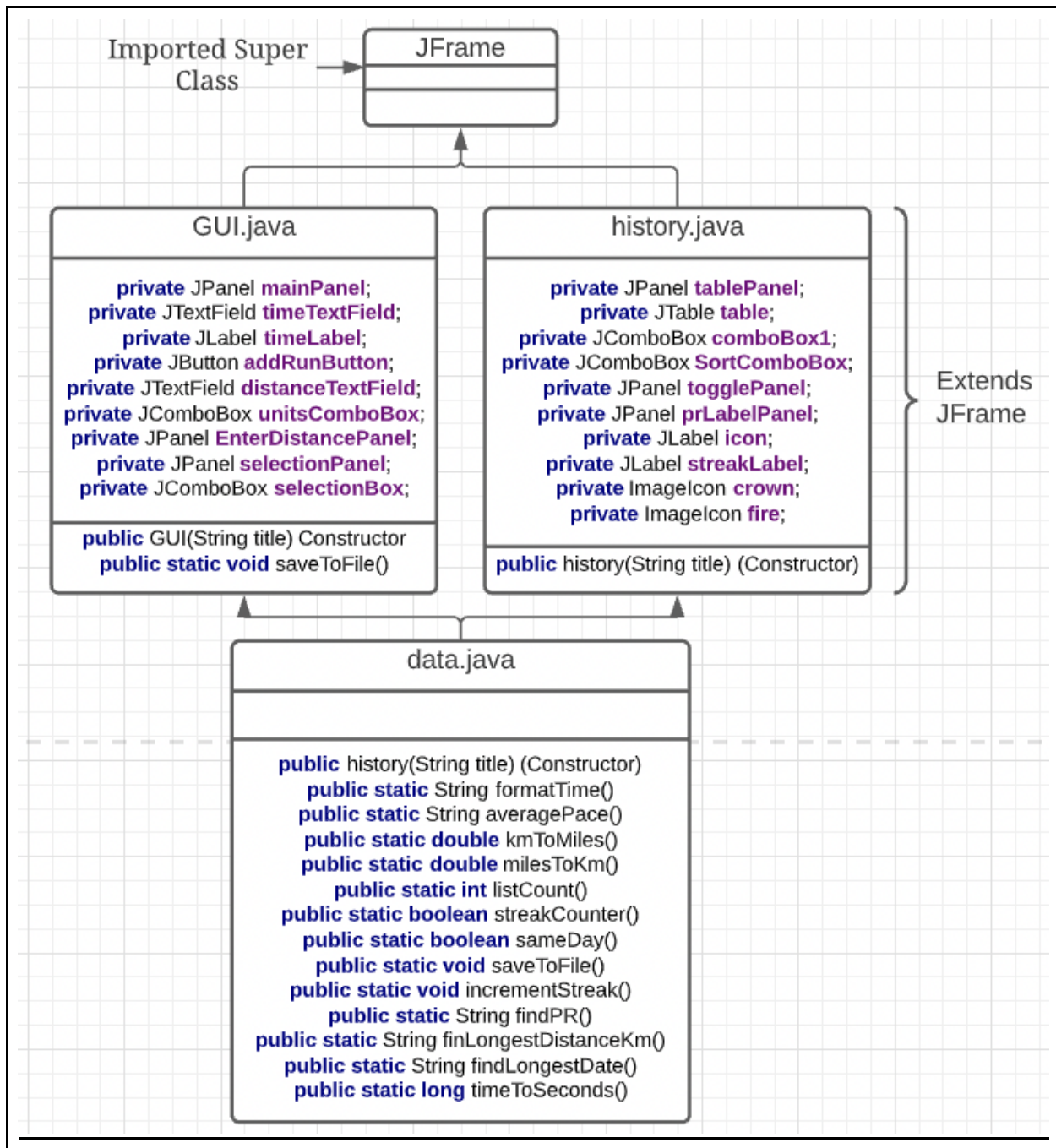

Functionality of each class:

- <u>GUI Class</u>: Creates the main UI for users to enter data. Passes inputted data into designated text files. Data is divided by "|" in the text files to sort different categories of each run.

```
1    2022/02/07 21:03:32|00:06:30|1.61|00:04:02
2    2022/02/07 21:03:49|00:18:30|5.0|00:03:42
3    2022/02/07 21:03:52|00:18:30|10.0|00:01:51
4    2022/02/07 21:04:02|01:18:30|42.195|00:01:52
```
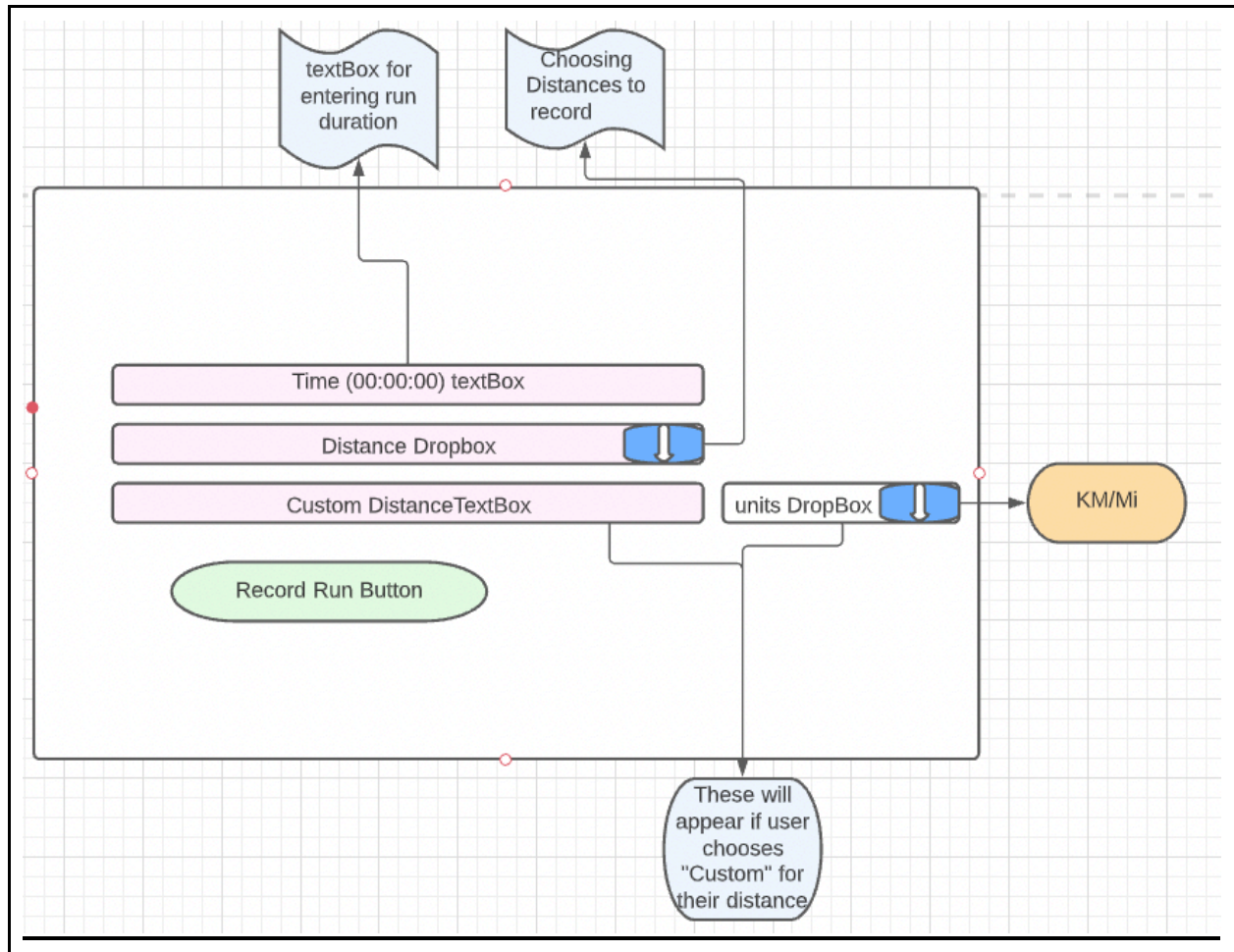
- <u>Text files</u>:
- <u>History Class</u>: Creates the Running Log UI and takes data from the text files to create a JTable displaying the run log. Calculations of streaks and PR are also displayed in this window.
- <u>Data Class</u>: Includes methods that are necessary for the functionality of the other two classes.

Figure 2 UML diagram for classes.

Imported Super Class → JFrame

GUI.java

private JPanel **mainPanel**;
private JTextField **timeTextField**;
private JLabel **timeLabel**;
private JButton **addRunButton**;
private JTextField **distanceTextField**;
private JComboBox **unitsComboBox**;
private JPanel **EnterDistancePanel**;
private JPanel **selectionPanel**;
private JComboBox **selectionBox**;

public GUI(String title) Constructor
public static void saveToFile()

history.java

private JPanel **tablePanel**;
private JTable **table**;
private JComboBox **comboBox1**;
private JComboBox **SortComboBox**;
private JPanel **togglePanel**;
private JPanel **prLabelPanel**;
private JLabel **icon**;
private JLabel **streakLabel**;
private ImageIcon **crown**;
private ImageIcon **fire**;

public history(String title) (Constructor)

Extends JFrame

data.java

public history(String title) (Constructor)
public static String formatTime()
public static String averagePace()
public static double kmToMiles()
public static double milesToKm()
public static int listCount()
public static boolean streakCounter()
public static boolean sameDay()
public static void saveToFile()
public static void incrementStreak()
public static String findPR()
public static String finLongestDistanceKm()
public static String findLongestDate()
public static long timeToSeconds()
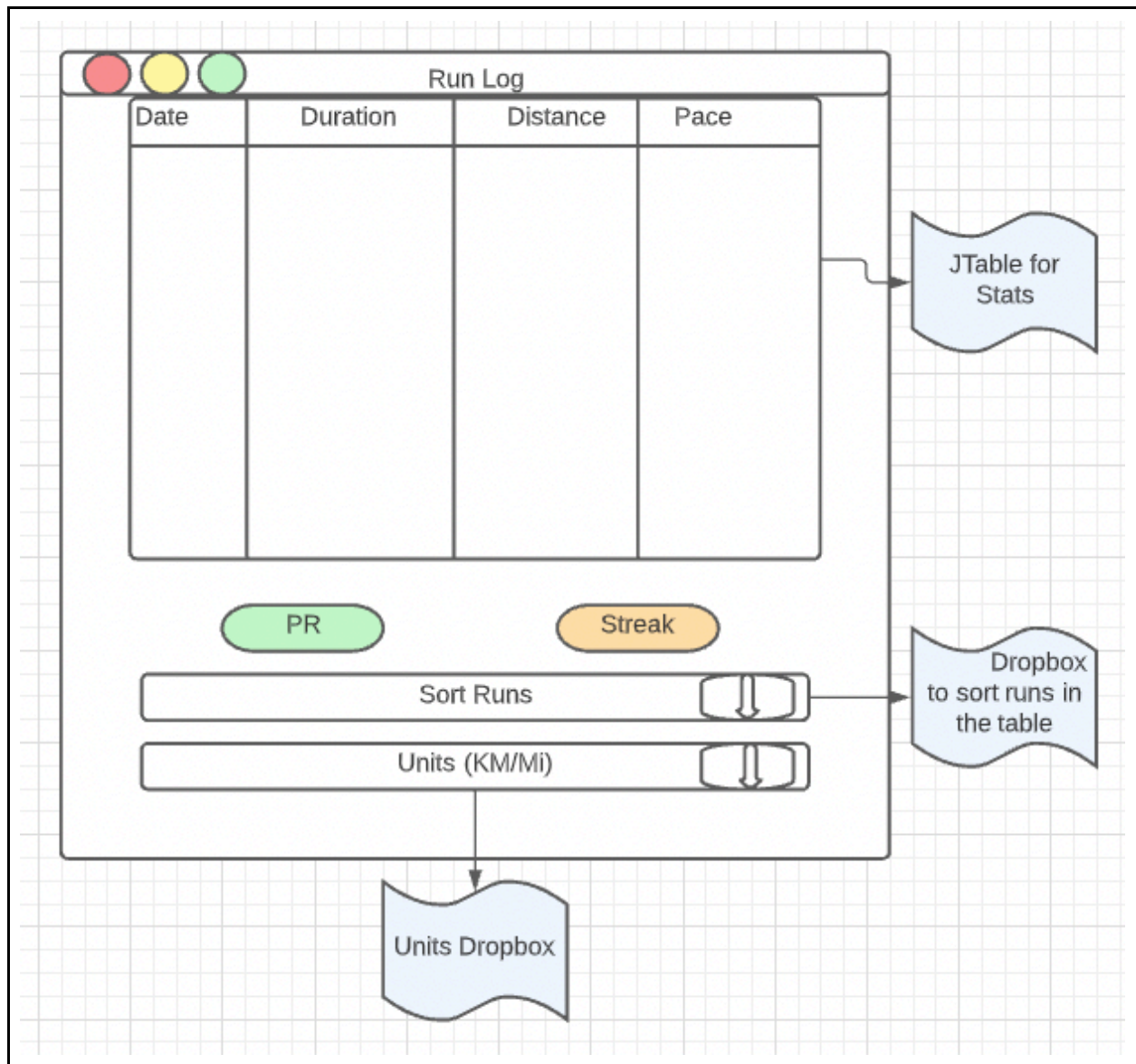
User Interface Prototype

Figure 3: Main UI for entering running Data

Main UI that will allow users to enter their running data.
- The Record Run button will process user inputted data and store it into a text file.

Figure 3: Running Log UI for entering running Data:

Run Log Panel will allow users to view their previous runs as well as their PRs and Streak.
- Different PRs will be displayed based on different distances.
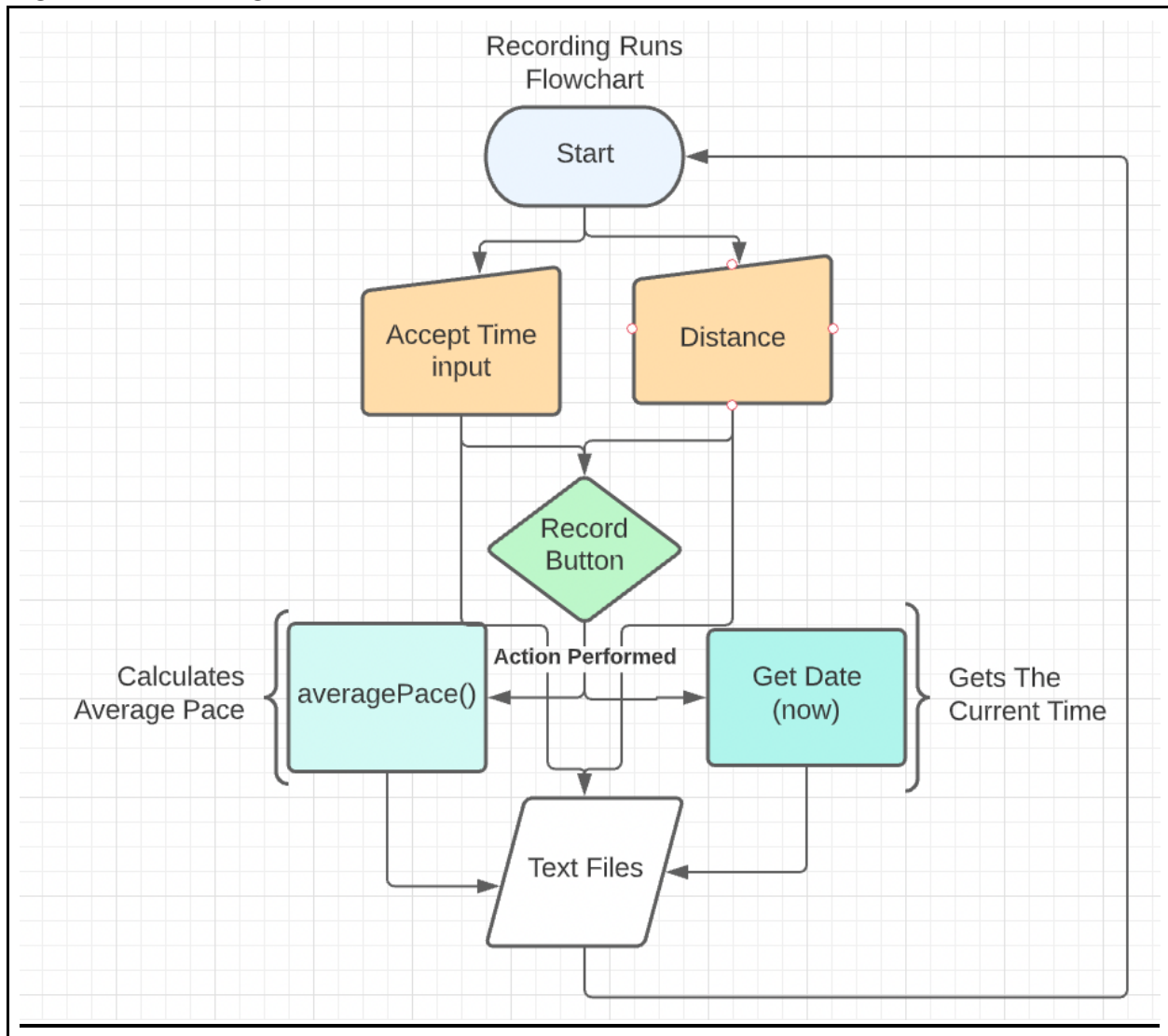
Process Description

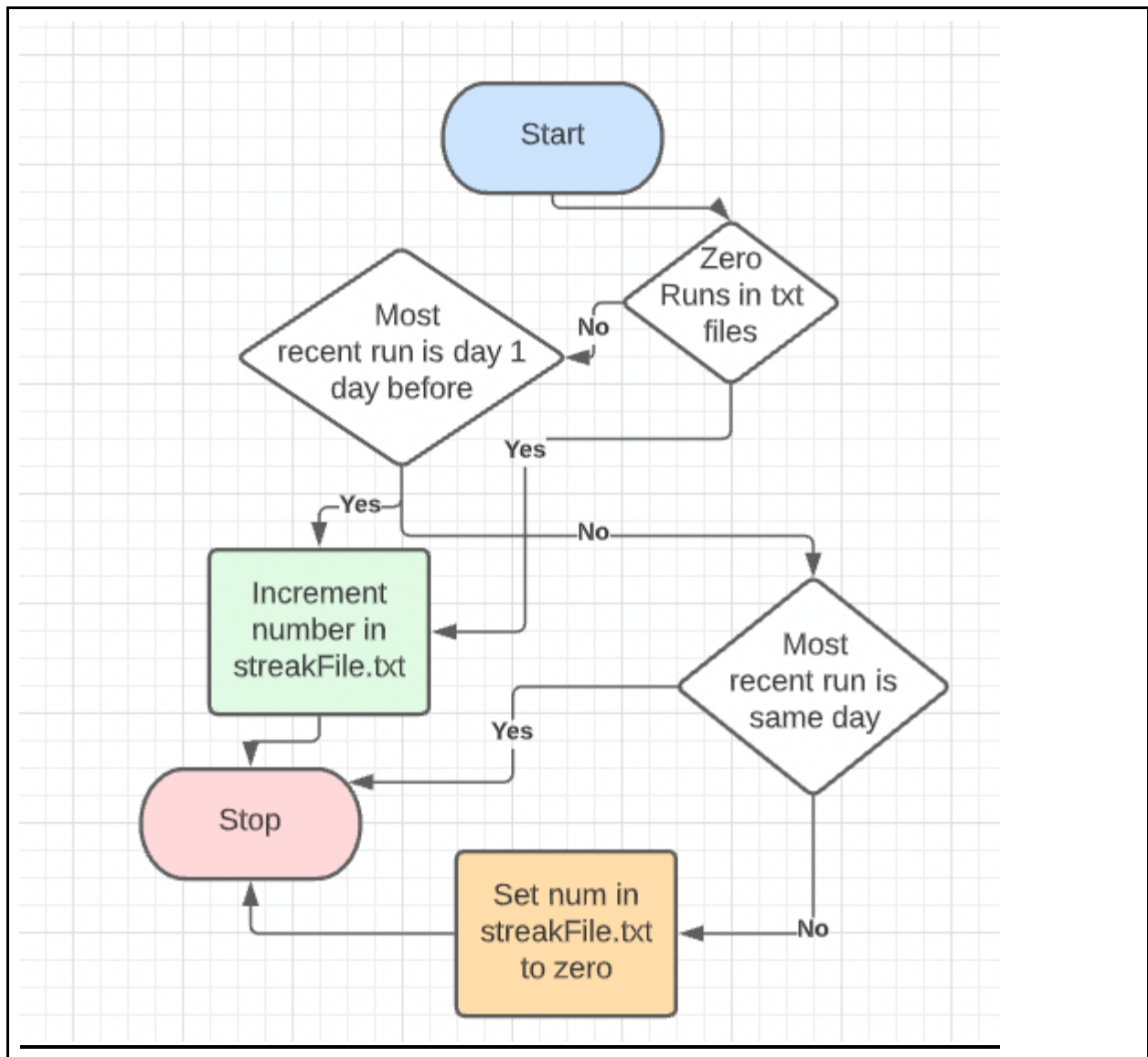Figure 4: Recording Runs Flowchart

Recording Runs
Flowchart

Start

Accept Time
input

Distance

Record
Button

Calculates
Average Pace

averagePace()

Action Performed

Get Date
(now)

Gets The
Current Time

Text Files

Figure 5: Streak-Counting System Flowchart

Figure 6: PR Calculating System Flowchart
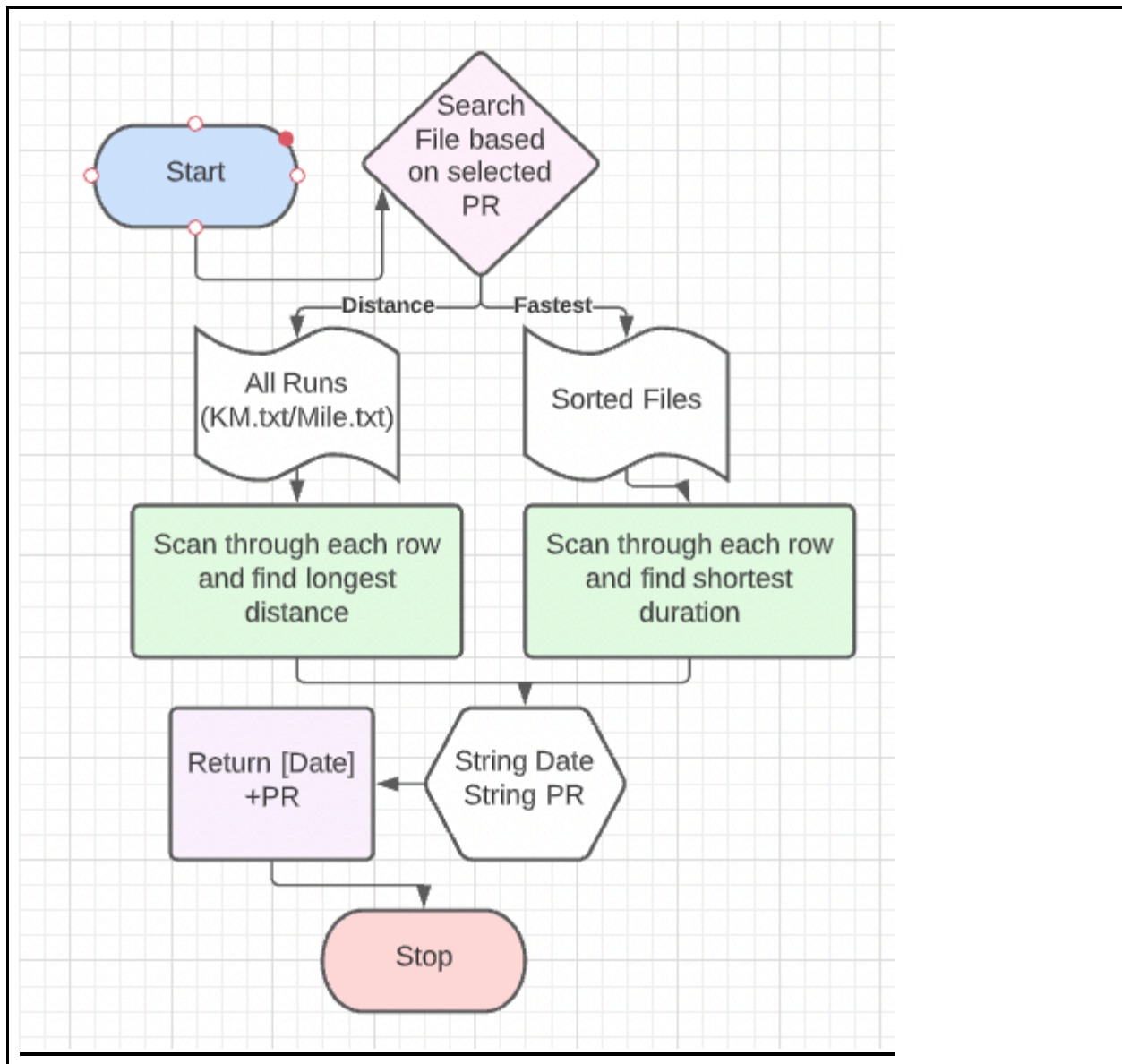
Figure 7: Find longest Distance Flowchart

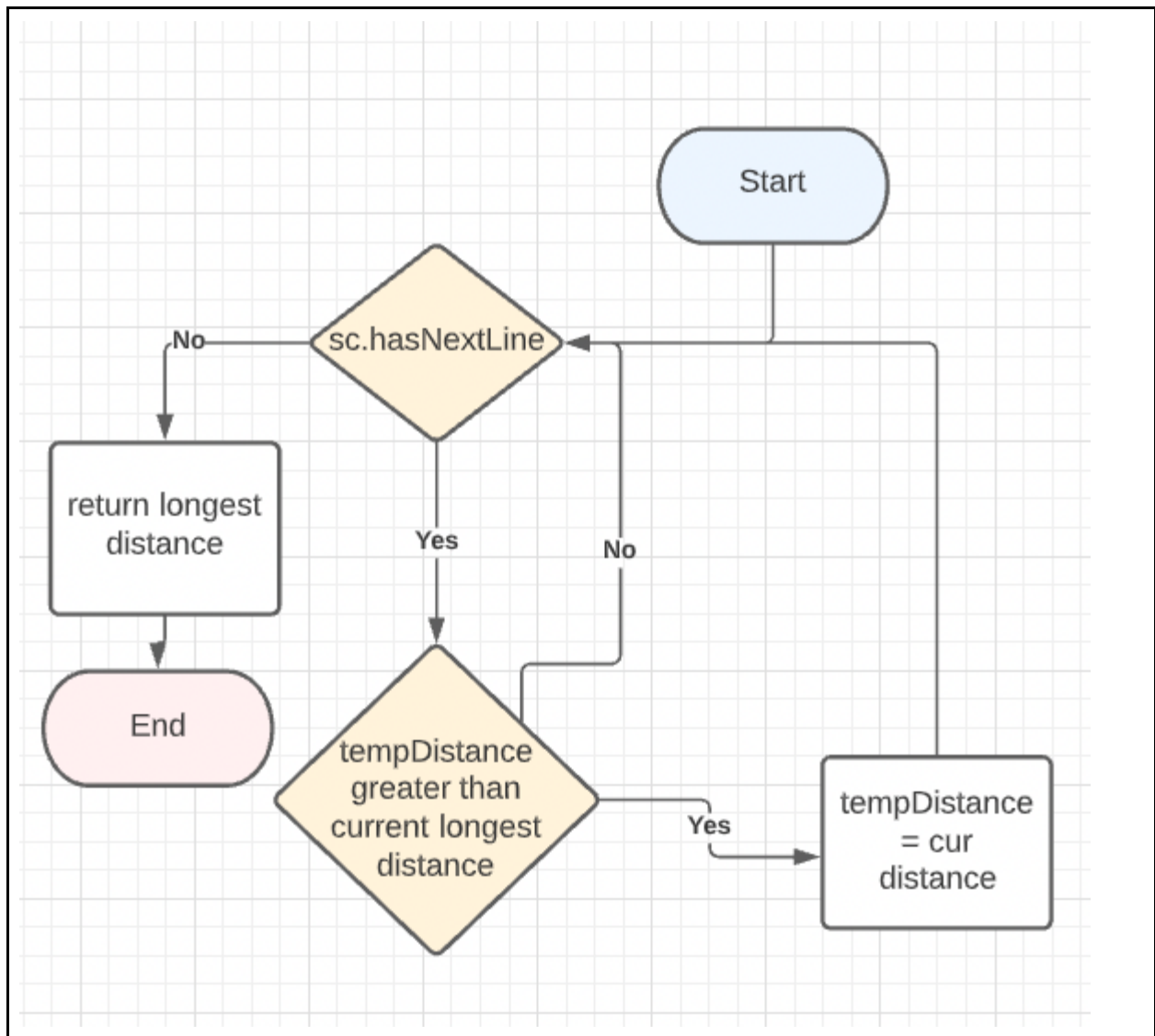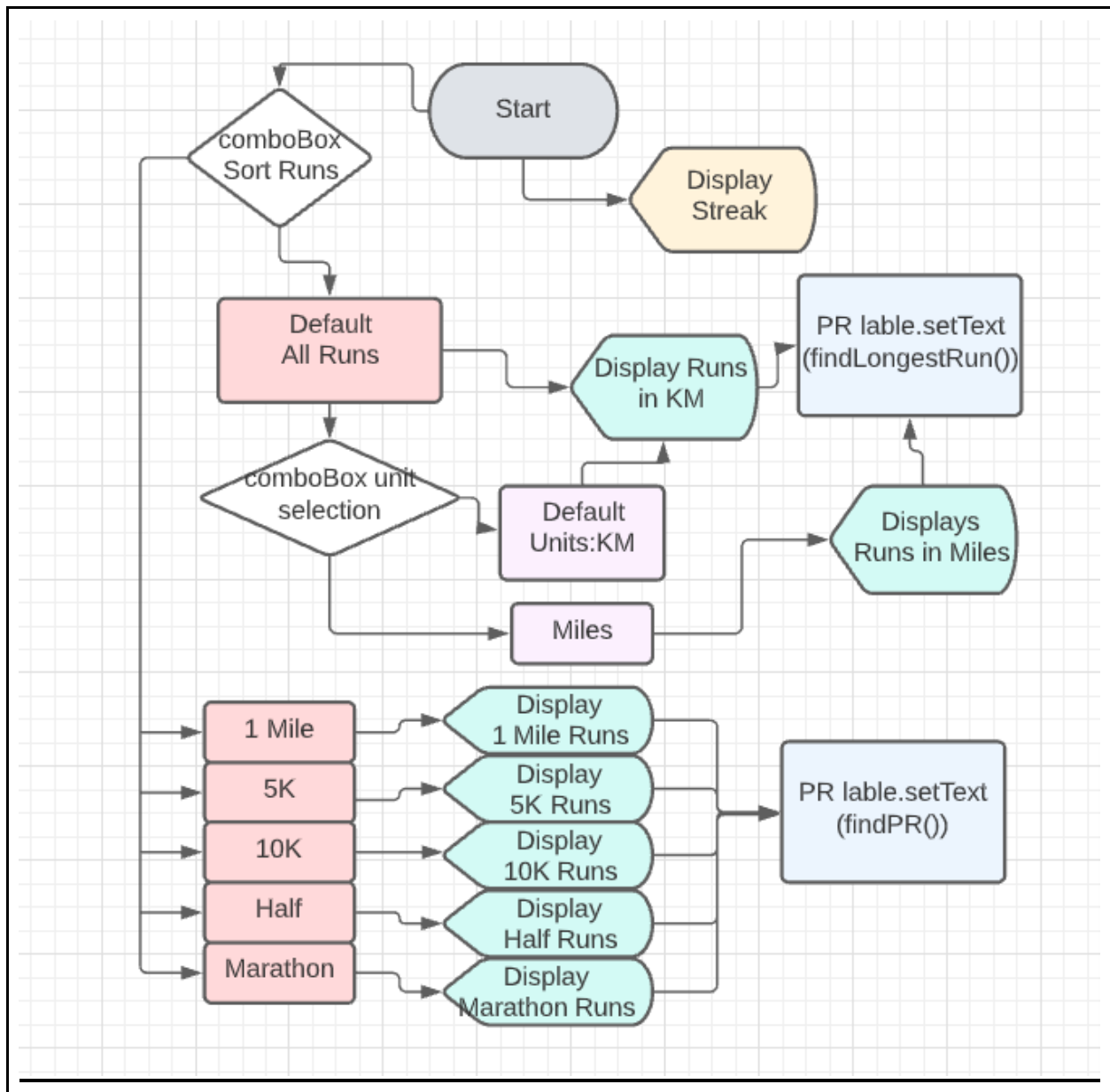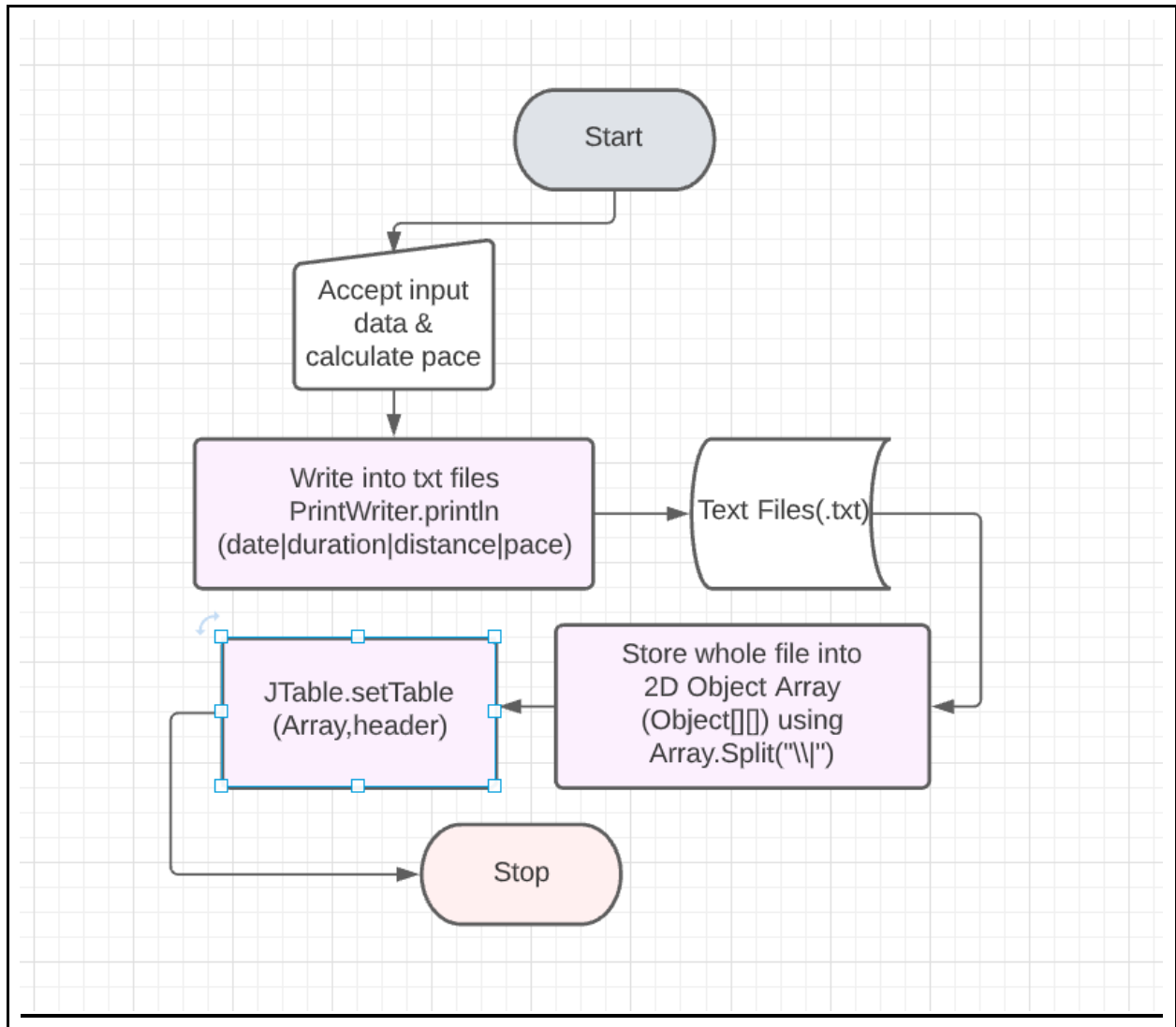Figure 7: Display Running Stats Flowchart

- Runs are stored in different text files so that it is easier to display sorted runs on the table.
- Storing Runs in separate text files also allows Personal records to be calculated easily.

Figure 8: Writing and Retrieving Data Flowchart

Text Files

<u>Specific Distances:</u>

specificDistanceFiles
- 1Mile
- 5K
- 10K
- HalfMarathon
- Marathon

- All files are .txt files

All Distances:
- kmData
- milesData

- All files are .txt files

Streak:
- streakFile

- streakFile.txt: stores an integer that represents the running streak.

Action Test	Way of testing and result	Success Criteria
Test to see if GUI button will work (simple program where button will increment number)	Run the program and click the button to see if the counter will increment.	N/A
Test to see if JTable will be correctly displayed.	Create a JTable and run to see if correct header and values will be displayed from a 2D Object array.	N/A
Test to see if the averagePace() method will calculate correct pace.	Call the static method in main and see if it returns correct pace given parameters specific (double distance, String time)	3

Test to see if user inputted data (run records) can be displayed correctly on the table.	Run the program to see if the correct values are displayed in each cell of the JTable. See if the number of runs match up with the number of rows in the JTable	4
Test to see if the Unit toggle dropbox works.	Run the program and enter "1" in the distance textbox and choose "mile" on the dropbox. Then check the JTable to see if 1.61km will show up when "km" is selected in the dropbox and if 1 mile will show up when "mile" is selected..	6 & 7
Test entering distances by using dropbox selection.	Select each distance and click the record run button to see if all the distances will show up correctly on the table and file.	2
Test to see if the table will display the correct set of runs when the user wants to sort their runs.	Run the program and toggle between the selections in the dropbox. See if the program will display the set of runs when specific distances are selected.	5
Test to see if the streak counting system works.	Enter a run to see if streak counting will increment. Then delete previous runs from the day before to see if streak resets if the user didn't input a run "yesterday."	1
Test findPR() method to see if the method will return the correct date and duration of fastest run.	Enter runs with lower time to each specific distance to see if it returns the correct PR. Also check if there's no runs, the PR will return "N/A"	8
Client testing: Give prototype to client to verify the prototype's functionality.	Clients will toggle with the controls and run tests to see if the program will meet their needs.	All

Word Count: 192