**Architecture**

**Design**

**Lap Time Recording Utility**

****

9/26/18

Customer: Solar Car Team

Instructor: Jane Hayes

*Team Members: Wesley Shiflet, Jake Yang, Hardikkumar Patel*

1. High-Level Design (Shiflet)

Our app is designed to record the lap times of the solar car team and their competitors in three easy and convenient ways. Using a simple interface design the user will be able to pick between three methods of recording lap times: manual entry, semi-automatic entry, and automatic entry. With Manual Entry the user is able to manually type in each time for each team which is useful for anytime that they may need to edit a time or if for some reason the other two methods aren't working or not feasible at the time. With the Semi-Automatic Entry there is a display with buttons that the user can use to conveniently track the lap times of each solar car team. With the Automatic Entry mode the app will use a webcam to watch cars as they pass the finish line and track their times by the team stickers on the side of the cars. In Figure 1.1 you can see a High Level Architecture Diagram of the setup of our application on the backend of all of the parts working together.

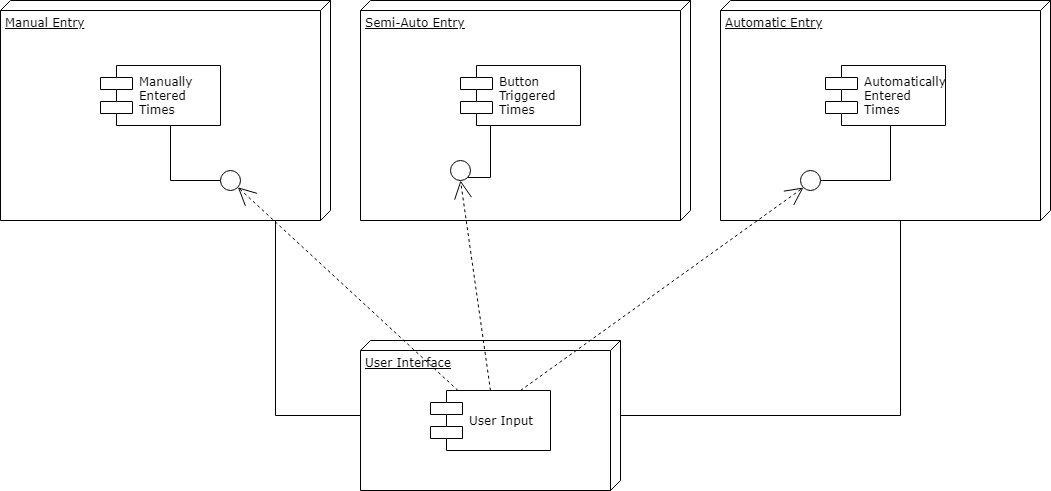


Figure 1.1: High Level Architecture Diagram of Lap Time Recording Utility

Our High Level structure consists of two main layers, the user interface and the then the method that will be used to record the lap time. These methods will work together in a smooth manner to allow the user to switch between them freely. Using each method when it is most appropriate. The manual entry mode will allow entry of a new lap time and also allow you to edit a previous time that had already been entered. The semi-automatic entry mode would be useful for when the automatic entry mode can’t be used due to a congested group of cars finishing at around the same time which would cause the stickers used from automated entry to be obstructed. And then the automated entry mode is useful for when cars are steadily streaming in and have their team sticker that is detected by the webcam clearly visible.

The data that is entered regardless of the method used will then be used to calculate and display in real time the last lap time, average lap time, slowest lap time, and fastest lap times. All of this data is relevant to the customer due to the nature of solar car races being more endurance based as opposed to speed based. As all of these times are recorded the data will be saved locally as well as to the cloud.

So our User Interface will interact with each method of recording the lap time by allowing the user to smoothly switch from each method for the most efficient process and displays relevant data to the team to allow to judge their performance in the race up until that point.

2. Detailed Design (Yang)

In the lap time record utility program, at the first we decided to use c++. However, after we met our customers, they wanted us to use python. In this program, we need three main windows for the user interface. First, we need a window that shows a table which has all data of lap times for each college. Second, we need a window that shows timer and which has start, lap, and stop buttons for each college. Third, we need a window for a graph that shows visual data for selected college. For time recording, we have three major class that can get and store data from a user.

LapTimeTable

- GetTeamName: function

- GetNumberOfLap: function

- GetLapTime: function

- MakeTable: function

First, We need LapTimeTable class that stores lap time data. GetTeamName function gets each team name or college name. And GetNumberOfLap function stores data into array so that we can define which team has which number and gets the number of laps for each team. GetLapTime gets the actual times for each lap times for each teams.

LapTimer

- GetTeamName

- startTimer: function

- oneLapTime: function

- stopTimer: function

- isStopped: bool

- isStarted: bool

- storeLaptime: array

Second, there is LapTimer class, there is startTimer function which indicates race start. The oneLapTime function gets time for one lap. The stopTimer function indicates race stop when a car passes finish line. isStopped is bool indicator which identifies race is stopped or not. IsStarted function is bool indicator which identifies race is started or not. The storeLaptime is an array that store lap times.

CollegeTeam

- teamName: function

- teamNumber: function

- AddNumberOfTeam: function

Third, we have CollegeTeam class. A user does not know how many teams he or she wants to put in data table. Therefore, using CollegeTeam class makes ability to add new team. The teamName function gets a name of team from user. The teamNumber function gets a team number. The AddNumberOfTeam counts the total number of teams in the race.

2.2 User interface design (Yang)

Figure 2-1

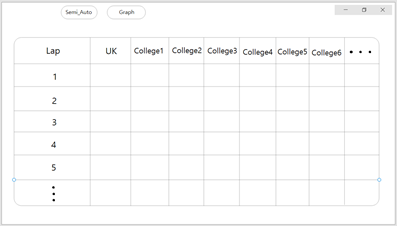


Figure 2-1 shows main interface of the lab time recording utility. Purpose of this utility is to show lap times of each college teams. This layout simply provides information of lap times. Using table makes easy to match lap times, numbers, and the colleges. The program does not know number of lap times and number of colleges. Therefore, user simply can add colleges. The user can add or remove colleges and can edit lap times. There are two buttons on the top. Semi-Auto button is for a new window.

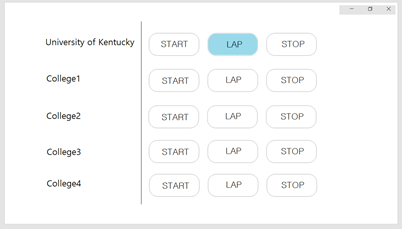
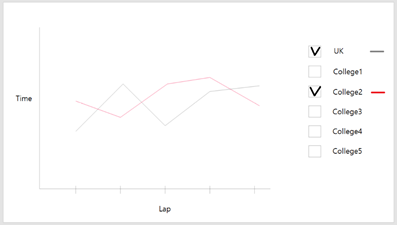
Figure 2-2  


Figure 2-2 shows the new window of Semi-Auto mode. In this mode we can add colleges if needed. There are three buttons for start, lap and stop. When lace is started, the start button initiate stop timer. The Lap button is for recording one lap times. Lastly, the stop button is used for the time when solar car passes finish line. In this figure, we provide three buttons for each team, because there is time when each team starts at different time and definitely they will pass each lap in different lap times. If we click the lap time button for each team. The lap time data will store in the table in figure 2-1.

Figure 2-3



In Figure 2-1, there is a graph button on the top. Those, Semi-Auto and graph buttons are not for taps. They will make new windows, because the user wants to see lap time table, semi-auto mode, and a graph at same time. In Figure 2-3, the graph is simple line graph which has time at y-axis and laps for x-axis. Our customer wants to compare lap times. By selecting teams on the left, the graph only shows selected team’s data. Therefore, the user easily can compare his or her records with the other team’s records.

2.3 Design Patterns (Yang)

Our program is for window operating system. This lap time record utility is for only one customer which is the UK solar car team. Therefore, the design is simple to show lap time data. We are not using many colors. Main theme color is gray that makes outline of a table, buttons, and outline of a graph. Our goal of the design is to show lap time data intuitively. Therefore, if there are many colors in main user interface, they will interrupt when the user is watching records. There is only one window that uses lots of color. The graph window uses many color, because there are many teams and in line graph using different color makes the graph intuitive.

3. Testing (Patel)

We have two user cases as semi-auto lap time and auto lap time to implement to our customer and these two user cases depends on user. In other words, functionality in two cases revolves around what the user can do with lap time. To do this, we have created test cases in order to check that all functions we implemented work correctly. We will use these test cases to check if the code works means that what we write in the code does what we intended to do. In the Figure 3.1, it contains the case number as only two cases, corresponding parts of that case number in any order, the test case, pass condition means that conditions that must be satisfied for corresponding test to be pass, and fail conditions means that conditions fail for corresponding test case.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case Number | Part | Test Case | Pass Condition | Fail Condition |
| 1 | 1A | User enter their name/team number then click enter button | It should display team name/number with manual entry mode as the start button, the lap button, and stop button after clicking on the enter button. | It not displaying the team name/number and/or manual entry mode. It might not display one of the buttons. |
| 1 | 1B | When user enter their name/team number after clicking the enter button, system receives data | System receives their name/team number | System doesn’t receive their name/team number |
| 1 | 2 | User click on the Start Button | It should start displaying the time on the screen when the Start Button clicked. | Not displaying the time on the screen nor time is starting when the Start Button clicked. |
| 1 | 3A | User click on the Lap Button for the first time | It should display lap with corresponding time and time should be continue moving when the Lap Button clicked for first time. | Not displaying the lap and/or time not continue moving when the Lap Button clicked for the first time. |
| 1 | 3B | User click on the Lap Button for more than one time. | It should display first lap with corresponding time and also display corresponding lap that user clicked after once with its corresponding time. And also, time should be continue moving when the Lap Button clicked for more than once. | It not displaying first lap and/or corresponding time. Nor displaying lap that user clicked more than once with corresponding time. And/or time not continue moving. |
| 1 | 3C | When user clicked the Lap Button first time, system collect the data | System collect the data for corresponding lap time with corresponding team name/number | System didn’t collect the data for lap time and/or didn’t respond with correct team |
| 1 | 3D | When user clicked the Lap Button for once or more | It should display corresponding lap time and lap number | It not displaying the lap time and/or lap number |
| 1 | 4A | User click on the Stop Button | It should stop the time and display the total time as well as the lap time and corresponding lap number | It not stopping the time and/or not displaying the total time and all lap time and/or corresponding lap number deleted |
| 1 | 4B | When user click on the Stop Button, system receive the total time | System receive the total time and total lap number | System didn’t receive the total time and/or total lap number |
| 1 | 5 | After all lap done, system has all team data | System receive all team data with corrected number of laps | System didn’t receive all team data and/or missing one or more team number of laps |
| 1 | 6 | System creates a table | System creates a table that shows team name/number with their corresponding lap number and time | System can’t create a table and/or not displaying right data |
| 1 | 7 | User edit button in system end for change time if the system didn’t receive right data that matches the user data | Edit button allows the user to make any changes if data doesn’t match with user and system | Edit button doesn’t allows the user to edit |
| 1 | 8 | User save button in system end that user save the data if user data matches the system data | Save button allows the user to save their data | Save button doesn’t allows the user to save their data. |
| 1 | 9A | User undo button in system end that activate when user edit their data | Undo button activate when in edit mode | Undo button doesn’t active under edit mode |
| 1 | 9B | User undo button in system end that goes back when clicked on undo button | Undo button works properly when clicked | Not working properly |
| 1 | 10 | Under system, every changed made in system must be saved to backhand | It saved the changes made in the backhand when changes made to secure the data from crash or loss of power | It’s not saving automatically in the backhand |
| 1 | 11A | Graph view activate when clicked on Graph mode | Graph view display when clicked on the Graph mode | Graph view not display when clicked on the Graph view |
| 1 | 11B | Graph view with check box to display its team on the graph | When clicked on the checkbox, it should display its data. | Not display its data when clicked on checkbox |
| 1 | 11C | Graph view with multiple checkbox to compare with another team | When clicked on the multiple checkbox, it should display corresponding team with different color to compare them. | Not displaying proper data when clicked on checkbox nor generating right color with legend. |
| 2 | 1 | User starts the automatic mode | It should start when race begin | Not responding when race started |
| 2 | 2 | User watches the automatic system is recording lap times | Automatic system working properly by looking at user display | It not working properly and not recording their lap time |
| 2 | 3 | System collects the data with corresponding team and its time | System collects the data with corresponding team with its time | System not collecting the data |
| 2 | 4 | System detect with webcam | Webcam works properly means its able to see the team number | Webcam not working properly means can’t able to see |
| 2 | 5 | User should assist the time if webcam fails when car passing | User should get message if webcam failed and start on manual mode | User didn’t get message nor they know webcam failed |
| 2 | 6 | User stop the automatic mode with the race is over | User should able to stop this mode when race is over | User can’t able to stop this mode |

**Figure 3.1**

Figure 3.1 shows all the possible test cases that we can use to test when we done with our code to check if all the functionality works properly. Under case two, backhand or system part is same as the case one like the graphical mode, and table mode. Therefore, we didn’t include this part in the Figure 3.1. If one of the test conditions failed, then we will rewrite the code and check again if works. Otherwise, if the all conditions passes, then we will think that our code work properly.

# **4. Hold a Review (Group)**

# **1. There is no graphical mode for the manual entry.**

# **2. For manual entry, we can create the simple table after we gather all the data.**

# **3. There is still confusion on whether we use classes or functions, but we state function as of now.**

# **4. There were some unclear wording in this assignment.**

# **5. There were 2 test cases that needed to clarification and there was one more test case that needed to be added.**

# **6. There was some more detailed needed in the High-Level Design section.**

# **7. There were some minor typos in Test Case section.**

# **8. We added better design after first one to make it more clear.**

# **5. Metrics**

# **Complexity of Overall System (Patel)**

# **The maximum depth of any inheritance tree in our code will be 3.**

# **Product Size**

# **We have planned our lap time utility based on the semi-auto case and auto cases. We have a total of 10 functions planned and 17 test cases for both user cases.**

# **Product Effort (Patel)**

|  |  |  |
| --- | --- | --- |
| **Name** | **Hours** | **Word Count** |
| **Hardikkumar Patel** | **9** | **1366** |
| **Wesley Shiflet** | **7** | **515** |
| **Jake Yang** | **11** | **840** |

# **Defects**

# **There were 15 noted defected that have been addressed.**

6. Keep a Developer Notebook

Our team uses Github to track our developer logs and other contents for project such as meeting times. Each team member can access and edit our project logs. We also use google documentation for the project writing, so that we can track contents of each other’s documents. We will add more contents throughout the semester. Our developer webpage is https://github.com/sya224/Lap-time-project/issues.