A drawing of a face

Description automatically generated

ICT 2101/2201 Introduction to Software Engineering

Software Requirements Specifications (SRS)

Milestone 2

AY 2021/2022, Trimester 1

**Robot Car Website System**

Lab group P1-4

Kidz Program Car Co.

Team Leader: Tiffany Ho(2002170)

Team Member: Alaric Tang (2000890)

Team Member: Nicholas Lum Ye Han (2001605)

Team Member: Lim Hong Ying (2001631)

Team Member: Tan Yu Hui (2001632)

**TABLE OF CONTENT**

[**1. Introduction**](#_heading=h.jwsa49l71ur) **4**

[1.1. Product Scope](#_heading=h.t6rgkox3xa3) 5

[1.2. Intended Audience and Document Overview](#_heading=h.i8chhyqkbble) 5

[1.3. Definitions, Acronyms and Abbreviations](#_heading=h.6qzmhnvbhe30) 5

[1.4. Document Conventions](#_heading=h.4sc1ceupmb8e) 5

[1.4.1 Formatting Conventions](#_heading=h.4eu4wglnbm10) 5

[1.4.2 Naming Conventions](#_heading=h.j3u5guxxn6a4) 6

[**2. Overall Description**](#_heading=h.7umyrt3knwa6) **6**

[2.1. Product Overview](#_heading=h.2et92p0) 6

[2.2. Product Functionality](#_heading=h.tyjcwt) 7

[2.3. Design and Implementation Constraints](#_heading=h.n1xz0ep1fbwa) 7

[2.4. Assumptions and Dependencies](#_heading=h.ygxmohtudq7g) 8

[**3. Specific Requirements**](#_heading=h.q48s0hxiofi0) **8**

[3.1 External Interface Requirements](#_heading=h.3dy6vkm) 8

[3.1.1 User Interfaces](#_heading=h.1t3h5sf) 8

[3.1.2 Hardware Interfaces](#_heading=h.4d34og8) 10

[3.1.3 Software Interfaces](#_heading=h.2s8eyo1) 10

[3.2 Functional Requirements](#_heading=h.ggs8fgo8bffq) 11

[3.3 Functional Requirements Validation](#_heading=h.94s5kl24x3vr) 11

[3.4 Use Case Model](#_heading=h.lnxbz9) 12

[3.4.1. Use Case #1 - View list of sensor inputs (Id: UC-1)](#_heading=h.gwu8ayytneog) 12

[3.4.2. Use Case #2 - View history of commands used (Id: UC-2)](#_heading=h.6qu5bbx9jya2) 13

[3.4.3. Use Case #3 - View chosen commands (Id: UC-3)](#_heading=h.9x7rtabvye8h) 14

[3.4.4. Use Case #4 - Select commands (Id: UC-4)](#_heading=h.lrfcteybrrmo) 15

[3.4.5. Use Case #5 - Delete chosen inputs (Id: UC-5)](#_heading=h.s0vbeufxmieq) 16

[3.4.6. Use Case #6 - View connection status between website and robot (Id: UC-6)](#_heading=h.bfoekt3wabb5) 16

[3.4.7. Use Case #7 - See feedback and status (Id: UC-7)](#_heading=h.n1zljjf10hez) 17

[3.4.8. Use Case #8 - Send command (Id: UC-8)](#_heading=h.l0x9hbw2riqo) 18

[3.4.9. Use Case #9 - Detect movement commands (Id: UC-9)](#_heading=h.4e7f0wh32ajv) 19

[3.4.10. Use Case #10 - Send sensor data (Id: UC-10)](#_heading=h.rb4tfoa00ylc) 20

[**3.4.11. Use Case #11 - Create map challenges (Id: UC-11)**](#_heading=h.auvnz9oem5th) **20**

[3.4.12. Use Case #12 - View basic tutorial mode(Id: UC-12)](#_heading=h.q37x4c8wjyzx) 21

[3.4.13. Use Case #13 - Play map challenge (Id: UC-13)](#_heading=h.26703d9tjux8) 22

[3.4.14. Use Case #14 - See feedback and status (Id: UC-14)](#_heading=h.2wa4pt6zqqgq) 23

[3.4.15. Use Case #15 - View connection status between website and robot (Id: UC-15)](#_heading=h.wm0ufth7dl1x) 24

[3.4.16. Use Case #16 - View History of Command Use Per Challenge Per User (Id: UC-16)](#_heading=h.od2ennpn00bb) 24

[3.4.17. Use Case #17 - View list of sensor inputs (Id: UC-17)](#_heading=h.hqx3e4smwgps) 25

[**4. Other Non-functional Requirements**](#_heading=h.pbduwm5m5b9n) **26**

[4.1. Performance Requirements](#_heading=h.rrspq7llnb79) 26

[4.2. Safety and Security Requirements](#_heading=h.xw58xtmvtq5e) 26

[4.3. Software Quality Attributes](#_heading=h.ig6wvzw2two9) 26

[4.3.1 Security](#_heading=h.4vyy2n4qbd5c) 26

[4.3.2 Usability](#_heading=h.h37xys8zyoj7) 26

[4.4 Non-functional Requirements Validation](#_heading=h.4ohyt66ki12) 26

[**5. Architectural and Detailed Design**](#_heading=h.cal2whcq3viw) **27**

[5.1. Class Diagram](#_heading=h.sx6vbiak7brr) 27

[5.1.1. Problem statement](#_heading=h.vuhtf3chrdlz) 27

[5.1.2. Identifying Candidate classes](#_heading=h.88zj8n30nwpw) 27

[**5.1.3. Identifying Subclasses**](#_heading=h.vzcn3e8l0vw0) **27**

[**5.2. Component Diagram**](#_heading=h.b61qn7t4usyk) **31**

[**5.3. Sequence Diagram**](#_heading=h.8xq43oopbgyb) **32**

[5.4. Software Architecture and Design Pattern](#_heading=h.q0qi8py268lu) 35

[5.4.1. Software Architecture](#_heading=h.74aj78hix1ma) 35

[5.4.2. Software Design Pattern](#_heading=h.pjfkegw8maex) 36

[**6. Testing and Evaluation**](#_heading=h.bnmcumb0c7ec) **39**

[**6.1. System State Diagram**](#_heading=h.zdrteo4a17s) **39**

[**6.2. System Test Cases (User Acceptance Testing)**](#_heading=h.qibzdgrhza5w) **39**

[**6.2.1 Test case 1 (ID:1)**](#_heading=h.cas60kshyyy0) **40**

[**6.2.1 Test case 2 (ID:2)**](#_heading=h.wijci6stv0pr) **40**

[**6.2.1 Test case 3.1 (ID:3)**](#_heading=h.e5q8e6lviof2) **40**

[**6.2.1 Test case 3.2 (ID:3.2)**](#_heading=h.7tcx53pysuvu) **41**

[**6.2.1 Test case 3.3 (ID:3.3)**](#_heading=h.lhn7yo8x0v36) **41**

[**6.2.1 Test case 4.1 (ID:4.1)**](#_heading=h.9rphgz7irfis) **41**

[**6.2.1 Test case 4.2 (ID:4.2)**](#_heading=h.7hlr3goon5tf) **42**

[**6.2.1 Test case 5 (ID:5)**](#_heading=h.gg8vd2gvs2d7) **42**

[**6.2.1 Test case 6 (ID:6)**](#_heading=h.s0xwkgfbpk5s) **42**

[**6.2.1 Test case 7 (ID:7)**](#_heading=h.6pdugyx8h1ov) **43**

[**6.2.1 Test case 8 (ID:8)**](#_heading=h.g81vb8na7b8c) **43**

[**6.2.1 Test case 9 (ID:9)**](#_heading=h.myaaahwjso53) **43**

[**6.2.1 Test case 10 (ID:10)**](#_heading=h.7ubejwukp37r) **44**

[**7. Work Distribution & Plan**](#_heading=h.u88yurmuyg0p) **44**

[7.1. SDLC Approach](#_heading=h.3f8g823wa83i) 44

[7.2. Project estimation and Scheduling](#_heading=h.a2hol62v53g1) 44

[**8. Conclusions**](#_heading=h.ajo9yzjvylo7) **45**

[**9. Individual Reflections**](#_heading=h.3wlylwp3qd5r) **46**

[Tiffany](#_heading=h.ajwzgfdr14u9) 46

[Alaric](#_heading=h.k52ypw4a7sxu) 46

[Nicholas](#_heading=h.ocyv90wdg441) 46

[Hong Ying](#_heading=h.yud5721sm0vi) 46

[Yu Hui](#_heading=h.z7hk1z6d7uod) 47

[**10. References**](#_heading=h.8qgckgxpqt6v) **47**

[**11. Appendix A**](#_heading=h.hrv88o431h5r) **47**

# 1. Introduction

This documentation is a Software Requirement Specification (SRS) for the Robot Car Website System. The document is not final and may be subject to changes in the future. In addition, this document abides by the format of IEEE conventions for SRS.

Robotics in education has become a prominent element in this era of cutting-edge technology. [1] Studies have shown that robots can improve problem-solving skills and help students learn computer programming, mathematics, and science. It is, therefore, imperative that schools integrate robotics learning with students at an early age. With that in mind, the Robot Car Website System works as a web-based application that serves as a platform for students of age group 7-12 to control a robotic car's movement. The objective is to encourage students to exercise critical thinking about the underlying logic of computers in a fun and engaging way.

## 1.1. Product Scope

The web portal will be the software used by students to learn programming logic via interaction with the robotic car. As an aid to teaching, the web portal is constructed in the form of a gamified dashboard. The dashboard is used because of its advantages. First of all, it is relatively easy to utilize for primary school students. Next, it is also highly interactive since it comes with all sorts of graphical representations. As such it allows feedback of the robotic car to be displayed out in an interesting manner. Furthermore, a gamified system provides an interactive interface that makes learning logic more engaging for students.

## 1.2. Intended Audience and Document Overview

This document is written with the intent for the client, developers, and professors working on the Robot Car Website System project. It consists of seven main sections. In section 2, the document will summarize the product features in layman's terms. In this way, both the client and the professor can understand what the product is.

Following that, there will be two sections detailing all the functional and non-functional requirements of the product in technical terms (Section 3, 4). These chapters focus on helping the developers to understand how the product works. As such, they can then focus on building the product according to the requirements. Next, the document will outline the SDLC adopted and closely followed in this project, including its cost and schedule.

Finally, the document will wrap up with a conclusion on the summary of the project and its timeline together with the learning points gained in writing this SRS (Section 5, 6, 7). This document is best to read chronologically to maximize its effectiveness.

## 1.3. Definitions, Acronyms and Abbreviations

* GUI - Graphical User Interface
* i.e. - In other words
* MCU - Microcontroller
* NA - Not Applicable
* SDLC - Software Development Life Cycle
* SRS - Software Requirements Specifications
* UC - Use Case
* UI - User Interface

## 1.4. Document Conventions

### 1.4.1 Formatting Conventions

* Sections follows 20px, Calibri, bold, black
* Subsections follows 16px, Calibri, bold, black
* Any other sections follows 14px, Calibri, black
* Normal text follows 11px, Arial, black
* Paragraphs are left align, margin space 1.5
* Numbering for all sections and subsections follows X.X.X format (Where X = a number)
* Tables are used mainly for writing UC descriptions with border of 1.5pt
* Bulleted points are used for highlight short points
* Parentheses are used to describe the point in more detail. Used at the end of the sentence
* Italics are used as a comment

### 1.4.2 Naming Conventions

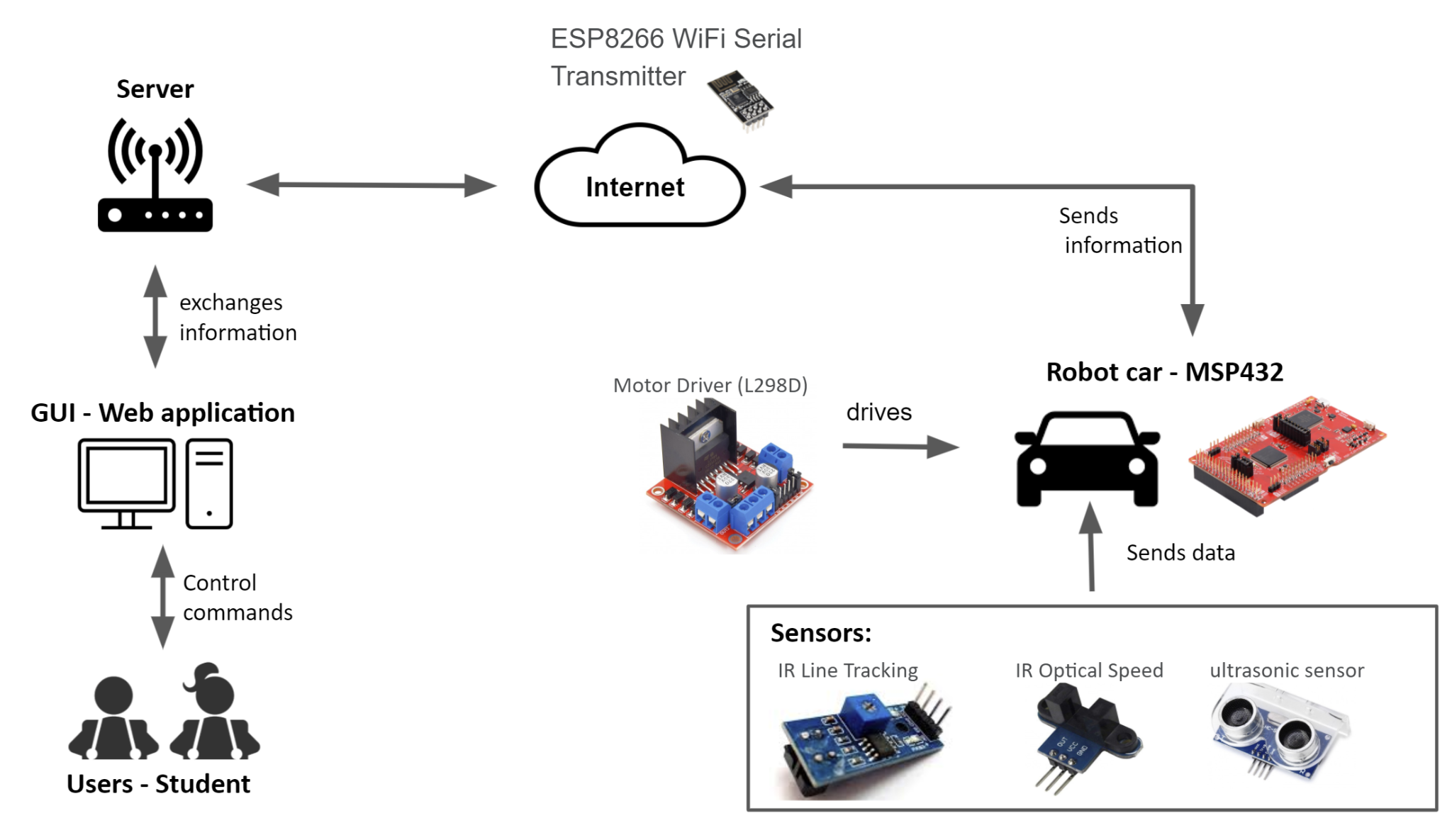
* Commands/ Inputs - refers to the instructions given from a student to the car (i.e. Sending directions for car to move)
* Feedback - refers to the status of the robot car (i.e. The current speed of the car, the current distance travelled by the car, etc.)
* TI MSP432/ MSP432 - Microcontroller that is created by Texas Instruments

# 2. Overall Description

## 2.1. Product Overview

The Robot Car Website System is intended for primary school students to use solely for education purposes. The system deployed is an online web interface where students can easily access it using desktops or laptops. This project will be a follow on product based on products such as Lego Mindstorms, however it is improved as for example with Lego Mindstorms, the programming needs to be done via connecting to the computer via a cable while for the Robot Car Website System, programming can be done wirelessly via WiFi. The online web portal will be the central interface used by students to send commands to control the movement of the robot car and receive feedback on the robot car. Feedback received will be displayed out via the web portal. Also on the web portal a student can create a profile while there will be a teacher account that will be able to check on the student’s progress and create challenges for students to complete.

Students must click on different car movement commands on the web portal to move the robot car successfully. Commands will then be stored on the server and sent over the internet to the car through an ESP8266 WiFi serial transmitter. From the external point of view, the car's built-in sensors will detect and change according to the commands indicated by the students. The sensors will then drive the robot car to exhibit various behaviors. These behaviors will reflect on the web portal through transferring data using WiFi and server.



*Figure 1: System architecture*

## 2.2. Product Functionality

The major functions of the Robot Car Website System are as follows:

* Students are able to select and send commands from the web portal to control the movement of the robotic car.
* The web portal allows students to choose a sequence of movement commands to the robotic car and the car will move accordingly as indicated.
* The web portal will display feedback of the robotic car to the students so that they can view the status of the car.
* Be able to create student profiles and a teacher mode where teachers can keep track of students' progress and create challenges for them.

## 2.3. Design and Implementation Constraints

The system will be constrained to only one device as the database is local. Since it can only be used by one device, in order to scale up the project, a cloud database will need to be used so that multiple users can use the system concurrently.

Furthermore, the system also identifies programming constraints. To ensure consistency among the developers when implementing the web portal, the team decided to limit the use of programming language to only driverlib from Texas Instrument, C, PHP and html.

There are design conventions in this system as well. There is a need for physical space for the robot to move around freely or according to the challenges set. The students are required to choose at least one command for the robot car to move and clear the stages. In any situation the car is moving and the student is in front of the robot, the car will stop moving and the system will display a message.

## 2.4. Assumptions and Dependencies

Assumptions:

* SQL databases are local and tables can only be edited by active developers
* The virtual and physical robot car will move concurrently
* The physical and virtual map will be the same.

Dependencies:

* The web portal depends on the robot
* The robot depends on the web portal

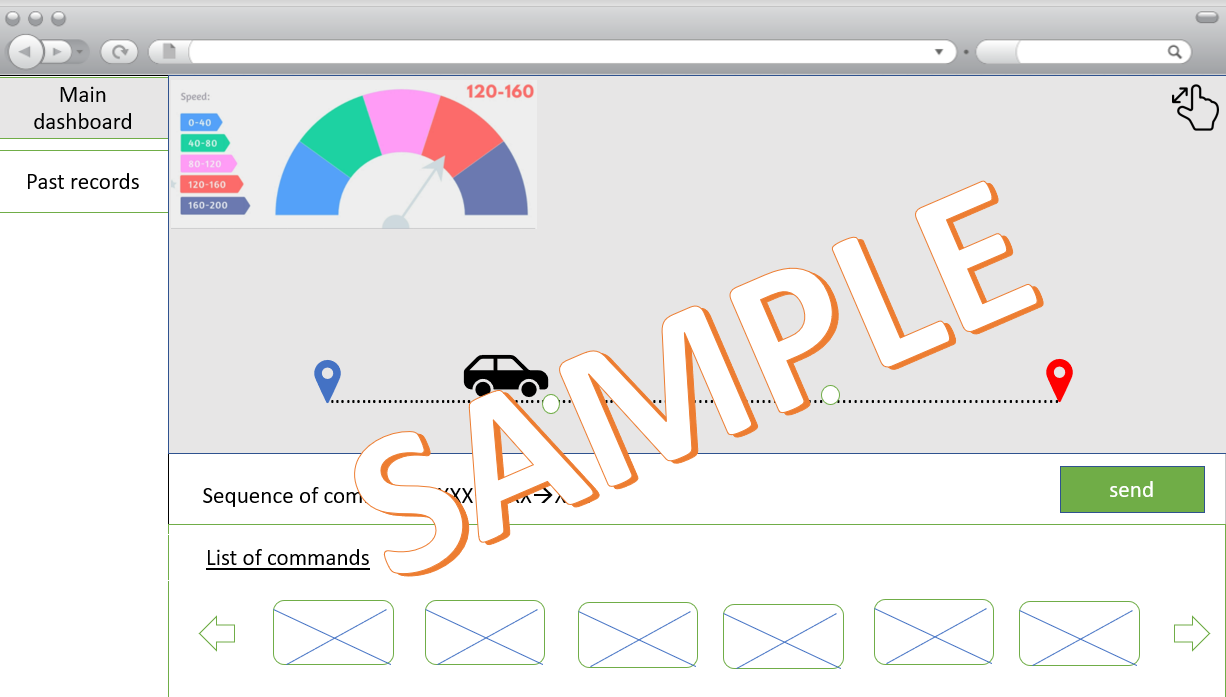
For instance, the web portal needs to get data feedback from the robot car. Without feedback, the portal will be unable to show car status information to the students. Hence, the web portal is dependent on the robot. On the other hand, the robot depends on a web portal for commands at the students' side to be sent to them in order to execute the movements.

# 3. Specific Requirements

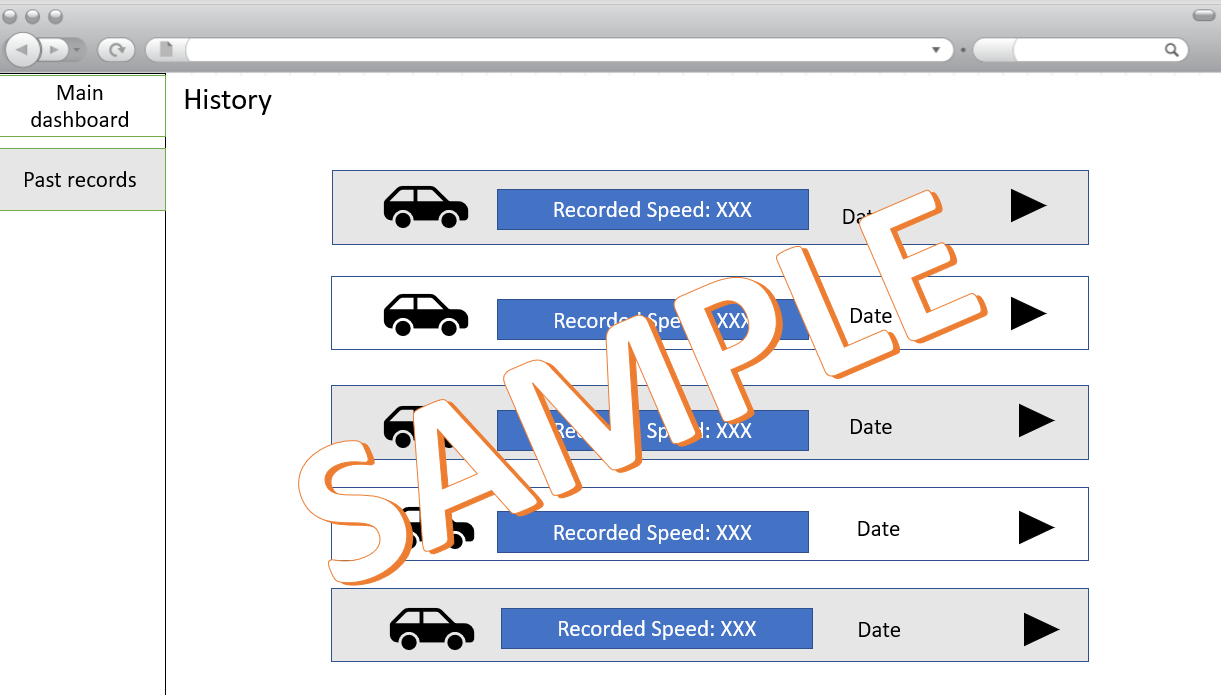
## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The students will interact with the software through a web portal. The web portal will consist of two dashboards and a board to display the available movement commands for the students to choose. The first dashboard will be used to display the feedback picked up by the robot’s sensors. On the other hand, the second dashboard will be used to display the student’s history of sequences of commands. In order to send the commands to the robot, the students will click the send button to send the commands to the robot wirelessly. Below shows the mockup of the web portal GUI.

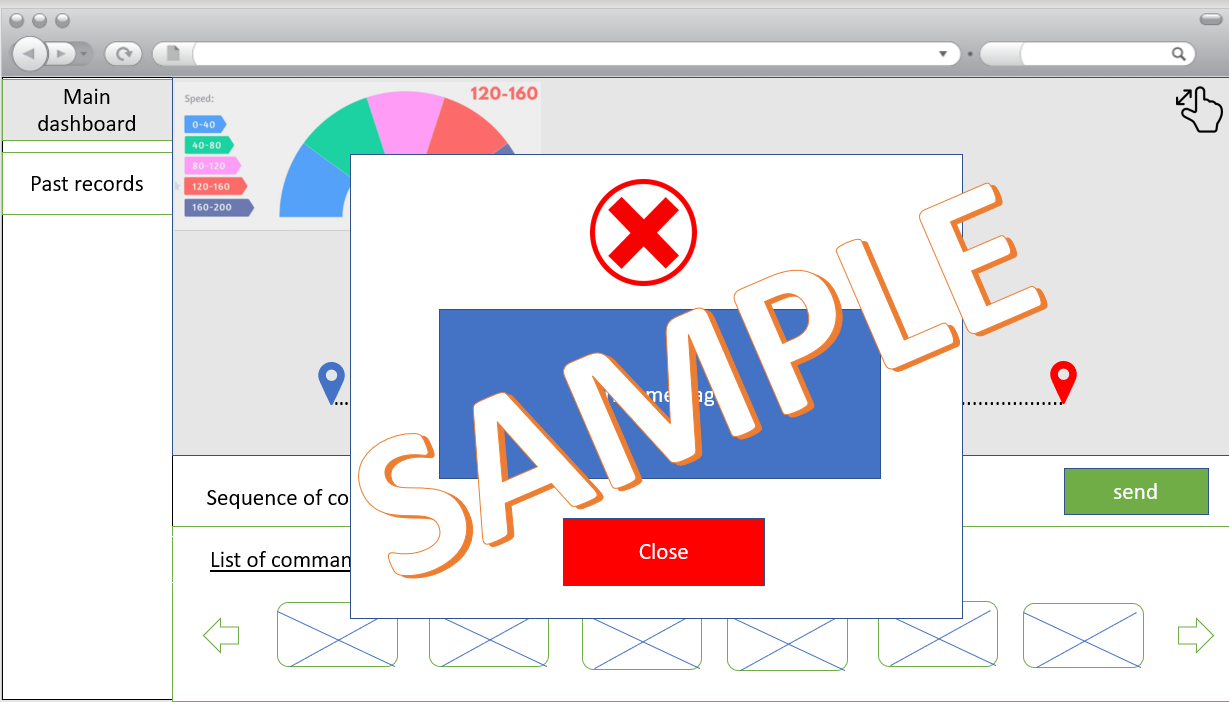


*Figure 2: Main dashboard consisting of students input commands and feedback*



*Figure 3: Main dashboard consisting of students input commands and feedback*

An alert will be displayed on the website to inform the students whether the website failed to send the commands to the robot or if there is a malfunction on the robot’s side. Finally, there will also be an indicator to show the robot’s connection status to the website.



*Figure 4: Alert*

### 3.1.2 Hardware Interfaces

Below is a list of hardware devices needed to interact with in order for Robot Car Website System to work

Laptops/ Desktop

These are hardware equipment needed for students to access the web portal. At the very least, it needs to be set up with an internet connection. It provides an interface for not only transferring of commands data to the car but also displaying car feedback to students.

ESP8266 WiFi module

The use of the WiFi module is to establish a connection between a web portal and a robot car. With the WiFi module, data such as commands can be sent from the web portal to the car and vice versa.

Motor

Both motor driver and Direct Current motor are needed to drive the car directions according to the commands selected by the students.

Sensors

Several sensors such as ultrasonic, IR line tracing, IR optical speed are used to detect and measure distance and speed. With distance and speed being measured, the car can send feedback to the web portal. There will be an interface to do basic reading on the current values detected by sensors.

MSP432 Launchpad

The microcontroller (MCU) will be used to control all the hardware interface (i.e. WiFi module, motor and sensors). MCU are programmed to directly connect all other sub modules and interact with the web portal.

### 3.1.3 Software Interfaces

The web portal can be supported by Windows operating system. Implemented using HTML and Javascript as the front-end, it allows more interactive web pages to be displayed. In addition, PHP will be used for the server-side scripting of the portal. Lastly, the programming on MSP432 to establish an interaction between the robot and the hardware component will be done in C.

## 3.2 Functional Requirements

The system consists of two parts, the webpage and the robot.

Webpage

* Student controls the robot via a board on the website by selecting the commands
* Website contains a board to show student’s chosen commands to be sent to the robot
* Student is able to add a command by clicking onto the command
* Student chosen command is added onto the board that contains selected commands
* Student is able to delete chosen inputs
* Users are able to view the robot's sensor inputs via a dashboard on the website
* Student is able to see feedback and status of the robot
* Student is able to play challenges set by the teacher
* Website contains a dashboard to view past command sequence sent to the robot
* Website to show connection status with robot
* Website must have a student and teacher mode
* Teachers shall be able to see student’s past command history and sensor data from the robot
* Teachers will be able to create digital maps for students to complete
* Website will have maps to show robot car in website moving in real time with the robot
* Website will have a board to show available challenges
* Website must have a basic tutorial mode for students

Robot

* Robot will be able to move left, right, up and down
* Robot will be able to connect to the webpage
* Robot will be able to send data to the webpage
* Robot will be able to move in direction sequence set by the student on the website
* Robot communicates with website through Wifi

## 3.3 Functional Requirements Validation

Client requirements are derived from client meetings and questionnaires. The above functional requirements are derived and stated based on client requirements. Below are the justifications.

The website will allow primary school students to learn critical problem solving and computation thinking skills as they will learn how to do programming by controlling the robot with the website through selecting direction commands to send to the robot. The student is also able to see a board on the website which contains the chosen commands to be sent to the robot, also as the robot is able to move in the direction sequence sent from the website, this shows that what is seen on the portal is reflected in real-life which is seeing that the robot moves according to the instructions sent to it.

Furthermore, since the student is able to add a command by clicking onto the command, delete chosen inputs and have chosen command is added onto the board that contains selected commands this shows that users are able to provide a list of commands to the robotic car. The data from the robotic car will be displayed on the dashboard as well which the teacher can view from all students’ profiles. The website will display the connection status of the robot and communicate with the robot through Wifi, which is the main communication mode will be through WiFi. The website will allow the user to choose if he is a student or teacher and if he is a student he will have to choose a student profile or name. The teacher will also be able to view data and history from all student profiles individually. Finally, The website will also have a basic tutorial mode for the students and they will be able to toggle the tutorial on or off which shows that there is an early tutorial that guides the student to step through the map.

## 3.4 Use Case Model

## 

[*https://drive.google.com/file/d/1tfqH3LXaXk6Qj09HHNr\_1zjG-FicXA7h/view?usp=sharing*](https://drive.google.com/file/d/1tfqH3LXaXk6Qj09HHNr_1zjG-FicXA7h/view?usp=sharing)

*Figure 5: Use Case Model*

### 

### 3.4.1. Use Case #1 - View list of sensor inputs **(Id: UC-1)**

| **Author** | Tiffany (Scrum Master) |
| --- | --- |
| **Purpose:** | UC-1 provides visuals for the student to see the different kinds of sensor inputs. |
| **Requirements Traceability:** | The student will only be able to see the sensor inputs if there is data sent from the robot car. |
| **Priority:** | This should be a high priority   * The robot has to communicate with the system and display in the GUI for the students to see |
| **Pre-conditions:** | * Sensor inputs has to be read and sent from the robot car |
| **Post conditions:** | * The student is able to see a list of data from the robot car in the GUI |
| **Actors:** | Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Web portal receives data from the robot car 2. Web portal displays the data to the specific section for the sensors to the students. |
| Alternate Flow   1. Web portal did not receive any data input due to system’s error    1. Web portal display errors about connection with the sensor inputs 2. Web portal connects to the sensors but the robot car is not moving    1. Web portal displays the value of the data with a warning |
| **Includes:** | UC-10 |
| **Notes/ Issues** | NA |

### 3.4.2. Use Case #2 - View history of commands used **(Id: UC-2)**

| **Author** | Tiffany (Scrum Master) |
| --- | --- |
| **Purpose:** | UC-2 allows students to view their history of commands used on the robot car. |
| **Requirements Traceability:** | The students will be able to see the history of commands if there are commands that have been used. |
| **Priority:** | This is a high priority.   * There is an involvement in the GUI and the students are able to see the recent commands they have used to control the robot car. |
| **Pre-conditions:** | NA |
| **Post conditions:** | The students will be able to see the history of commands used in an order. |
| **Actors:** | Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Student login to the student profile. 2. Web portal displays history of commands. |
| Alternate Flow   1. Web portal is unable to display a history of commands due to empty data    1. displays error message “No commands used” |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 3.4.3. Use Case #3 - View chosen commands **(Id: UC-3)**

| **Author** | Yu Hui (Microcontroller engineer) |
| --- | --- |
| **purpose:** | UC-3 displays the sensor commands in sequential order according to what the student has previously selected. The purpose is to allow them to view the selection they indicate to the robot car and understand how the robot performs for each command. |
| **Requirements Traceability:** | The requirement is that students are able to view their chosen commands of the robot movements. |
| **Priority:** | This is considered a medium priority.   * It is possible to communicate with the robot car and have it work as intended without having the student to view their selected commands on the web portal. However, that would not be interactive as students will probably forget the commands they choose. |
| **Pre-conditions:** | * Students have selected the robot car commands from the command list in the web portal * Student selection are temporarily stored in the web portal |
| **Post conditions:** | * Students will be able to view the commands they have selected. |
| **Actors:** | Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow  1. Web portal retrieves the student’s selected commands.  2. Web portal displays the student’s selected commands.  3. Students view their selected commands.  4. The use case end (success) |
| Alternate Flow:  2a. Web portal not displaying the retrieved student’s selected commands on the first try.  2ai. Display error message  2aii. Clear cache and reload the web portal.  2aiii. Go to UC-4. Repeat UC-4 main flow.  2aiv. Repeat UC-3 main flow. |
| Exception Flow:  1a. Web portal unable to retrieve student’s chosen commands due to system error.  1ai. Web portal shows error message and notify developers. Use case terminated.  2b. Web portal unable to display retrieved student’s selected commands after more than one try  2bi. Web portal display error message. System failed. Use case terminated. |
| **Includes:** | Not applicable |
| **Notes/ Issues** | UC-3 is not the same as storing the history of a student's inputs.   * UC-3 will not be considered as history of student’s inputs until the car has successfully executed all the student’s commands. |

### 3.4.4. Use Case #4 - Select commands **(Id: UC-4)**

| **Author** | Yu Hui (Microcontroller engineer) |
| --- | --- |
| **purpose:** | UC-4 allows students to make their own decision for the commands that they want the robot to execute. UC-8 is an extension from UC-4 where students can decide whether to execute the selected commands on the robot car or not. |
| **Requirements Traceability:** | The requirement is that students is able to select commands to control robots |
| **Priority:** | This has a high priority.   * Commands must be selected and indicated by the student. Without selecting the commands, no commands will be sent over for the robot car to function. |
| **Pre-conditions:** | * The web portal retrieves and displays a list of commands. |
| **Post conditions:** | * Student successfully selected the commands for the robot car. |
| **Actors:** | Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. The student selects the command from the list of commands. 2. The web portal temporarily stores the selected commands. 3. The web portal reflects the commands chosen 4. Student proceed commands   4ai. Students select to proceed with UC-8.  4aii. Web portal prompt confirmation to proceed  4aiii. Send the selected commands and process flow to UC-8. Continue from UC-8. |
| Alternative Flow:  4b. Student did not proceed with the commands |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 3.4.5. Use Case #5 - Delete chosen inputs **(Id: UC-5)**

| **Author** | Hong Ying (Microcontroller engineer) |
| --- | --- |
| **Purpose:** | UC-5 allows students to delete the input chosen. This allows students to remove unwanted activities previously input. |
| **Requirements Traceability:** | The requirement is that the student is able to view the chosen input. |
| **Priority:** | This is considered medium priority   * Without the delete of chosen input, students are unable to remove unwanted input selected previously. |
| **Pre-conditions:** | * Input had been selected by the student previously. |
| **Post conditions:** | * Unwanted input had been removed by the student. |
| **Actors:** | Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Web portal displayed a list of input selected by the student. 2. Students select input to be deleted. 3. Web portal prompt confirmation message. 4. Students accept the confirmation message. 5. Web portal update record in database. |
| Alternate Flow  4a. Students reject the confirmation message.  4ai. Back to main flow 1. |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 3.4.6. Use Case #6 - View connection status between website and robot **(Id: UC-6)**

| **Author** | Hong Ying (Microcontroller engineer) |
| --- | --- |
| **Purpose:** | UC – 6 displays the connection status between website and robot. This allows the student to know if the robot is ready to be triggered by the web portal. |
| **Requirements Traceability:** | The requirement is that the robot sends data to the web portal. |
| **Priority:** | This is considered medium priority   * Without the viewing of connection status, students will be unable to check if the connection issue is the reason causing the robot not to respond. |
| **Pre-conditions:** | * Robot must send data to the website. * Students have input commands to the web portal. |
| **Post conditions:** | * Connection between the student and robot had developed. |
| **Actors:** | * Student * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Web portal displays the connection status between the robot and the web portal. |
| Alternate Flow  1a. Loss connection between the robot and the web portal  1ai. Students will have to reconfigure the connection between the robot and the web portal.  1aii. Proceed to main flow 1. |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 3.4.7. Use Case #7 - See feedback and status **(Id: UC-7)**

| **Author** | Nicholas Lum ( Web Developer ) |
| --- | --- |
| **purpose:** | UC-7 |
| **Requirements Traceability:** | The requirement is that the student is able to see feedback and status of the robot. |
| **Priority:** | This should be considered Low Priority   * Even without this feature the function and purpose of the robot will still be met |
| **Pre-conditions:** | * Robot must have connection to the website * Robot must send data to the website |
| **Post conditions:** | * Data and Feedback is sent to the website and shown on the webpage |
| **Actors:** | * Student * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. Student clicks to see the data and feedback from the robot 2. Feedback and data is sent to the webpage from the robot 3. Feedback and data is shown to the student. |
| Alternate Flow:  2.a Feedback and data failed to send to the webpage due to  connection error  2.a.i Student will see that there is a connection error on the  webpage  3.a Feedback and data is not shown to the student  3.a.i Student will see that there is a connection error on the  webpage |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.8. Use Case #8 - Send command **(Id: UC-8)**

| **Author** | Nicholas Lum (Web Developer) |
| --- | --- |
| **purpose:** | UC-8 displays the sending of commands to the robot after the user has selected the commands and confirms the commands. |
| **Requirements Traceability:** | The requirement is that the command needs to be sent to the robot in order for the robot to move. |
| **Priority:** | This should be considered High Priority   * The student will not be able to interact with the robot if the commands aren’t sent to the robot |
| **Pre-conditions:** | * Student must have selected commands and have decided to send them * There must be a connection between the website and the robot |
| **Post conditions:** | * Robot will receive the commands that has been sent by the user through the website |
| **Actors:** | * Student * Robot |
| **Extends:** | Select Command (UC-4) |
| **Flow of events:** | Main Flow:   1. Command is received from student 2. Command is sent to Robot 3. User gets confirmation that command is sent successfully |
| Alternate Flow:  3.a. Connection is unstable between website and Robot  3.a.1 Student gets an error saying that the command is not sent due to connectivity issue |
| **Includes:** | Detect movement commands ( UC-9 ) |
| **Notes/ Issues** | NA |

### 

### 

### 3.4.9. Use Case #9 - Detect movement commands **(Id: UC-9)**

| **Author** | Alaric (Web Developer) |
| --- | --- |
| **purpose:** | **UC-9** displays that the robot is able to detect the movement commands the user sends from the website. |
| **Requirements Traceability:** | This traces back to the requirement of the student being able to control the robot via the website as without being able to detect the movement commands from the robot, the student will not be able to control the robot via the website. |
| **Priority:** | This should be **High** priority   * If the robot is unable to detect movement commands sent from the user sending the commands from the website the robot will not be able to move. |
| **Pre-conditions:** | * The user must send the commands chosen on the website’s graphical user interface * Student must click the send button * Robot and website must be connected |
| **Post conditions:** | * Robot must move according to the student’s chosen commands |
| **Actors:** | * Student * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Movement commands are received from the student sending from the website 2. Robot processes the commands 3. Robot moves |
| Alternate Flow:  2.a Robot fails to process the command  2.a.1. Robot sends the type of error that has occurred to  the user  3.a Robot faces mechanical failure  3.a.1 Robot sends a notification to website to let user know  that it failed to move |
| **Includes:** | Send Command (UC-8) |
| **Notes/ Issues** | NA |

### 

### 3.4.10. Use Case #10 - Send sensor data **(Id: UC-10)**

| **Author** | Alaric (Web Developer) |
| --- | --- |
| **purpose:** | **UC-10** displays that robot will send data that it has detected from its sensors to the website |
| **Requirements Traceability:** | This traces back to the requirement of the student being able to view the robot’s sensor inputs via a dashboard on the website and the robot will be able to send data to the webpage. This is because if the robot is not able to send the sensor data, then the website will not be able to display the robot’s sensor data. |
| **Priority:** | This should be medium priority   * If the robot fails to send the data to the website, the robot will still be able to move regardless * This is still important as it is a requirement of the client to enhance the user’s learning experience |
| **Pre-conditions:** | * Robot must be moving to send data to the website |
| **Post conditions:** | * The website will be display the data sent by the robot and be displayed on the dashboard in a logs format |
| **Actors:** | * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Robot will pick up data with its sensors 2. Robot sends the data to the website |
| Alternate Flow:  1.a Robot’s sensors are not working  1.a.1 Website’s dashboard will not present new data from  the robot  2.a Robot fails to send data to website  2.a.1 Website’s dashboard will not present new data from  the robot |
| **Includes:** | UC-9 |
| **Notes/ Issues** | NA |

### 

### 3.4.11. Use Case #11 - Create map challenges **(Id: UC-11)**

| **Author** | Nicholas Lum ( Web Developer ) |
| --- | --- |
| **purpose:** | UC-11 is to allow teachers to create map challenges for the students to attempt. |
| **Requirements Traceability:** | Teachers will be able to create digital maps for students to complete |
| **Priority:** | This should be considered Medium Priority   * This is needed so that the students have challenges to attempt, but not as urgent as getting the website and robot to work. |
| **Pre-conditions:** | * User must be a teacher |
| **Post conditions:** | * Map challenge is created and stored on the website |
| **Actors:** | * Teacher |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. Teacher selects the create map challenge button 2. Teacher creates map challenge 3. Map challenge is created and stored for student to attempt |
| Alternate Flow:  3.a Website lost connection due to connection issue, map challenge is not saved  3.a.i Teacher will see that their map challenge was not saved and will be prompted to try again |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.12. Use Case #12 - View basic tutorial mode**(Id: UC-12)**

| **Author** | Alaric ( Web Developer ) |
| --- | --- |
| **purpose:** | **UC-12**  allows student to learn by viewing the tutorial mode. |
| **Requirements Traceability:** | The requirement that the website needs to have a tutorial mode for students. |
| **Priority:** | This should be considered Low Priority   * Even without this feature the key function which is where the robot can be controlled from the website is not affected |
| **Pre-conditions:** | * Student must choose to view the tutorial mode |
| **Post conditions:** | * Student will learn the basics of how to control the robot through the website. |
| **Actors:** | * Student |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. Student clicks the tutorial option 2. Website will show the students the different types of dashboards 3. Website will show what do the buttons on the website will do |
| Alternate Flow: |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.13. Use Case #13 - Play map challenge **(Id: UC-13)**

| **Author** | Alaric ( Web Developer ) |
| --- | --- |
| **purpose:** | UC-13 is to allow Students to play the map challenges set by the Teacher. |
| **Requirements Traceability:** | This ties to the requirement that Teacher are able to set map challenges for students to complete |
| **Priority:** | This should be considered Medium Priority   * This is because even without this feature, the robot can still be controlled by the website, however with this feature, it allows students to be able to learn critical thinking and computation problem solving skills at a much faster rate. |
| **Pre-conditions:** | * Map Challenges need to be posted by teacher * Connection status between robot and website need to be up * Website needs to be able to control the robot |
| **Post conditions:** | * Student either completes or fails the challenge |
| **Actors:** | * Student * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. Students select which challenge to try. 2. System will display the maze that is to be completed 3. Student will set the command sequence to complete the maze 4. The system will check the next block from robot’s current position on the maze if it is red or white 5. The system will check the next block until it reaches the green block 6. If the robot on the maze reaches the green block, the game ends and the student succeeds the challenge |
| Alternate Flow:  4.a System detects next block is a red block  4.a.i The website will continue checking the commands for the robot.  4.b System detects next block is a white block  4.b.i The website will move the virtual car and notify the user that they hit a wall.  4.b.ii System will end the challenge  6.a Robot fails to process the command  6.a.i Robot sends the type of error that has occurred to  the user  6.a.ii System ends the challenge |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.14. Use Case #14 - See feedback and status **(Id: UC-14)**

| **Author** | Tiffany ( Scrum Master ) |
| --- | --- |
| **purpose:** | UC-14 allows the teacher to be able to see the feedback and the status of the robot. |
| **Requirements Traceability:** | The requirement is that the robot is able to send the feedback and status. |
| **Priority:** | This is considered Low Priority   * As without being able to see the feedback and status of the robot, the function and purpose of the robot will still be met. |
| **Pre-conditions:** | * Robot must have connection to the website * Robot must send data to the website |
| **Post conditions:** | * Data and Feedback is sent to the website and shown on the webpage |
| **Actors:** | * Teacher * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. Teacher clicks to see the data and feedback from the robot 2. Feedback and data is sent to the webpage from the robot 3. Feedback and data is shown to the Teacher. |
| Alternate Flow:  2.a Feedback and data failed to send to the webpage due to  connection error  2.a.i Teacher will see that there is a connection error on the  webpage  3.a Feedback and data is not shown to the teacher  3.a.i Teacher will see that there is a connection error on the  webpage |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 3.4.15. Use Case #15 - View connection status between website and robot **(Id: UC-15)**

| **Author** | Tiffany (Scrum Master) |
| --- | --- |
| **purpose:** | UC-15 enables the teacher to see the connection status between the website and the robot |
| **Requirements Traceability:** | The requirement is that the robot and website connection is able to communicate. |
| **Priority:** | Medium priority   * If the teachers are unable to view connection status, they will be unable to check if the connection issue is the reason causing the robot not to respond. |
| **Pre-conditions:** | * Robot must send data to the website. * Teachers have input commands to the web portal. |
| **Post conditions:** | * Connection between the website and robot had developed. |
| **Actors:** | * Teachers * Robot |
| **Extends:** | NA |
| **Flow of events:** | Main Flow   1. Web portal displays the connection status between the robot and the web portal. |
| Alternate Flow  1.a Loss connection between the robot and the web portal  1ai. Teachers will have to reconfigure the connection between the robot and the web portal.  1aii. Proceed to main flow 1. |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.16. Use Case #16 - View History of Command Use Per Challenge Per User **(Id: UC-16)**

| **Author** | Hong Ying (Microcontroller engineer) |
| --- | --- |
| **purpose:** | UC-16 allows teachers to view the command completed by students for the specific challenge. |
| **Requirements Traceability:** | The requirement is that the teacher is able to view the student name and challenges completed. |
| **Priority:** | This is considered medium priority   * Without the viewing of the command completed, the teacher will be unable to track and monitor the learning outcome of the student. |
| **Pre-conditions:** | Teacher had logged in and the student had completed at least one challenge. |
| **Post conditions:** | Teacher view command of past challenges complete by student |
| **Actors:** | Teacher |
| **Extends:** | NA |
| **Flow of events:** | Main Flow:   1. System displayed a list of student names. 2. Teacher selects the student that he/she wants to view. 3. System displayed list of challenges completed by the student. 4. Teacher selects the challenge he/she wants to view. 5. System displayed the steps that the student does to complete the challenge. |
| Alternate Flow:  3a. Student did not complete any challenge.  3ai. Students displayed “No Completed Challenge” |
| **Includes:** | NA |
| **Notes/ Issues** | NA |

### 

### 3.4.17. Use Case #17 - View list of sensor inputs **(Id: UC-17)**

| **Author** | Yu Hui (Microcontroller engineer) |
| --- | --- |
| **purpose:** | UC-17 allows the teachers to view a list of sensor inputs of each student so that they can use the data for testing and troubleshooting. The purpose is to check whether the sensors are working. In the event the sensors are not working, necessary technical support can be made. |
| **Requirements Traceability:** | Ties to the requirement that users are able to view robot’s sensor data via a dashboard on the website |
| **Priority:** | This is considered a high priority   * Without viewing the list of sensor inputs, teachers are unable to send sensor data (UC-10) to test whether the robot is working, making troubleshooting difficult. |
| **Pre-conditions:** | The user must be a teacher. |
| **Post conditions:** | Teachers can view a list of sensor inputs of each student |
| **Actors:** | Teacher |
| **Extends:** | NA |
| **Flow of events:** | Main flow   1. Teacher will select which student’s data (sensor inputs) to view 2. System will retrieves and show sensor inputs data of the selected student |
| Alternate flow  2a. If system is unable to retrieve and show sensor inputs data of selected student  2ai. system prompt error message |
| **Includes:** | Send sensor data (UC-10) |
| **Notes/ Issues** | Testing the student program (commands) but not exactly playing it |

# 4. Other Non-functional Requirements

## 4.1. Performance Requirements

* The robot shall be able to respond to the student's commands within 2 seconds.
* UI should be intuitive with fewer words and more graphics so that students can use the system easily.
* UI should be responsive and cater to multiple desktop screen resolution size

## 4.2. Safety and Security Requirements

* The robot should not be able to be controlled by anything but the students using the website to send commands to the robot
* The website should prevent SQL injection
* The robot should be properly connected and have the proper grounding to not shock the students
* The robot shall not hit the students by sensing objects in front and would stop at an appropriate distance.

## 4.3. Software Quality Attributes

### 4.3.1 Security

The non-functional requirements stated are able to achieve security. The web portal created aims to prevent SQL injection by validating the teacher inputs during the creation of questions. By doing so, it prevents manipulating the database and gaining access to potentially valuable information. The validation can be done through the use of parameterized queries instead of concatenating the input within the query.

### 4.3.2 Usability

The software also aims to provide usability as the targeted audience are primary students. The web portal aims to have an ease of usage by providing simple navigation and creating more graphics to attract the students attention. By doing so, it will be more engaging and easy for the primary school students to learn and use the application.

## 4.4 Non-functional Requirements Validation

The non-functional requirements listed above are based on the client requirements. First and foremost, the client requires some form of safety features to be incorporated in the robot so as to minimise student injuries. To ensure that, the robot will be programmed such that the interface of the robot’s parts are properly connected to prevent shocks. Next, the robot can also sense the object in front which will stop at an appropriate distance. This prevents it from hitting the students. Finally, the requirement for the UI to be simple and straight-forward is being considered as well. For instance, the website is responsive which allows for multiple desktop screen resolution sizes. This allows flexibility in using the website. Furthermore, with fewer words and more graphics, it makes it simpler and intuitive for the students to use. Hence, satisfy the client requirement of making UI straightforward and easy to use for students.

# 5. Architectural and Detailed Design

## 5.1. Class Diagram

### 5.1.1. Problem statement

The website must have two types of profiles, one for students and one for teachers. Each student profile will be made by the backend and in order to use that profile, the student’s name and register number are entered into the website, which will be checked in the database to ensure it is a valid student. Students are able to control the robot movements by selecting commands on a dashboard of commands that is on the website’s interface. The website will have a dashboard that shows the history of the currently selected student profile chosen commands. The website will present data received from the robot’s sensors on a dashboard in a logs format. When the student clicks on a command on the board, it will add that command on the board of selected commands. The website will show the sequence of commands and the student is also able to modify the commands before it is sent out to the robot. The website must show the connection status of the robot and will show an alert if the website receives an error signal from the robot. The teacher will be able to select and see the selected student’s history of commands and the sensor data that was received. Teachers are able to create maps for students to complete which will be shown in a board of challenges on the student’s profile. The student will be able to see the robot move in real time with the robot moving in the map on the website. Website will provide a tutorial mode for the student to use.

\* underline words are nouns

### 5.1.2. Identifying Candidate classes

Nouns identified = website, student, teachers, profile, backend, name, register, number, database, robot, commands, interface, dashboard, history, sensors, logs, sequence, connection, status, error, signal, maps, challenges, tutorial, mode

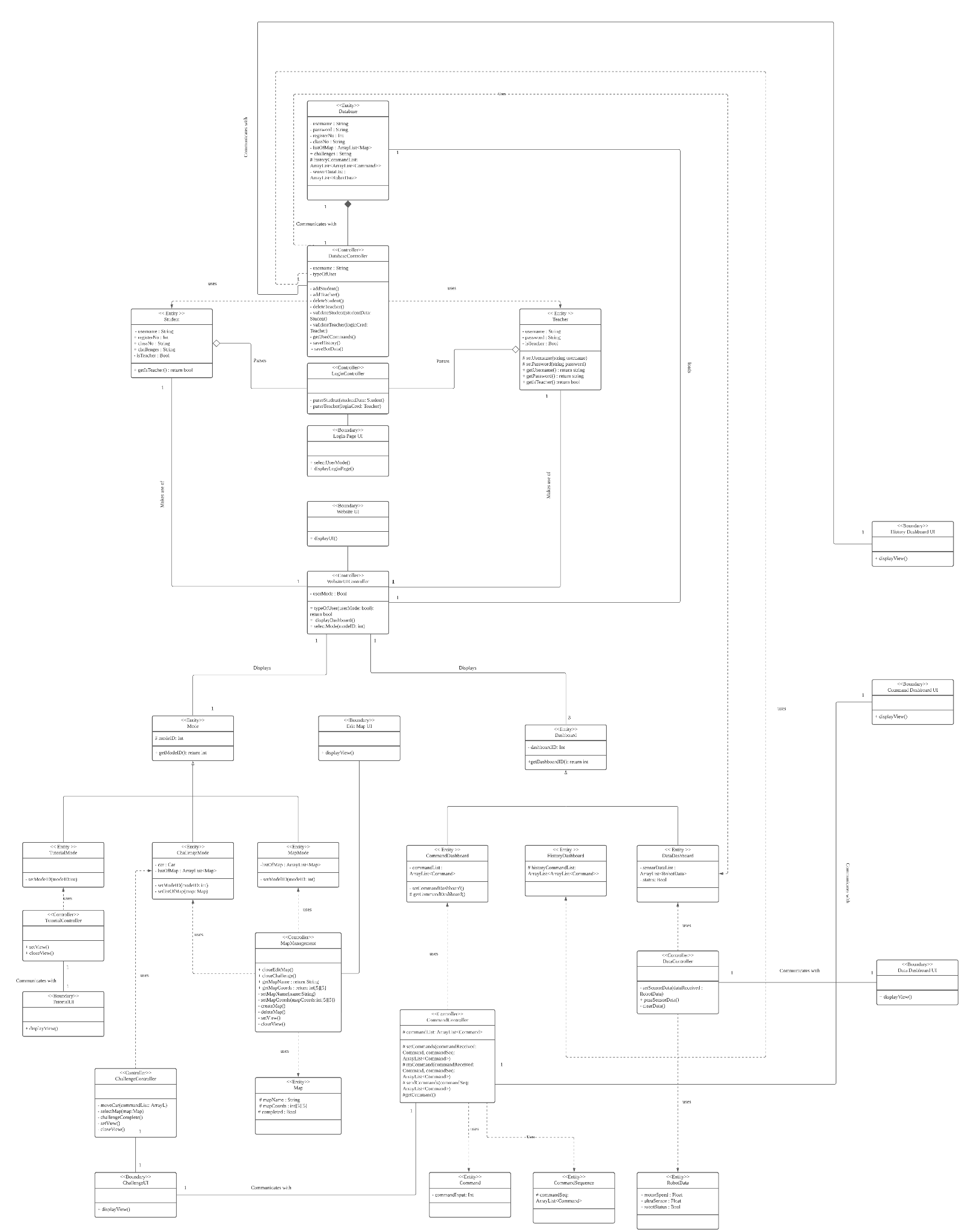
Candidate classes identified = profile, database, dashboard, interface, mode

### 

### 5.1.3. Identifying Subclasses

From the candidate classes identified, the team is able to then derive the subclasses for the system.

Subclasses identified = student profile, teacher profile, history dashboard, sensor dashboard , logs dashboard, command dashboard, challenge mode, tutorial mode, map mode.



<https://drive.google.com/file/d/1esF_nnxmUGG0-TQu4nWz5_lIqzUznUKt/view?usp=sharing>

*Figure 6: Class diagram*

| << Controller >> | |
| --- | --- |
| DatabaseController | * Adds and deletes users * Validates login data from student and teacher after login data has been parsed * Gets command history * Saves history and bot data to database |
| LoginController | * Parses login data from student and teacher |
| WebsiteUIController | * Displays the webpage to the user according to which user they are |
| TutorialController | * Updates TutorialUI |
| MapManagement | * Updates Edit Map UI * Create new maps * Delete existing maps |
| CommandController | * Add command to command list * Remove command from command list * Send command to robot |
| DataController | * Get sensor data from robot * Pass sensor data to data dashboard UI * Clear data |
| ChallengeController | * Moves virtual car in the webpage * Selects map * Attempts challenge |

| << Entity >> | |
| --- | --- |
| Database | * Stores login credentials of both Teacher and Student * Stores maps * Stores history command list * Stores robot sensor data * Stores challenge completion |
| Student | * Stores student login credentials * Stores challenge completion * Stores user mode |
| Teacher | * Stores teacher login credentials * Stores user mode |
| Mode | * Stores mode ID |
| TutorialMode | * An entity mode that separates itself from other modes |
| ChallengeMode | * Stores car object * Stores list of map challenges |
| MapMode | * Stores list of maps made |
| Dashboard | * Stores dashboard ID |
| CommandDashboard | * Stores a list of commands |
| HistoryDashboard | * Stores a list of history of commands |
| DataDashboard | * Stores robot sensor data * Stores robot status |
| Map | * Stores map name * Stores map coordinates * Stores whether map is completed |
| Command | * Stores command input |
| CommandSequence | * Stores a list of command sequence |
| RobotData | * Stores robot sensor data |

| << Boundary >> | |
| --- | --- |
| Login Page UI | * Displays Login page |
| Website UI | * Displays Website UI |
| Tutorial UI | * Displays Tutorial page |
| Challenge UI | * Displays Challenge page |
| Data Dashboard UI | * Displays Data Dashboard UI |
| Command Dashboard UI | * Displays Command Dashboard UI |
| Edit Map UI | * Displays Edit Map page |
| History Dashboard UI | * Display History Dashboard UI |

## 

## 5.2. Component Diagram

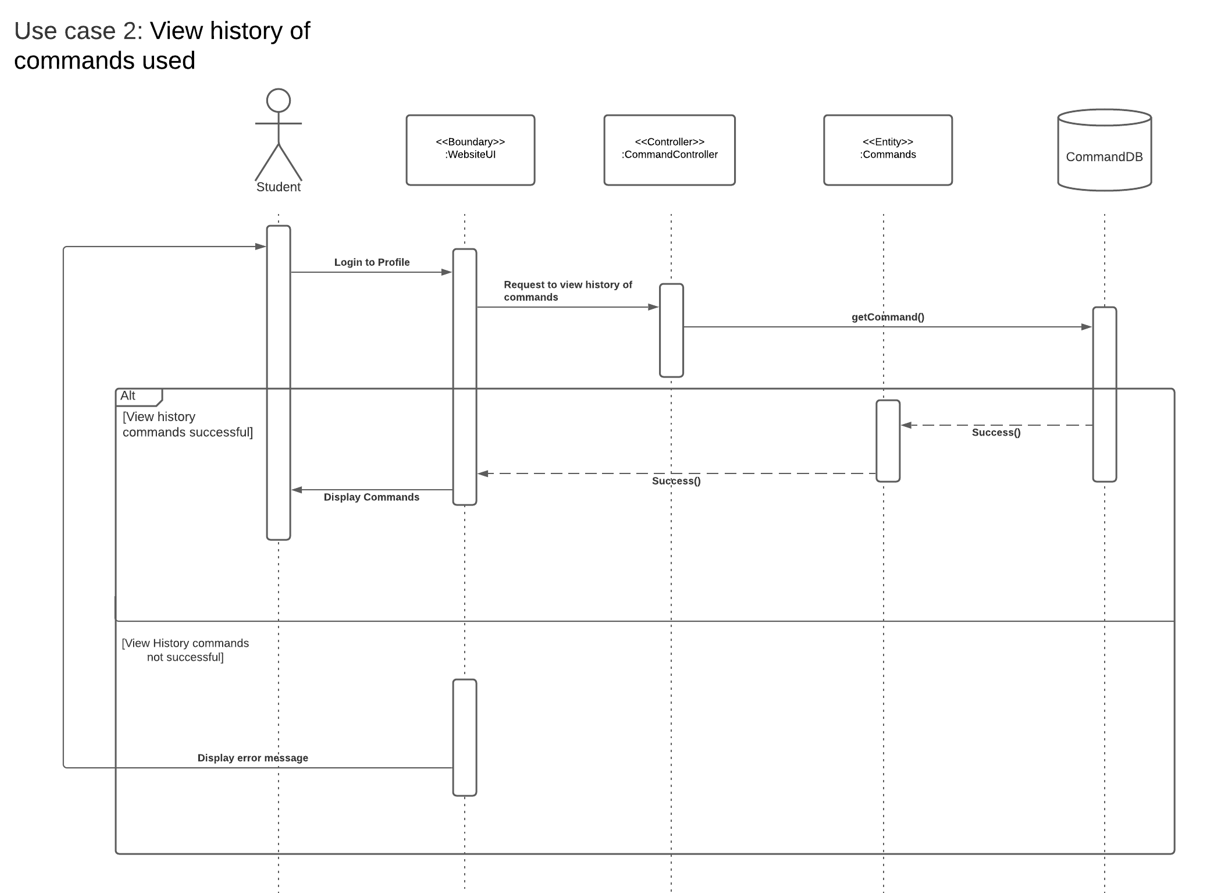
## 

[*https://lucid.app/lucidchart/d5ac0fc2-8017-415a-84dc-d5a4100f738b/edit?invitationId=inv\_01902d8c-2e39-49fc-9e63-38b4b95905b3*](https://lucid.app/lucidchart/d5ac0fc2-8017-415a-84dc-d5a4100f738b/edit?invitationId=inv_01902d8c-2e39-49fc-9e63-38b4b95905b3)

*Figure 7: Component diagram*

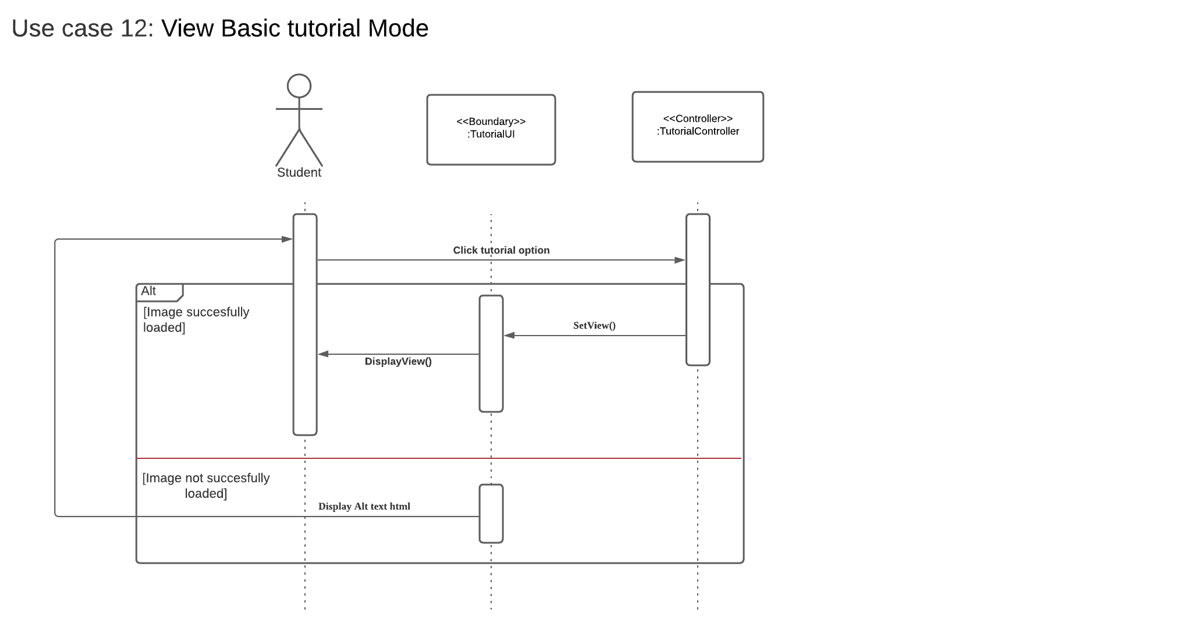
After developing the class diagram, the classes can be grouped into 4 main components. The Account Management component will have two subcomponents which are the Student and Teacher subcomponents as the Mode component will provide an interface with these subcomponents differently. The Mode component will need to have 2 other subcomponents that require an interface with the Database component, ChallengeMode and MapMode in order to save their respective data to the database also the mode component will provide an interface for other or new components to interact with the Mode component by using the iTutorial, iChallenge and iMap interface. Also, there is a Dashboard component which requires an interface by the Database component in order to load and save data to the Database component and it provides interfaces iCommand, iSensor and iHistory to interact with other or new components in the system. Finally, the database component will provide an interface to the 3 other components as they require to access the database in the database component.

## 5.3. Sequence Diagram



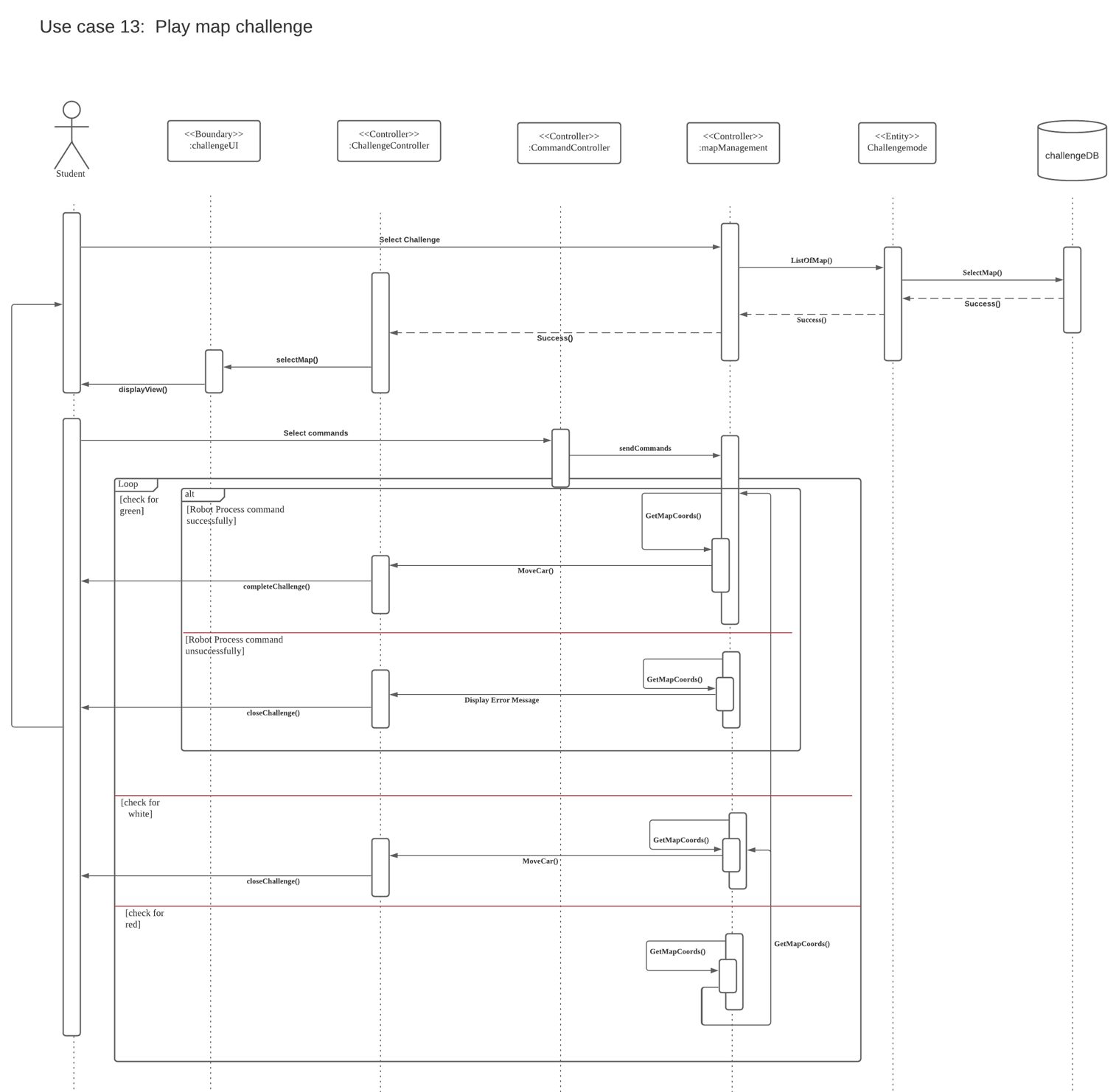
<https://drive.google.com/file/d/1Pkm8cAPw8l86-TyVPqo4Mye2_aLAzmz7/view?usp=sharing>

*Figure 8: Sequence diagram for Use Case 2*



<https://drive.google.com/file/d/1Pkm8cAPw8l86-TyVPqo4Mye2_aLAzmz7/view?usp=sharing>

*Figure 9: Sequence diagram for Use Case 12*



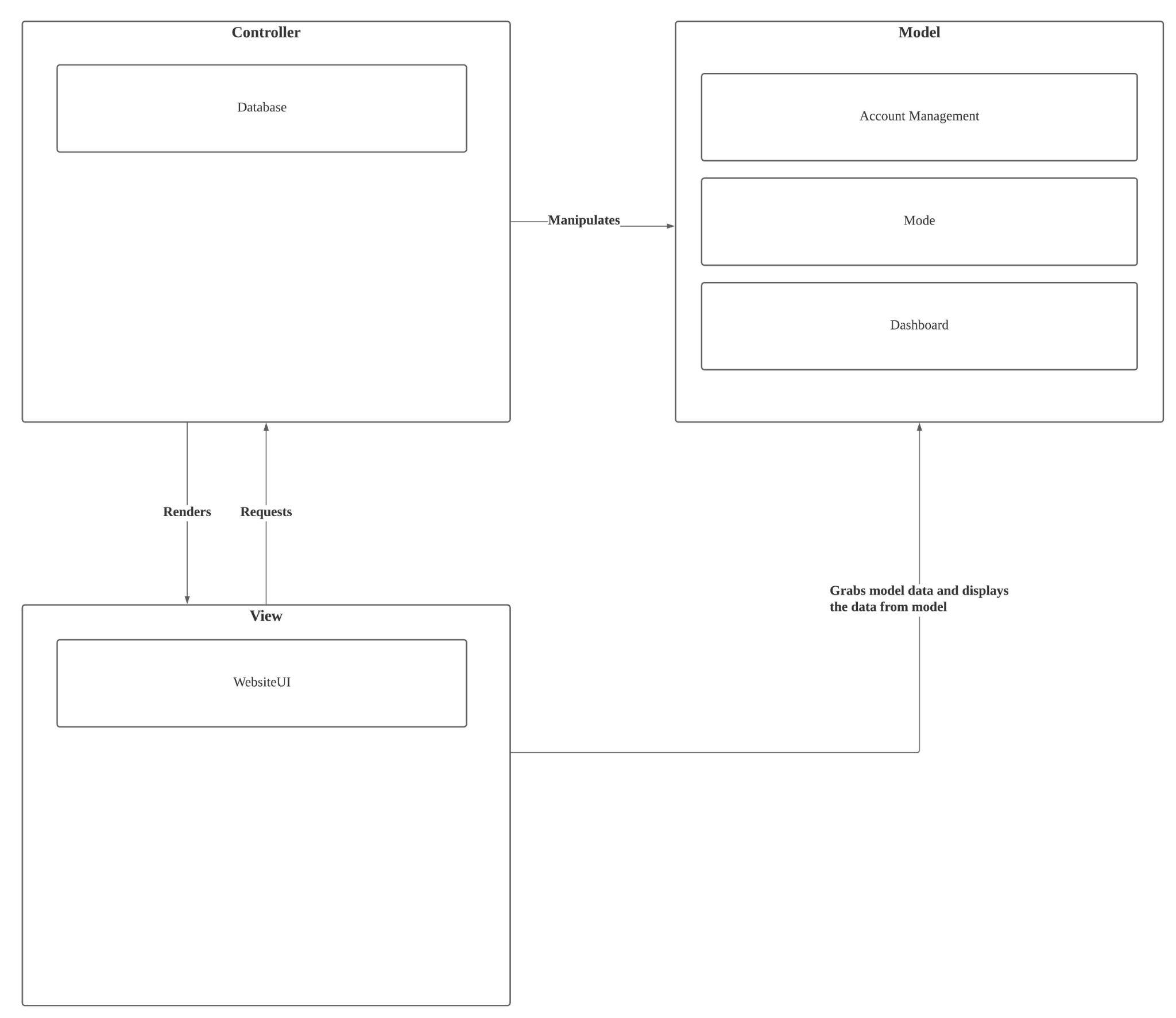
<https://drive.google.com/file/d/1Pkm8cAPw8l86-TyVPqo4Mye2_aLAzmz7/view?usp=sharing>

*Figure 10: Sequence diagram for Use Case 13*

## 

## 5.4. Software Architecture and Design Pattern

### 5.4.1. Software Architecture

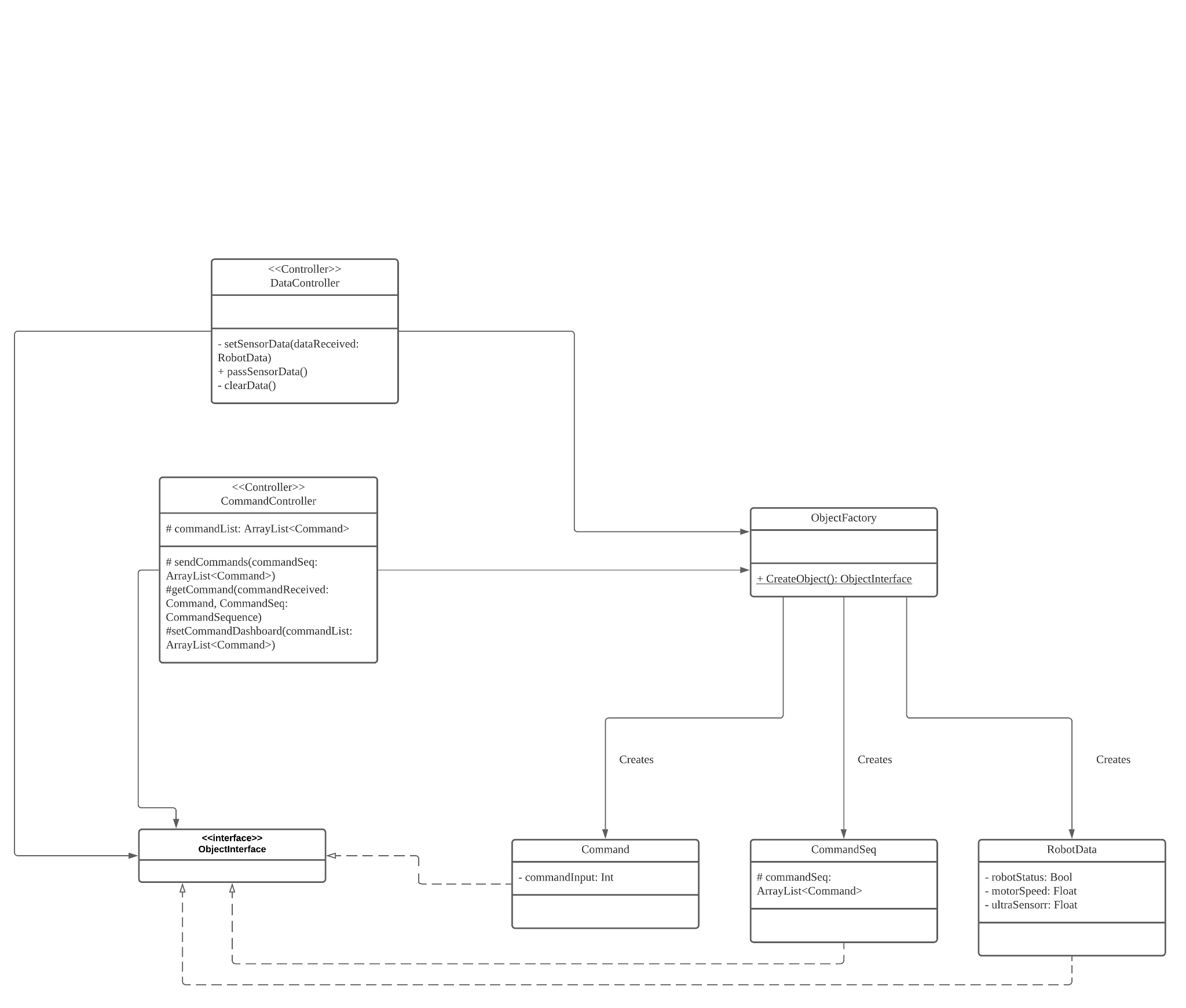


[*https://lucid.app/lucidchart/d5ac0fc2-8017-415a-84dc-d5a4100f738b/edit?invitationId=inv\_01902d8c-2e39-49fc-9e63-38b4b95905b3*](https://lucid.app/lucidchart/d5ac0fc2-8017-415a-84dc-d5a4100f738b/edit?invitationId=inv_01902d8c-2e39-49fc-9e63-38b4b95905b3)

*Figure 11: Model View Controller*

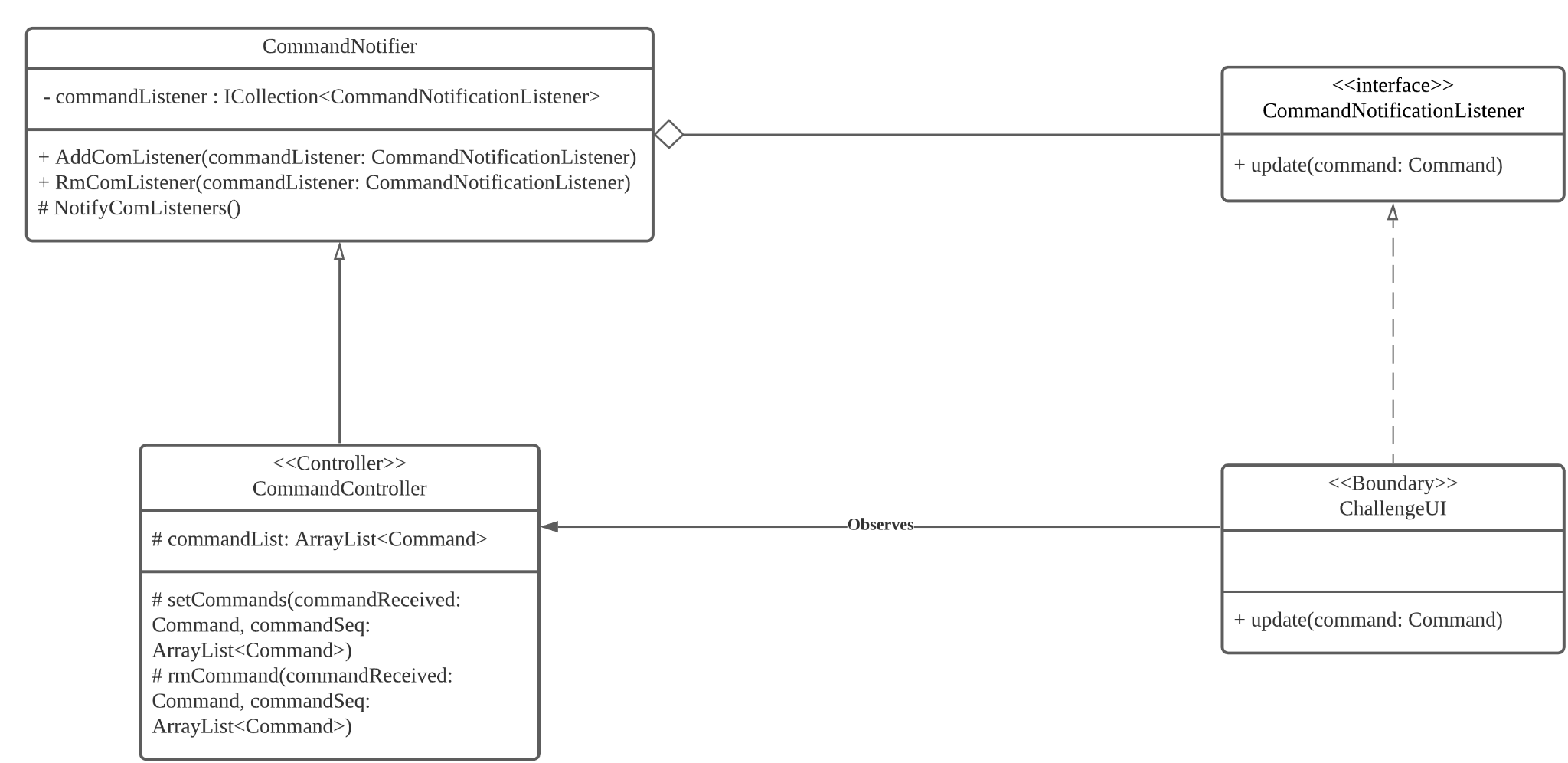
The MVC architecture was selected as the software architecture pattern as it supports concurrency in development. Being loosely-coupled, developers are able to work separately on the components. For instance, both development of the front-end (View) and the back-end (controller) can occur simultaneously. Furthermore, it allows a more maintainable code as if any changes made to a component will not affect the other components such as a view can be easily added with the architecture without affecting other parts of the software. With the usage of the Controller component, grouping of actions can be done to manipulate the data for multiple models. Therefore, MVC architecture allows greater flexibility in the development of the software.

### 5.4.2. Software Design Pattern



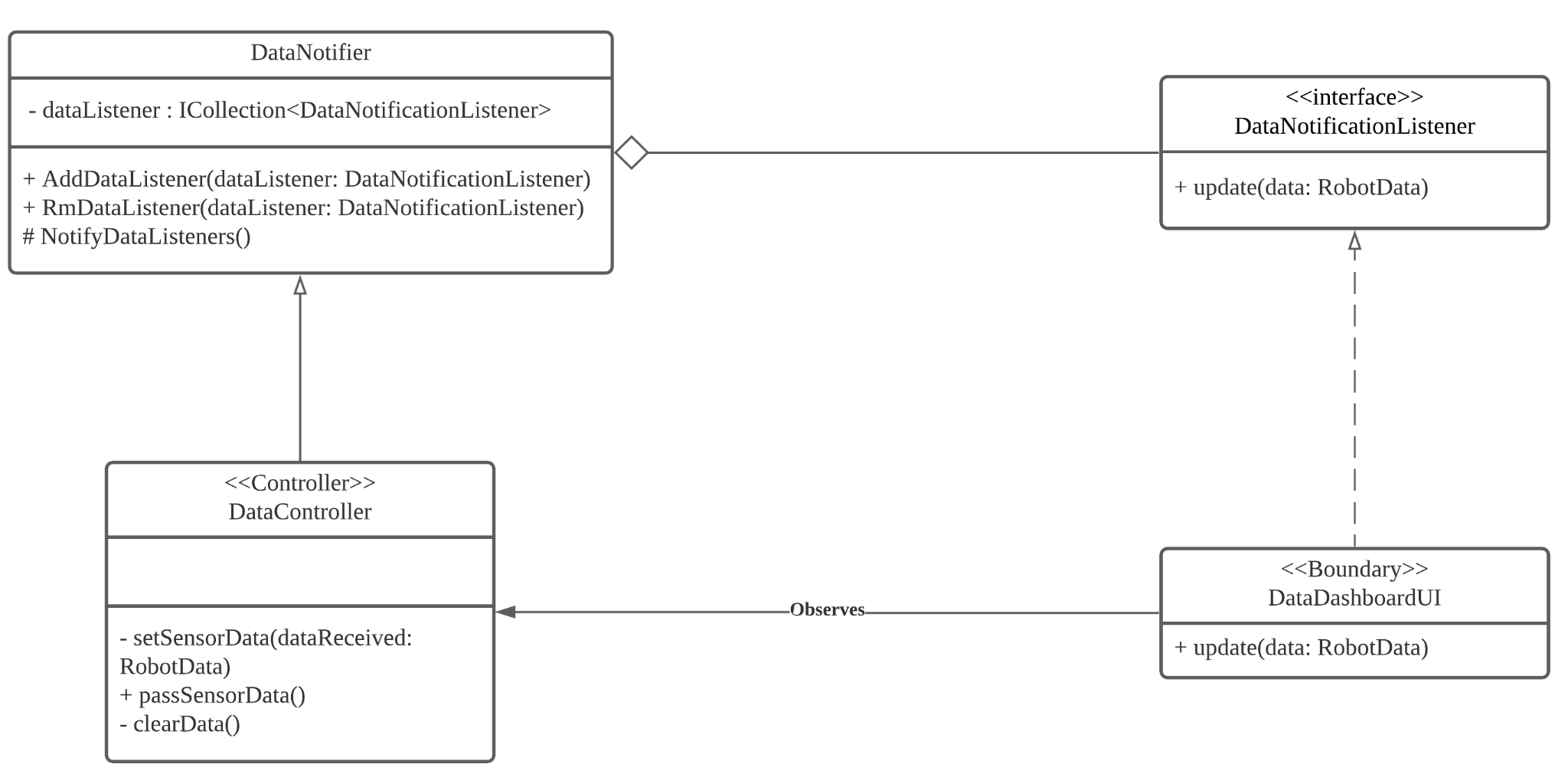
[*https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv\_43504dc4-64dd-4657-81fa-a59080c65f8e*](https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv_43504dc4-64dd-4657-81fa-a59080c65f8e)

*Figure 12: Factory Design Pattern to create instances of Command, CommandSeq, and RobotData*



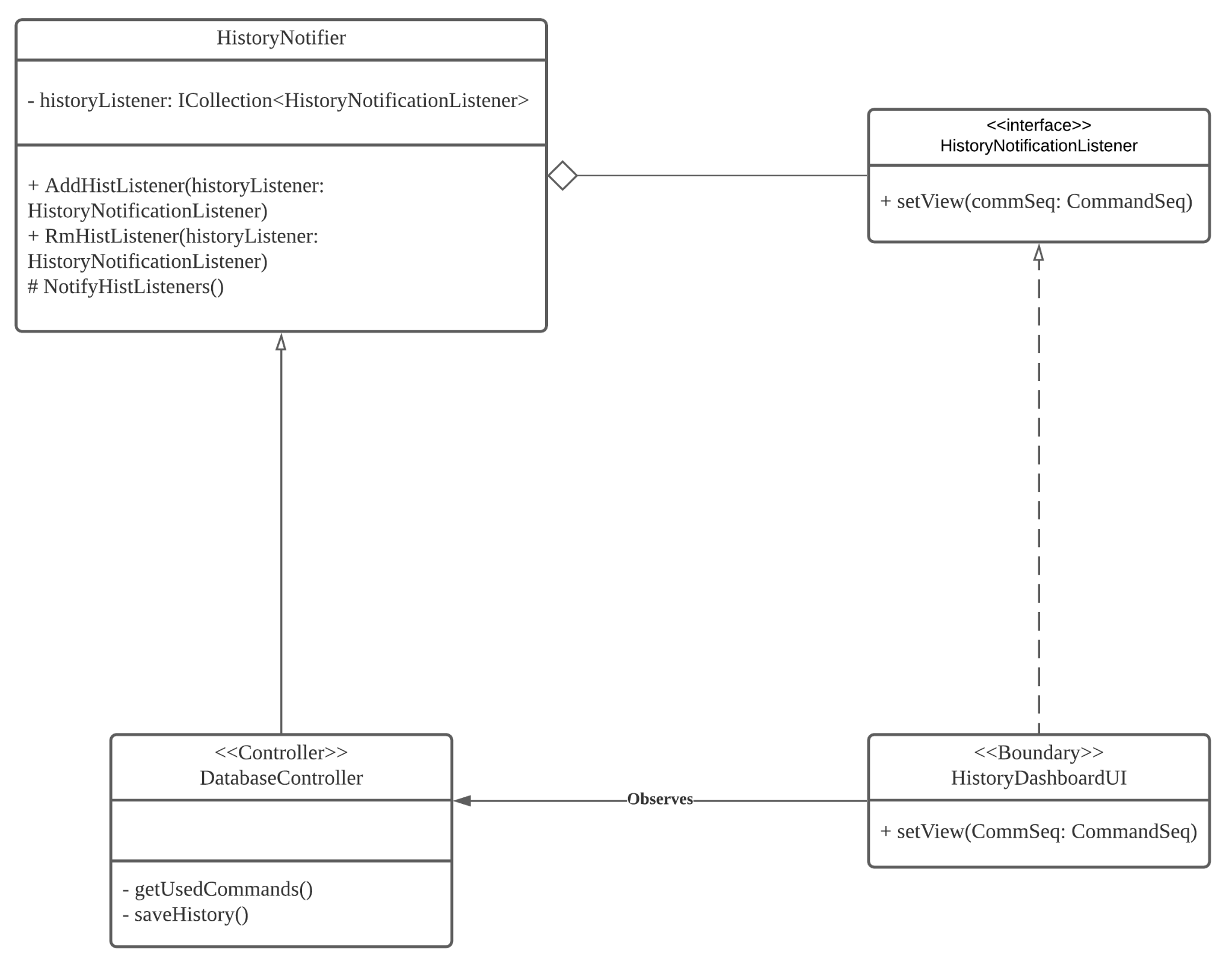
[*https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv\_43504dc4-64dd-4657-81fa-a59080c65f8e*](https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv_43504dc4-64dd-4657-81fa-a59080c65f8e)

*Figure 13: Observer Design Pattern to update ChallengeUI when a command sequence is modified*

**

[*https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv\_43504dc4-64dd-4657-81fa-a59080c65f8e*](https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv_43504dc4-64dd-4657-81fa-a59080c65f8e)

*Figure 14: Observer Design Pattern to update DataDashBoardUI when a data is received from Robot Car*



[*https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv\_43504dc4-64dd-4657-81fa-a59080c65f8e*](https://lucid.app/lucidchart/0d93fc62-0ded-4e34-8f5a-b62eb5239a32/edit?invitationId=inv_43504dc4-64dd-4657-81fa-a59080c65f8e)

*Figure 15: Observer Design Pattern to update HistoryDashboardUI when a a command sequence is sent to robot*

Factory Design Pattern (objectFactory with static method CreateObject) is used to create instances of the classes at runtime. Such pattern is applied to facilitate the followings scenarios:

* To instantiate RobotData with the data of robot sensor data sent to the webpage from the Robot Car.
* To create a Command Object when a command is entered.
* To create an ArrayList<Command> to be added to store the sequence of Commands with and ArrayList of Command instances.

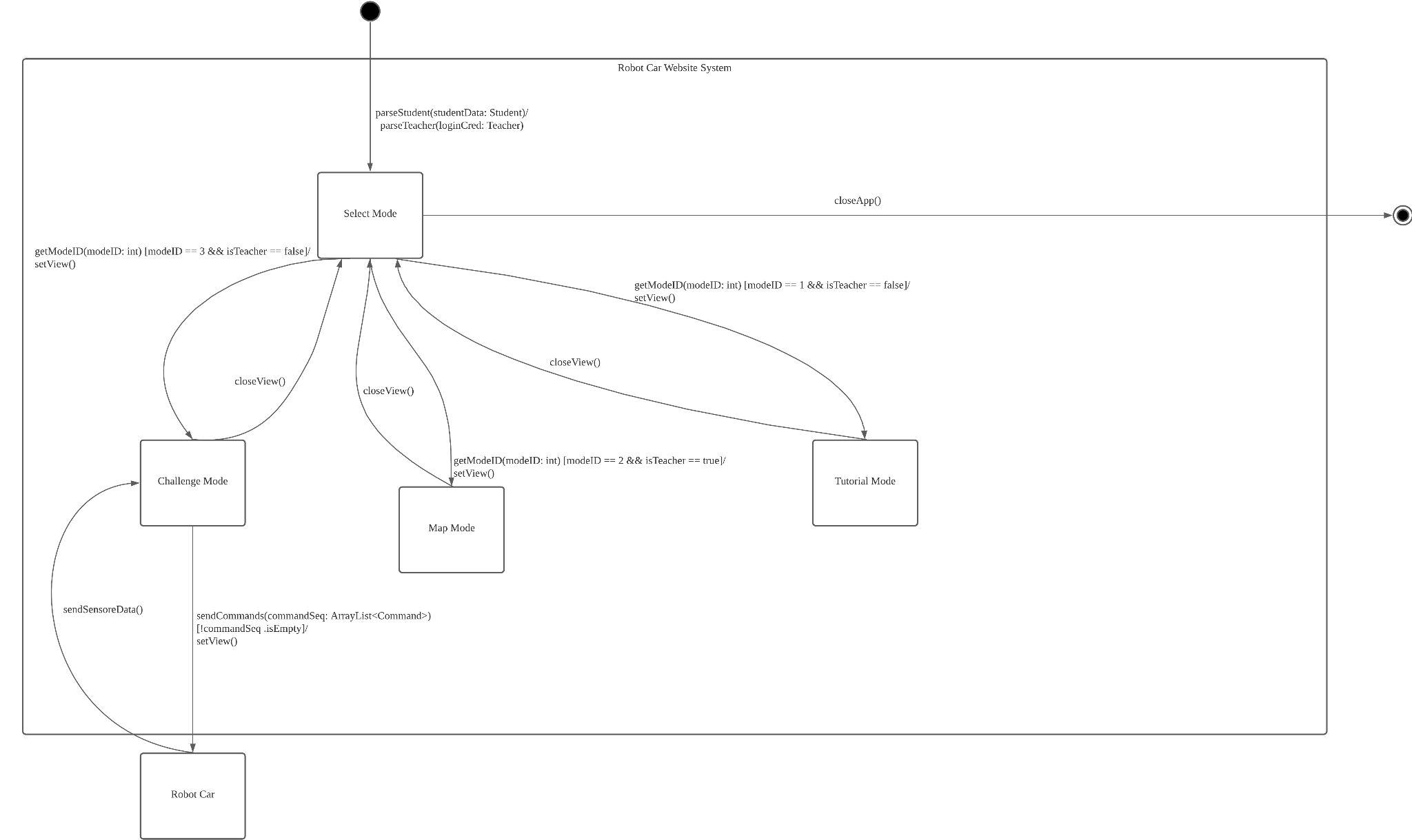
Next, the Observer Design Pattern (created by the ObjectFactory’s static method, CreateObject) is being used to tell the notifier classes to update their respective observer. The CommandController, Datacontroller and DatabaseController ensures that:

* DataDashboardUI, ChallengeUI and HistoryDashboardUI (Observers) wait for the respective notifier class to update in the notifiers’ ICollection list.
* Updating the DataDashboardUI when a new RobotData object is made.
* Updating ChallengeUI with the updated sequence of commands when the ArrayList of Commands is modified with a new Command instance or a Command instance is removed
* Updating the HistoryDashBoardUI when a new ArrayList of Commands is sent.

# 6. Testing and Evaluation

Black box testing has been applied to test and evaluate the Robot Car Website System. This section is categorise to two parts - Test cases and State Diagram

## 6.1. System State Diagram



[*https://lucid.app/lucidchart/9751a8a8-25e3-410e-9e02-a37743171b18/edit?invitationId=inv\_7a63d896-f872-499a-aaaa-d86190b2a580*](https://lucid.app/lucidchart/9751a8a8-25e3-410e-9e02-a37743171b18/edit?invitationId=inv_7a63d896-f872-499a-aaaa-d86190b2a580)

*Figure 16: System State Diagram*

## 

## 6.2. System Test Cases (User Acceptance Testing)

| **Selected Feature** | **Input** | **Output** |
| --- | --- | --- |
| parseStudent() | Enter student login credential | Get to students web portal view |
| parseTeacher() | Enter teacher login credential | Get to teachers web portal view |
| getMode(modeID: int) | modeID == 1 && isTeacher == False | Set view to tutorial view |
| modeID == 2 && isTeacher == True | Set view to mapmode view |
| modeID == 3 && isTeacher == False | Set view to challenge view |
| setView() | User selects view mode | Opens the viewUI |
| moveCar(commandList: List) | commandList != empty | Moves the car based on the commands in the list |
| closeView() | User selects to close view | Closes the viewUI |

## 

### **6.2.1 Test case 1 (ID:1)**

| Test Case | Student logins to web portal |
| --- | --- |
| Pre-condition/ (s) | Student with credential ready on hand |
| Steps | 1. Select login page 2. Enter credentials |
| Expected Result | login successfully |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 2 (ID:2)**

| Test Case | Teacher logins to web portal |
| --- | --- |
| Pre-condition/ (s) | Teacher with credential ready on hand |
| Steps | 1. Select login page 2. Enter credentials |
| Expected Result | Login successfully |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 3.1 (ID:3)**

| Test Case | Get tutorial mode for student |
| --- | --- |
| Pre-condition/ (s) | Student must successfully login |
| Steps | 1. Select view different mode 2. Mode display options - tutorial and challenge 3. Select tutorial mode |
| Expected Result | Set view to tutorial mode so that student can view the tutorial |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 3.2 (ID:3.2)**

| Test Case | Get challenge mode for student |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * User must be Student |
| Steps | 1. Select view different mode 2. Mode display options - tutorial and challenge 3. Select challenge mode |
| Expected Result | Set view to challenge mode so that student can view the challenge |
| Actual Result |  |
| Pass / Fail |  |

# 

### **6.2.1 Test case 3.3 (ID:3.3)**

| Test Case | Web portal displays Teacher view |
| --- | --- |
| Pre-condition/ (s) | * Teacher login to portal * User must be Teacher |
| Steps | 1. Select Teachers can choose to see selected student’s history of commands 2. Teacher can only see an option to select map mode |
| Expected Result | Set view to map mode so that teacher can view the map to set challenge |
| Actual Result |  |
| Pass / Fail |  |

### 

### **6.2.1 Test case 4.1 (ID:4.1)**

| Test Case | Web portal displays Student view |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * Student selects mode to view |
| Steps | 1. Web portal displays student view. 2. Student can select any 2 mode 3. Web portal displays modes that can be selected, Challenge Mode and Tutorial mode |
| Expected Result | Updates the web portal to display specific instructions (?) |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 4.2 (ID:4.2)**

| Test Case | Web portal displays Teachers view |
| --- | --- |
| Pre-condition/ (s) | * Teacher login to portal |
| Steps | 1. Teacher is able to select Student to view his history on HistoryDashboardUI 2. Teacher can use Map Mode |
| Expected Result |  |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 5 (ID:5)**

| Test Case | The virtual car execute the commands from command list |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * Student in challenge mode * Student submits at least 1 command |
| Steps | 1. Student selects the commands 2. Student submit the commands 3. Web portal displays a successful message. |
| Expected Result | The virtual car execute commands |
| Actual Result |  |
| Pass / Fail |  |

### 

### **6.2.1 Test case 6 (ID:6)**

| Test Case | Web portal is able to close all 3 types modes |
| --- | --- |
| Pre-condition/ (s) | * Student or teacher login to portal * Student or teacher is inside a page |
| Steps | 1. Select close view |
| Expected Result | Shows available modes to choose |
| Actual Result |  |
| Pass / Fail |  |

### **6.2.1 Test case 7 (ID:7)**

| Test Case | Student can delete chosen commands from the command list. |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * Command list is not empty |
| Steps | 1. Student selected some commands to move the car 2. Student deletes a command. |
| Expected Result | Student is able to delete a chosen command in the command list. |
| Actual Result |  |
| Pass / Fail |  |

### 

### **6.2.1 Test case 8 (ID:8)**

| Test Case | Student and Teacher is able to see the connection between the robot and web portal |
| --- | --- |
| Pre-condition/ (s) | Student or Teacher has login |
| Steps | 1. Student / Teacher login into the web portal 2. Web portal displays the status of connection between the robot and website |
| Expected Result | Web portal displays the status of connection between the robot and website |
| Actual Result |  |
| Pass / Fail |  |

### 

### **6.2.1 Test case 9 (ID:9)**

| Test Case | Student view chosen commands |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * Student select at least one commands |
| Steps | 1. Go challenge mode 2. Select commands 3. Display commands |
| Expected Result | List of commands selected |
| Actual Result |  |
| Pass / Fail |  |

### 

### **6.2.1 Test case 10 (ID:10)**

| Test Case | Student view history of commands |
| --- | --- |
| Pre-condition/ (s) | * Student login to portal * Student select at least one command * commands submitted by student |
| Steps | 1. Select view history page |
| Expected Result | See list of past commands in history page |
| Actual Result |  |
| Pass / Fail |  |

# 7. Work Distribution & Plan

## 7.1. SDLC Approach

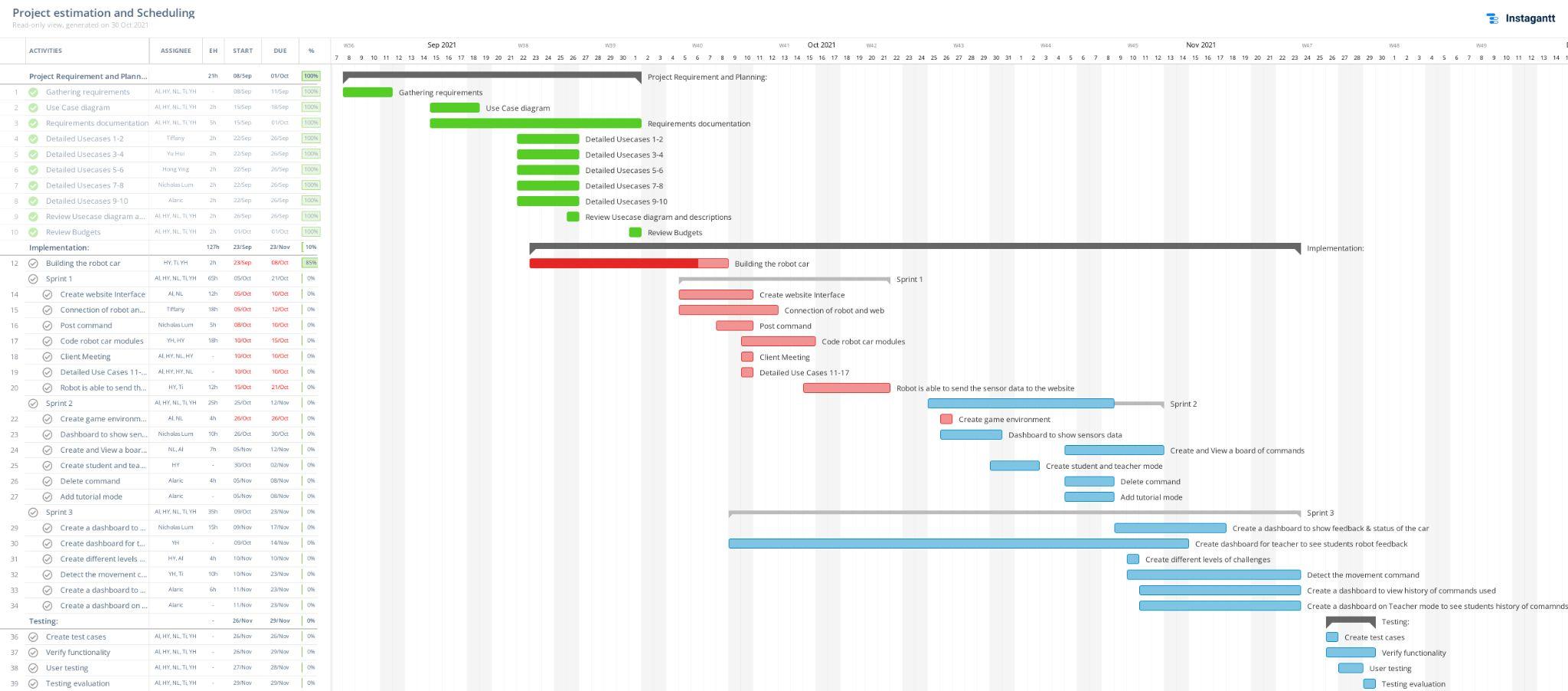
This project implemented Agile Scrum methodology as its SDLC approach. The Agile scrum methodology is used because of its ability to provide flexibility in a project that has a strict timeline. Furthermore, it allows instant adaption to project changes. As such, this methodology enables the team to conduct frequent meetings to keep themselves updated and be on track with the most recent discussion.

Other methodologies such as Waterfall or Spiral are not feasible as the robot car website system is a relatively new system that the team is not familiar with. This makes it not advisable to work with the Waterfall methodology as it requires clear understanding of the project at the start. In addition, Spiral is not advisable to be used as the project time frame is too short and it uses iterations that could take up to half a year. Furthermore, Spiral requires experts in risk which the team does not have and hence it is not advisable to use this methodology.

Finally, since this is a medium sized project that can be broken down into modules, the project can be done effectively by using the Agile Scrum methodologies as it can be broken into user stories with varying priorities to be completed by the team.

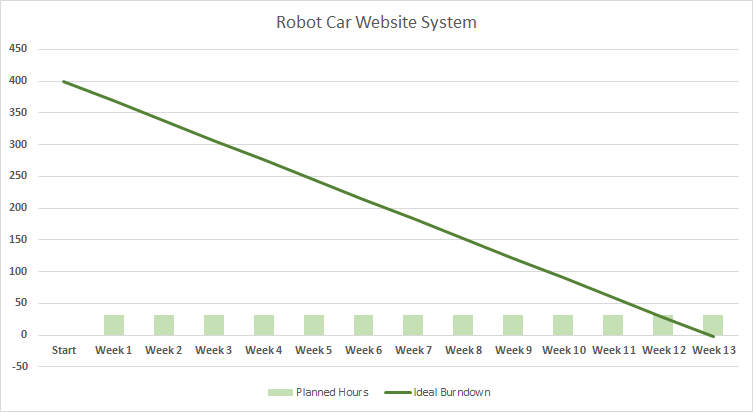
## 7.2. Project estimation and Scheduling

To estimate the project size and scheduling, the size estimation must be first calculated. Refer to appendix B for the workings. From calculating the use case points, which is used to calculate the Effort Estimation which is 322.6608 to 645.3216 hours (Refer to Appendix A for calculations)

Gantt Chart

*Figure 17: Gantt Chart for task distribution*

Burndown chart



*Figure 17: Burndown Chart for estimation for tasks to be completed*

# 8. Conclusions

In conclusion, the design of the robot car website system aims to help students develop critical-thinking skills in a fun and engaging way. Students are required to complete the challenges based on the system that is being developed. The purpose of this documentation is to aid the process of development and planning of the system product. As such, this document highlights the functional and non-functional requirements, Use Case model and descriptions, SDLC approach and external interface requirement. Furthemore, a detailed estimation on the size of project as well as the time needed to complete each component is done by gathering and eliciting client requirements. Due to the tight timeline for this project, the Agile SCRUM methodology will be used.

# 9. Individual Reflections

### Tiffany

In milestone 1, I have a deeper understanding on creating a Software Requirement Specification (SRS) document and have gained knowledge on how crucial the planning and requirement process is at the start of the project. The process of planning was meticulous and through this, I have acquired new knowledge on project estimation as well as familiarising and sharpening my skills with gantt charts. In M2, it surprised me at how many different diagrams are needed in the SRS documentation.

### Alaric

From M1, I have learnt about project management and why planning is so critical before even starting on implementing the solution. It has taught me about how to break down the entire task into easier to understand components and help see the bigger picture of how our system works. In M2, I have learned how to plan a software system in detail by breaking down the software plan in order to achieve a maintainable code that can be developed quickly due to it being able to be developed concurrently due to loosely coupled components in the code. I have also learned about the available architectures that can be used and design patterns that can be used to solve many issues faced that can be used to fill in the missing logic between classes in the class diagram. Finally, I have also learnt how to process and plan how to test the system to ensure that it is meeting the client’s requirements.

### Nicholas

From M2, I felt like I have learnt a whole lot more new things in-depth as compared to M1. I learnt how to draw a class diagram, design pattern diagram and many more! I am grateful for the teammates I have gotten as they are very helpful to me and one another. Furthermore, M2 gave me further insight to the importance of diagrams and how they help one another, an example would be that we would often miss out on certain functions in our class diagram that we later added to as we are doing our system state diagram. I also learnt from this how time consuming it is to make mistakes at this point of the project as one wrong mistake would lead us to change a lot of stuff, an example would be that we changed one of our use cases to fit better to what we intend for our website to do, however, due to this minor change, we needed to recalculate our UCPs etc. And from this I wouldn’t even want to imagine what it is like if a waterfall SDLC would be like if an urgent change needed to be made at the end of the development phase.

### Hong Ying

With the given feedback about the M1, we managed to modify some of the things such as the functional requirement and redefine the use case diagram and use case description. It is to provide more accurate information to meet the client requirement.

During the process of M2, we have included additional UML diagrams such as class diagrams, component diagrams and sequence diagram to provide us with the overall structure of our system and also understand better how the system will be interacting with users.

For the system architecture design and design pattern, we are able to group the component into the MVC so that we can visualize better the flow in the system. We also have applied some of the methods such as factory and observer to help us in resolving object class issues that we might have during the implementation phase.

With the Blackbox testing that we have learnt during the class, we have created a state diagram of our system and also derived test cases that will be used to test our system to determine whether the result that our system produces will be matched with the expected result.

Overall, I have learnt about the importance of creating a proper UML diagram. It is because, without the class diagram and the sequence diagram, it will be difficult for us to see the relationship between the objects that our system will be using. Moreover, I have understood better the architecture design and the Blackbox testing as all these are important to understand how parts or messages or data flow within the system and also testing about our system.

### Yu Hui

I learned how to construct a better SRS document in Milestone 1. This is something I did not get to do during my polytechnic days. In the past, they refer to this as a report hence I was unappreciative of doing the report because I never knew the purpose in crafting out use case descriptions, drawing out use case diagrams, etc back then. However, through this M1, I realised the importance of documenting requirements as accurately as possible. This is because SRS provides a standardisation for every team involved in the development of the project to follow. It provides crucial information to teams such as developers, clients, maintenance etc. Every single word that is specified in the SRS will have some form of legal consequences and a single mistake could possibly cause confusion to others. For instance, the developers might misunderstand a particular feature or the client might misunderstand your intention of the project. Misunderstanding is costly in this case because the team has to redo it again and as such we do not want such to occur. Furthermore, I have also learned new things such as estimating the project costs. Though I am very tempted to dive straight into doing the project, M1 taught me to stay cool about it and focus on gathering project requirements and from there to strategize/ plan. Overall, I am pretty amazed how one is able to learn so many things in just mere 3-4weeks!

For M2, I find that the complexity is on a next level. It is mindblowing that there are so many diagrams needed before we get to embark on the implementation.

# 10. References

[1] S. Anwar, N. A. Bascou, M. Menekse, and A. Kardgar, “A systematic review of studies on Educational Robotics,” *Purdue e-Pubs*. [Online]. Available: https://docs.lib.purdue.edu/jpeer/vol9/iss2/2/. [Accessed: 01-Oct-2021].

# 11. Appendix A

Calculation of unadjusted weight of all use cases:

| Use Case | Number of transactions | Weight |
| --- | --- | --- |
| UC-1 | 4 | 10 |
| UC-2 | 2 | 5 |
| UC-3 | 8 | 15 |
| UC-4 | 6 | 10 |
| UC-5 | 4 | 10 |
| UC-6 | 2 | 5 |
| UC-7 | 4 | 10 |
| UC-8 | 3 | 5 |
| UC-9 | 3 | 5 |
| UC-10 | 3 | 5 |
| UC-11 | 3 | 5 |
| UC-12 | 2 | 5 |
| UC-13 | 7 | 10 |
| UC-14 | 4 | 10 |
| UC-15 | 2 | 5 |
| UC-16 | 4 | 10 |
| UC-17 | 2 | 5 |

Calculating unadjusted weight of all use cases:

| Use case complexity and weight | Number of use cases | Product of weight with number |
| --- | --- | --- |
| Simple (5) | 9 | 9 \* 5 = 45 |
| Average (10) | 7 | 7 \* 10 = 70 |
| Complex (15) | 1 | 1 \* 15 = 15 |

Total unadjusted weight of use cases = 45 + 70 + 15 = 130

Calculating all actors:

| Actor Type | Number of Actors | Product |
| --- | --- | --- |
| Simple (1) | 0 | 0 \* 1 = 0 |
| Average (2) | 8 | 8 \* 2 = 16 |
| Complex (3) | 16 | 16 \* 3 = 48 |

Total unadjusted actors weight = 0 + 16 + 48 = 64

Total unadjusted points (UUCW) = 64 + 130 = 194

Technical Complexity Factor:

| Factor | Description | Weight | Degree of influence | Product |
| --- | --- | --- | --- | --- |
| T1 | Distributed system | 2.0 | 4 | 8.0 |
| T2 | Response time/performance objectives | 1.0 | 4 | 4.0 |
| T3 | End-user efficiency | 1.0 | 2 | 2.0 |
| T4 | Internal processing complexity | 1.0 | 3 | 3.0 |
| T5 | Code reusability | 1.0 | 3 | 3.0 |
| T6 | Easy to install | 0.5 | 0 | 0.0 |
| T7 | Easy to use | 0.5 | 5 | 5.0 |
| T8 | Portability to other platforms | 2.0 | 3 | 3.0 |
| T9 | System maintenance | 1.0 | 3 | 3.0 |
| T10 | Concurrent/parallel processing | 1.0 | 3 | 3.0 |
| T11 | Security features | 1.0 | 4 | 4.0 |
| T12 | Access for third parties | 1.0 | 0 | 0.0 |
| T13 | End user training | 1.0 | 1 | 1.0 |

Degree of Influence (DI) = 8 + 4 + 2 + 3 + 3 + 5 + 3 + 3 + 3 + 4 + 1 = 39

Technical Complexity Factor = 0.6 + 0.01 \* 39 = 0.99

Environment Factors:

| Environment Factors | Weight | Assessment |
| --- | --- | --- |
| Familiar with development process | 1.5 | 3 |
| Part time workers | -1 | 0 |
| Analyst capability | 0.5 | 2 |
| Application experience | 0.5 | 4 |
| Object Oriented Experience | 1 | 4 |
| Motivation | 1 | 5 |
| Difficult Programming Language | -1 | 5 |
| Stable Requirements | 2 | 5 |

EFactor = 3 + 2 + 4 + 4 + 5 + 5 + 5 = 28

EF = 1.4 + (-0.03 \* 28) = 0.56

With this, derive the Use Case Points which is :

UUCW \* TCF \* EF

194 \* 0.99 \* 0.56 = 107.5536

Effort Estimation using Use Case Points:

15 \* 107.5536 to 30 \* 107.5536 (In Hours)

1613.304 to 3226.608 hours for one person

Estimated effort in team:

1613.304 / 5 to 3226.608 / 5 = 322.6608 to 645.3216 hours