

**Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ver. Rel. No. | Release Date | Prepared By | Module Name | Reviewed By | Remarks/Revision Details |
| 1.0 | 16/02/2022 | Balaji P  40020526 | C Programming on Multiple platforms |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Essentials of Embedded System |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Applied SDLC and Software Testing |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | OOPS with Python |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Applied Model Based Design Module |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Mastering Microcontrollers with Embedded Driver Development module |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Overview of Automotive Systems |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Applied Control Systems and Vehicle Dynamics |  |  |
| 1.0 | 16/02/2022 | Balaji P  40020526 | Classic Autosar Basic to Intermidiate |  |  |

Contents

[Miniproject – 1: Shuttle Score [Individual] 4](#_Toc96429210)

[Modules: 4](#_Toc96429211)

[Requirements 4](#_Toc96429212)

[SWOT Analysis 5](#_Toc96429213)

[4W's and 1'H 5](#_Toc96429214)

[High Level Requirements 5](#_Toc96429215)

[Low Level Requirements 5](#_Toc96429216)

[Design 6](#_Toc96429217)

[Test Plan 6](#_Toc96429218)

[High Level Test Plan 6](#_Toc96429219)

[Low Level Test Plan 7](#_Toc96429220)

[Implementation and Summary 7](#_Toc96429221)

[Git Link: 7](#_Toc96429222)

[Git Dashboard 7](#_Toc96429223)

[Summary 8](#_Toc96429224)

[Git Inspector Summary 8](#_Toc96429225)

[Miniproject 2 – HeatControlSystem [Individual] 8](#_Toc96429226)

[Modules 8](#_Toc96429227)

[Requirements 8](#_Toc96429228)

[Introduction 8](#_Toc96429229)

[Features 8](#_Toc96429230)

[4W's and 1H's 9](#_Toc96429231)

[High Level Requirements 9](#_Toc96429232)

[Low Level Requirements 9](#_Toc96429233)

[Design 10](#_Toc96429234)

[Test Plan 12](#_Toc96429235)

[Implementation and Summary 13](#_Toc96429236)

[Git Link: 13](#_Toc96429237)

[Git Dashboard 13](#_Toc96429238)

[Miniproject 3 – NFT Marketplace [Team] 13](#_Toc96429239)

[Modules 13](#_Toc96429240)

[Requirements 14](#_Toc96429241)

[4W's and 1 H's 14](#_Toc96429242)

[High Level Requirements 14](#_Toc96429243)

[Low Level Requirements 14](#_Toc96429244)

[Design 15](#_Toc96429245)

[Test Plan 16](#_Toc96429246)

[High Level Test Plan 16](#_Toc96429247)

[Low Level Test Plan 17](#_Toc96429248)

[Implementation and Summary 17](#_Toc96429249)

[Git Link: 17](#_Toc96429250)

[Individual Contribution and Highlights 17](#_Toc96429251)

[Summary 17](#_Toc96429252)

[Miniproject 4 – Attendance Automation[Team] 17](#_Toc96429253)

[Modules 17](#_Toc96429254)

[Requirements 18](#_Toc96429255)

[High Level Requirements 18](#_Toc96429256)

[Low Level Requirements 18](#_Toc96429257)

[Test Plan 19](#_Toc96429258)

[High Level Test Plan 19](#_Toc96429259)

[Low Level Test Plan 19](#_Toc96429260)

[Implementation and Summary 20](#_Toc96429261)

[Git Link: 20](#_Toc96429262)

[Git Dashboard 20](#_Toc96429263)

[Git Inspector Summary 21](#_Toc96429264)

[Individual Contribution and Highlights 21](#_Toc96429265)

[Miniproject 5 – Landrover Project[Team] 21](#_Toc96429266)

[Modules 21](#_Toc96429267)

[Requirements 21](#_Toc96429268)

[Design 22](#_Toc96429269)

[Miniproject 6 – Wiper Control[Team] 22](#_Toc96429270)

[Modules 22](#_Toc96429271)

[Requirements 23](#_Toc96429272)

[Features 23](#_Toc96429273)

[4W's and 1 H'S 23](#_Toc96429274)

[SWOT Analysis 23](#_Toc96429275)

[High Level Requirements 23](#_Toc96429276)

[Low Level Requirements 24](#_Toc96429277)

[Design 25](#_Toc96429278)

[Test Plan 27](#_Toc96429279)

[High Level Test Plan 27](#_Toc96429280)

[Low Level Test Plan 27](#_Toc96429281)

[Implementation and Summary 27](#_Toc96429282)

[Git Link: 27](#_Toc96429283)

[Individual Contribution and Highlights 27](#_Toc96429284)

[Miniproject 7 – Jaguar Project[Team] 28](#_Toc96429285)

[Modules 28](#_Toc96429286)

[Requirements 28](#_Toc96429287)

[Design 29](#_Toc96429288)

[Implementation and Summary 29](#_Toc96429289)

[Git Link: 29](#_Toc96429290)

[Individual Contribution and Highlights 29](#_Toc96429291)

[Miniproject 8 – EV Bike[Team] 30](#_Toc96429292)

[Modules 30](#_Toc96429293)

[Requirements 30](#_Toc96429294)

[Implementation and Summary 30](#_Toc96429295)

[Individual Contribution and Highlights 30](#_Toc96429296)

[Miniproject 9 – Parking System[Individual] 31](#_Toc96429297)

[Modules 31](#_Toc96429298)

[Requirements 31](#_Toc96429299)

[Design 32](#_Toc96429300)

[Implementation and Summary 32](#_Toc96429301)

[Git Link: 32](#_Toc96429302)

[Individual Contribution and Highlights 33](#_Toc96429303)

**List Of Figures**

[Figure 1 Behavior Diagram 7](#_Toc96434889)

[Figure 2 Git Dashboard 8](#_Toc96434890)

[Figure 3 Git Inspector Summary 9](#_Toc96434891)

[Figure 4 Block Diagram 11](#_Toc96434892)

[Figure 5 Structure Diagram 12](#_Toc96434893)

[Figure 6 Git Dashboard 14](#_Toc96434894)

[Figure 7 Behavior Diagram 16](#_Toc96434895)

[Figure 8 UserFlow Diagram 17](#_Toc96434896)

[Figure 9 Structure Diagram 17](#_Toc96434897)

[Figure 10 Git Dashboard 22](#_Toc96434898)

[Figure 11 Git Inspector Summary 23](#_Toc96434899)

[Figure 12 Structural Diagram 24](#_Toc96434900)

[Figure 13 Structure Diagram 27](#_Toc96434901)

[Figure 14 Behaviour Diagram 28](#_Toc96434902)

[Figure 15 Structure Diagram 32](#_Toc96434903)

[Figure 16 VFB Diagram 35](#_Toc96434904)

# 

# Miniproject – 1: Shuttle Score [Individual]

## Modules:

1. C Programming
2. Git

## Requirements

**Introduction**

My proposed application named ShuttleScore calculates the points of each of the players of the shuttle badminton game with respective to the point they secure. It also displays the name of the players, set points and points scored in each of the sets.

**Research**

In the previous method, they have already done the shuttle score board. Now I am trying on my own method to display the shuttle badminton score.

**Cost and Features and Timeline**

It is developed to bring a new visual to the score board in a better way.

**Defining our system**

Instead of using the same visual or the same format, we can see something new in appearance.

### SWOT Analysis

**Strength:** Very much useful in viewing the scoreboard in a new style.

**Weakness:** No additional information is displayed.

**Opportunity:** This will be helpful in knowing the points easily and appropiately.

**Threat:** May be according to every individual it may not be liked by everyone.

### 4W's and 1'H

**What:** The app that is made to view the shuttle scoreboard in a new style.

**Who:** It will be beneficial for the game viewers.

**When:** To view the score in an impressive manner.

**Where:** This can be used in the television.

**How:** Implementing the scoreboard using C language.

### High Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HLR\_1 | Get the players name | Implemented |
| HLR\_2 | Display the set score | Implemented |
| HLR\_3 | Diplay the current set score | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HLR\_1 | Players names are taken as string | Implemented |
| HLR\_2 | Set score is printed as integer | Implemented |
| HLR\_3 | Current set score is printed as integer | Implemented |

## Design

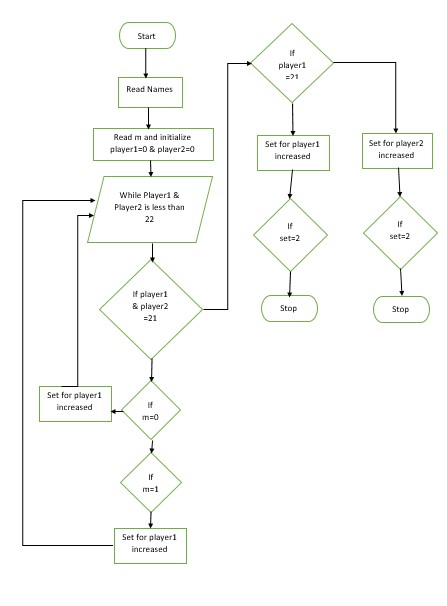


Figure 1 Behavior Diagram

## Test Plan

### High Level Test Plan

| **ID** | **Description** | **Expected I/P** | **Expected O/P** | **Actual O/P** | **Type Of Test** |
| --- | --- | --- | --- | --- | --- |
| HLTP\_1 | User Input | Enter Values | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_2 | Player Input 1 | Enter Values | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_3 | Player Input 2 | Enter Values | SUCCESS | SUCCESS | Requirement Based |

### Low Level Test Plan

| **ID** | **Description** | **Expected I/P** | **Expected O/P** | **Actual O/P** | **Type Of Test** |
| --- | --- | --- | --- | --- | --- |
| LLTP\_1 | User Input | (0) | Value Increases For Player 1 | Value Increases For Player 1 | Requirement Based |
| LLTP\_2 | User Input | (1) | Value Increases For Player 2 | Value Increases For Player 2 | Requirement Based |

## Implementation and Summary

### Git Link:

Link: <https://github.com/muthupbalag1310/M1_APPLICATION_SHUTTLESCORE.git>

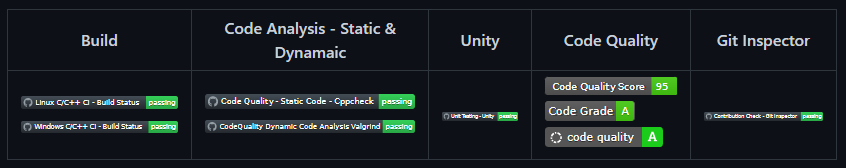
Git Dashboard 

Figure 2 Git Dashboard

### Summary

### Git Inspector Summary

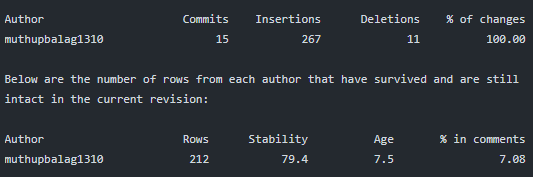


Figure 3 Git Inspector Summary

# Miniproject 2 – HeatControlSystem [Individual]

## Modules

1. C Programming
2. Embedded System
3. SimulIDE
4. Git

## Requirements

### Introduction

The heat control system is used to control the temperature of the vehicle seat. When a person gets seated on a car, the button sensor will be activated. After that, the user gets access to turn ON the heater. The temperature sensor keeps monitoring the temperature and sends the analog value to the microcontroller. The microcontroller processes the analog input of the temperature sensor and outputs a temperature value through serial communication. All the activities of the control system are done on a Atmega328 microcontroller.

### Features

The System is capable of telling whether a person is seated or not. A person once seated gets the access to turn ON the heater. The temperature sensor keeps monitoring the temperature and sends the analog value to the microcontroller.

**Strength**

Cost effective. Easily accessible. High efficiency.

**Weakness**

Overheating.

**Opportunities**

This system can be expanded by adding few more features depending on the user requirement. Usage of different Microcontrollers and sensors.

**Threats**

Competition

### 4W's and 1H's

**What**

Heat control system in a vehicle.

**Where**

Used in almost all of the passenger vehicles.

**When**

When temperature is low.

**Who**

To maintain body temperature of passengers.

**How**

By using sensors and microcontroller.

### High Level Requirements

| **ID** | **Description** |
| --- | --- |
| HLR1 | Microcontroller unit |
| HLR2 | Switches |
| HLR3 | Temperature sensor |
| HLR4 | Heater |
| HLR5 | Display CDD CRO |

### Low Level Requirements

| **ID** | **Description** | **HLR ID** |
| --- | --- | --- |
| LLR1 | ATMega328 | HLR1 |
| LLR2 | ADC and Potentiometer | HLR3 |
| LLR3 | Thermo electric module | HLR4 |
| LLR4 | LCD and LED, PWM | HLR5 |

## Design

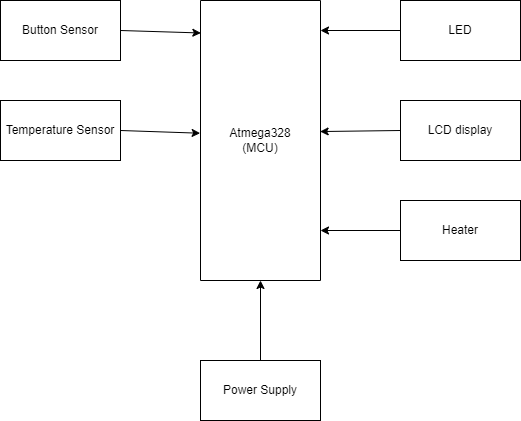


Figure 4 Block Diagram

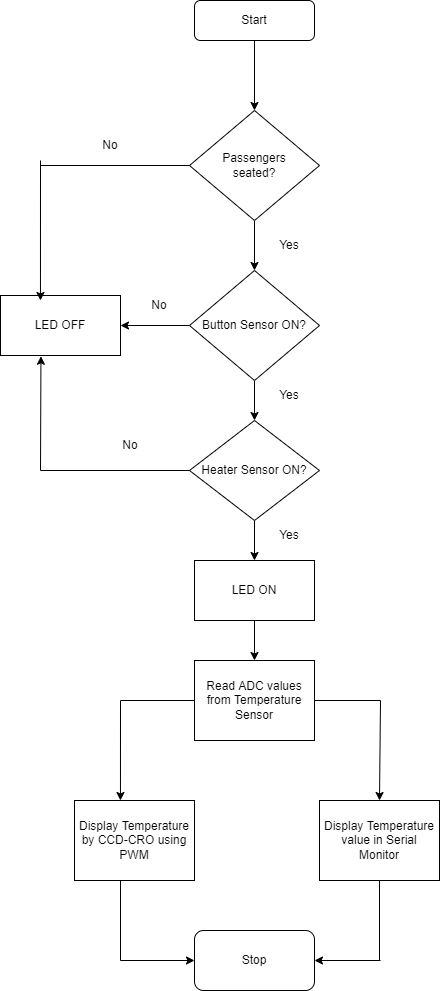


Figure 5 Structure Diagram

## Test Plan

| **Test ID** | **Description** | **EXP I/P** | **EXP O/P** | **Actual O/P** | **Type of Test** |
| --- | --- | --- | --- | --- | --- |
| T01 | If passengers seated on the car button sensor will be activated | button sensor switch closed | Button sensor on | Button sensor on | Requirement based |
| T02 | After passenger and driver seated, the user needs to enable the heater sensor | Heater sensor switch closed | Heater sensor on | Heater sensor on | Requirement based |
| T03 | Enabling both button and heater sensor, LED will be ON (binary output) | Both switches closed | LED ON | LED ON | Requirement based |
| T04 | Enabling any one of sensor not both | Any one of the switch open | LED OFF | LED OFF | Boundary based |
| T05 | Reads temperature information from temperature sensor and convert analong inputs to digital using ADC | Read ADC from temperature sensor | Successfully read and covert digital values | Successfully read and covert digital values | Requirement based |
| T06 | Display CDD-CRO will give the temperature value by showing PWM | Read ADC values | Successfully displayed temperature | Successfully displayed temperature | Requirement based |
| T07 | Display digital temperature values in serial monitor using USART communication protocol | Read ADC values | Successfully displayed temperature | Successfully displayed temperature | Requirement based |

## Implementation and Summary

### Git Link:

Link: <https://github.com/muthupbalag1310/M2-Embedded_HeatControlSystem.git>

### Git Dashboard

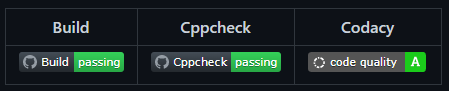


Figure 6 Git Dashboard

# Miniproject 3 – NFT Marketplace [Team]

## Modules

1. SDLC
2. Git

## Requirements

### 4W's and 1 H's

**Why:**

1. It can be used by anyone at any place.
2. Digital Items are the Future
3. Individual Creators can use this platform to sell the Digital products.

**Where:**

1. This can be used in our daily lives.
2. Can be used for international transactions

**Who**

1. It can be used by anyone.
2. It can be used as a reference for marketplace.

**When:**

1. One can buy, sell or create anytime.
2. The project can be used when anyone wants to buy an NFT.
3. Can be used without any centralised authority

**How:**

1. By using a crypto wallet anyone can Buy or Bid on NFT.
2. It will be helpful for Digital Creators.

### High Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HLR\_1 | Create NFT | Implemented |
| HLR\_2 | Sell NFT | Implemented |
| HLR\_3 | Bid NFT | Implemented |
| HLR\_4 | Buy NFT | Implemented |
| HLR\_5 | Contact | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| LLR\_1 | Sign In | Implemented |
| LLR\_2 | Register | Implemented |
| LLR\_3 | Connect Wallet | Implemented |
| LLR\_4 | Activity | Implemented |
| LLR\_5 | Forgot password | Implemented |
| LLR\_6 | Signup | Implemented |

## Design

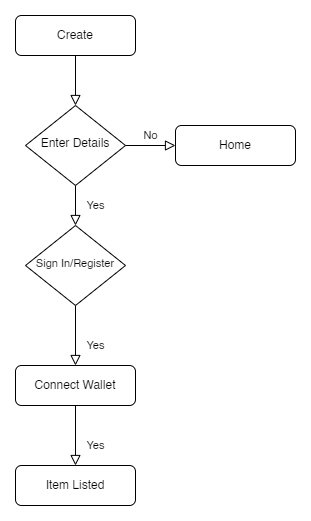
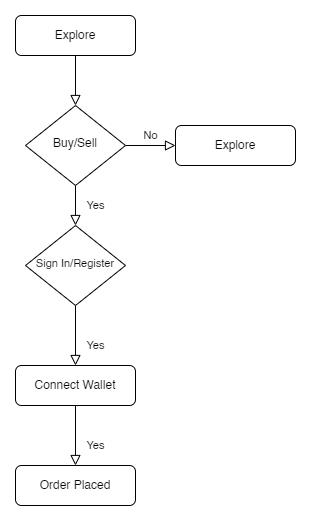


Figure 7 Behavior Diagram

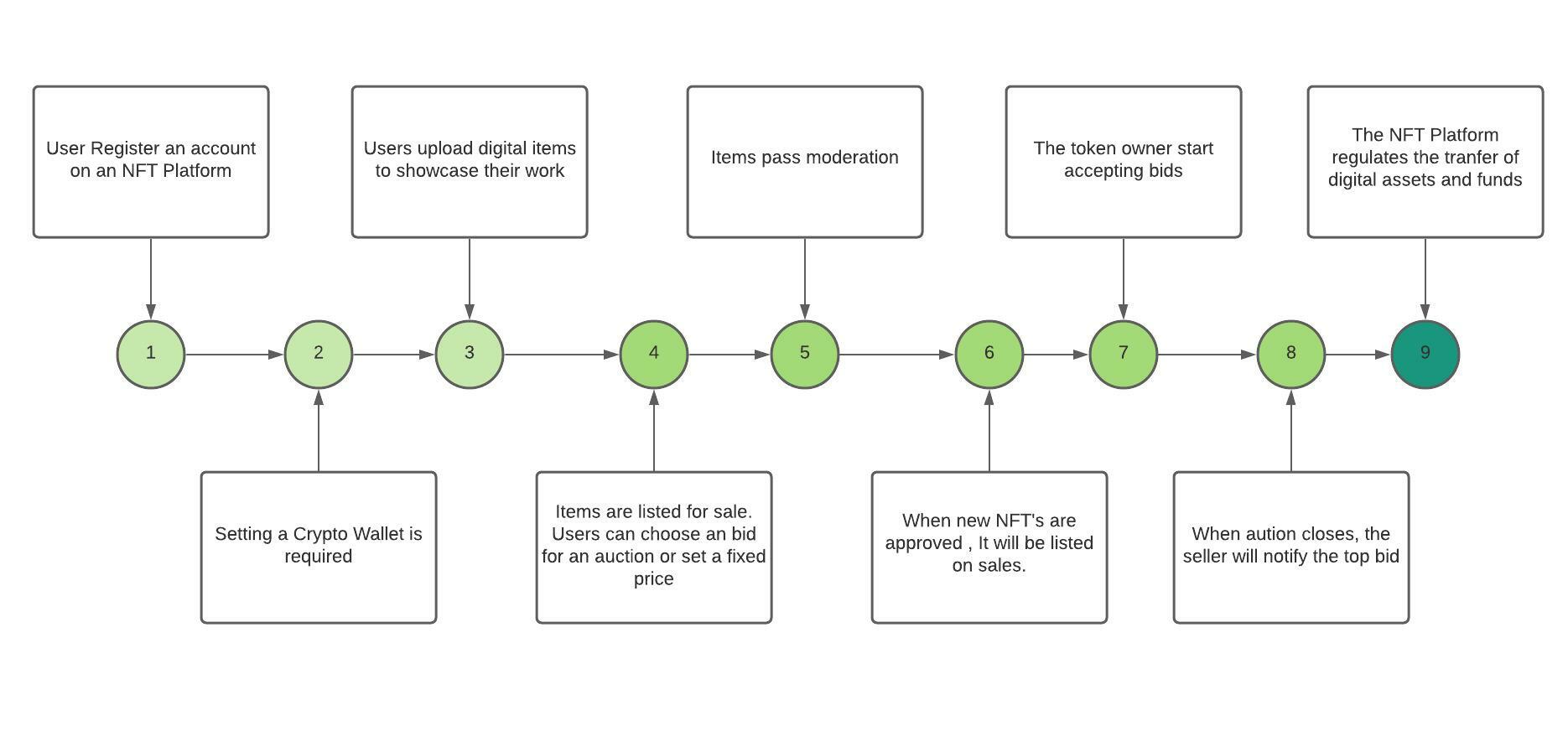


Figure 8 UserFlow Diagram

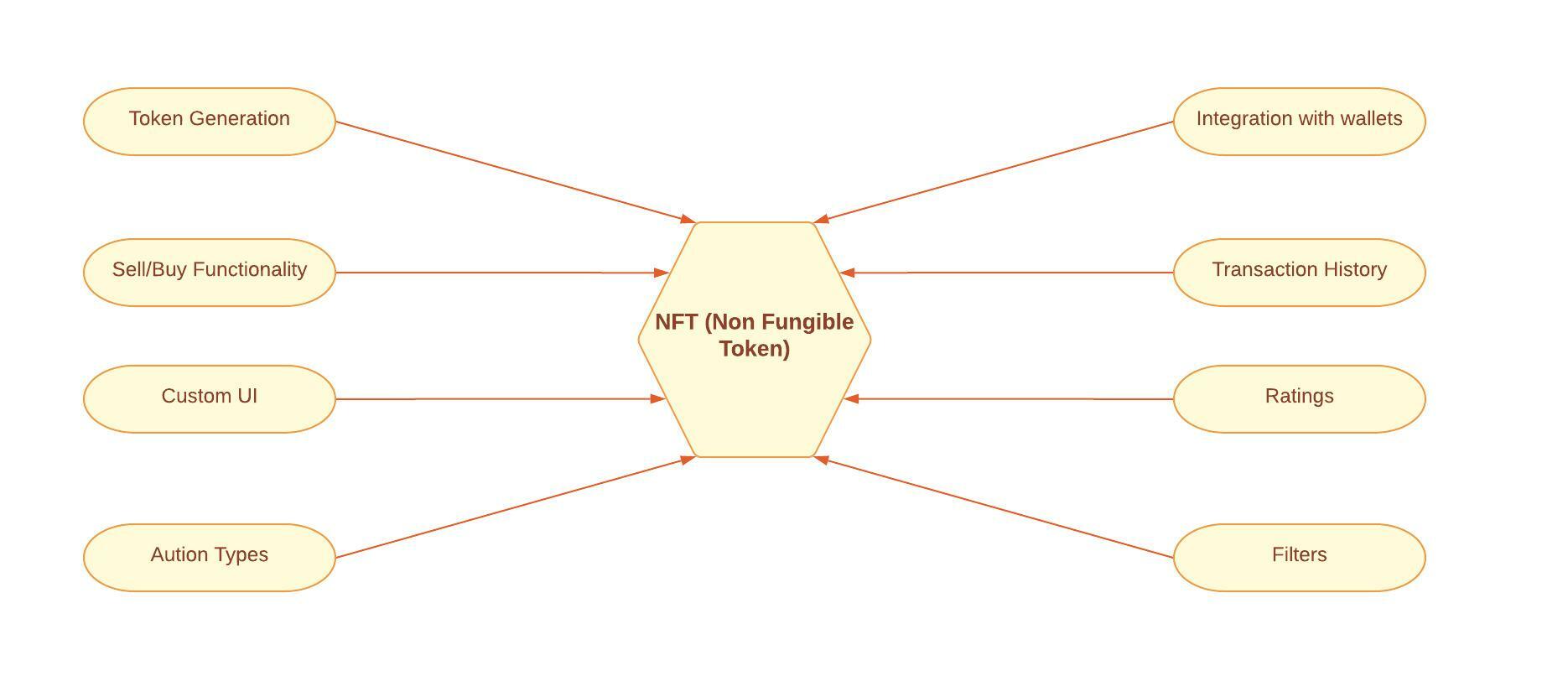


Figure 9 Structure Diagram

## Test Plan

### High Level Test Plan

| ID | Description | Expected I/P | Expected O/P | Actual O/P | Type Of Test |
| --- | --- | --- | --- | --- | --- |
| HLTP\_1 | Create NFT | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_2 | Sell NFT | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_3 | Bid NFT | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_4 | Buy NFT | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_5 | Contact | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_6 | Sign In | Click | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_7 | Register | Click | SUCCESS | SUCCESS | Requirement Based |

### Low Level Test Plan

| ID | Description | Expected I/P | Expected O/P | Actual O/P | Type Of Test |
| --- | --- | --- | --- | --- | --- |
| LLTP\_1 | Connect Wallet | Click | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_2 | Activity | Click | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_3 | Forgot password | Click | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_4 | To check whether none of the fields should be empty | Empty value in the input module | Prompt message mandatory field missing | SUCCESS | Requirement Based |
| LLTP\_5 | E-mail ID should be in the perfect format i.e. [group2@gmail.com](mailto:group2@gmail.com) | [group2@gmail.com](mailto:group2@gmail.com) | Prompt message invalid E-mail ID | SUCCESS | Requirement Based |

## Implementation and Summary

### Git Link:

Link: <https://github.com/GENESIS2021Q1/Applied_SDLC-Dec_Team_2>

Live Project Link: <https://alrichroshan.com/nft>

### Individual Contribution and Highlights

### Summary

1. Designed Homepage
2. Search Option
3. Header
4. Footer
5. Integrating All Pages Together

**Role in Project Team**

1. Designer: Designed Webpages Using HTML, CSS, JavaScript
2. Integrator: Integrated All the Pages Together
3. Tester: Testing the Webpage Performance and Bugs

# Miniproject 4 – Attendance Automation [Team]

## Modules

1. Python
2. Git

## Requirements

### High Level Requirements

| **ID** | **Feature** | **Status** |
| --- | --- | --- |
| HLR\_01 | GUI | Not Implemented |
| HLR\_02 | Attendance Status | Implemented |
| HLR\_03 | User Details | Implemented |
| HLR\_04 | User load sheet | Implemented |
| HLR\_05 | Output file generation | Implemented |

### Low Level Requirements

| **ID** | **Feature** | **High Level ID** | **Status** |
| --- | --- | --- | --- |
| LLR\_01 | GUI should allow user to enter inputs | HLR\_01 | Not Implemented |
| LLR\_02 | Input Files For Different Sessions | HLR\_01 | Not Implemented |
| LLR\_03 | User can get the Attendance Status | HLR\_02 | Implemented |
| LLR\_04 | User can enter status input to get the Attendance Status | HLR\_02 | Implemented |
| LLR\_05 | User can get the user details | HLR\_03 | Implemented |
| LLR\_06 | User will get the details after the successfull attendance entry | HLR\_03 | Implemented |
| LLR\_07 | User can load different sheets | HLR\_04 | Implemented |
| LLR\_08 | User can also modify the existing sheets as it is dynamic | HLR\_04 | Implemented |
| LLR\_09 | Output file gets generated | HLR\_05 | Implemented |
| LLR\_10 | Multiple files can be generated with different inputs | HLR\_05 | Implemented |

## Test Plan

### High Level Test Plan

| **ID** | **Description** | **Expected I/P** | **Expected O/P** | **Actual O/P** | **Type Of Test** |
| --- | --- | --- | --- | --- | --- |
| HLTP\_01 | Attendance Status | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_02 | User details | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_03 | User load sheet | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_04 | Output file generation | User Input | SUCCESS | SUCCESS | Requirement Based |

### Low Level Test Plan

| **ID** | **HLTP ID** | **Description** | **Expected I/P** | **Actual O/P** | **Type Of Test** |
| --- | --- | --- | --- | --- | --- |
| LLTP\_01 | HLTP\_01 | User can get Attendance Status | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_02 | HLTP\_01 | User can enter Status input to get the Attendance Status | SUCCESS | SUCCESS |  |
| LLTP\_03 | HLTP\_02 | User can get the User details | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_04 | HLTP\_02 | User will get the details after the successful attendance | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_05 | HLTP\_03 | User can load different sheets | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_06 | HLTP\_03 | User can also modify the existing sheets as it is dynamic | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_07 | HLTP\_04 | Output file gets generated | SUCCESS | SUCCESS | Requirement Based |
| LLTP\_08 | HLTP\_04 | Multiple files can be generated with different inputs | SUCCESS | SUCCESS | Requirement Based |

## Implementation and Summary

### Git Link:

Link: <https://github.com/alrichroshan/Attendance_Automation_Team_14>

### Git Dashboard

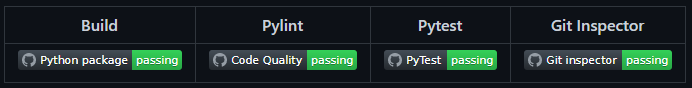


Figure 10 Git Dashboard

### Git Inspector Summary

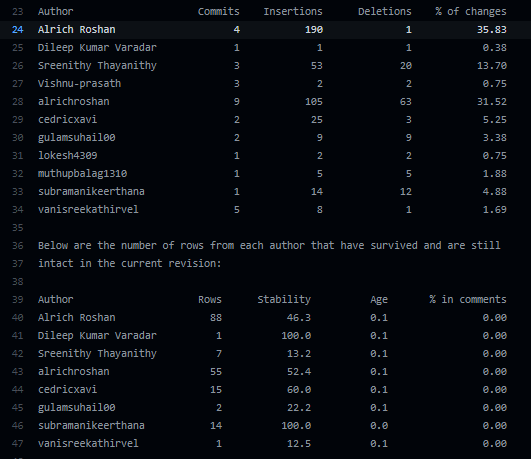


Figure 11 Git Inspector Summary

### Individual Contribution and Highlights

1. Improved implementation of Python Programming
2. Source code management using GitHub

**Role in Project Team**

1. Programmer: Done Programming for Attendance Automation
2. Integrator: Integrated all the codes
3. Tester: Writing Testcases and testing the integrated code

# Miniproject 5 – Landrover Project[Team]

## Modules

1. Matlab
2. Git

## Requirements

We have implemented following features

1. Airbag Control System
2. Door Lock System
3. Anti-Lock Braking System
4. Power Side Mirror system
5. Wiper Control System
6. Adaptive Cruise Control System
7. Power Window

## Design

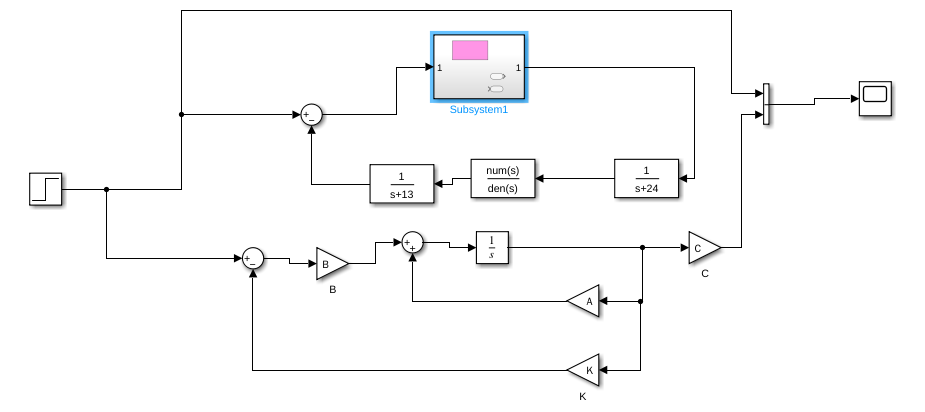


Figure 12 Structural Diagram

# Miniproject 6 – Wiper Control[Team]

## Modules

1. C Programming
2. STM32

## Requirements

### Features

1. Low power Consumption
2. High performance
3. Real-time capabilities

### 4W's and 1 H'S

**Why**

1. To understand basic concepts in STM32
2. To control wiper system by switching LED in STM32

**Where**

1. It can be used for many projects.
2. It has too many realistic features in this STM32 microcontroller

**Who**

1. It can be used by students.

2. It can be used by anyone who are new to embedded programming language.

**When**

It can be used for both small projects and end-to-end platforms.

**How**

1. By using softwares to exceute the program.
2. By loading the program in STM32

### SWOT Analysis

**Strengths**

1. Simple program to understand.

2. Time effective.

**Weakness**

Single operating program is used here

**Opportunities**

Program can be made more complex by adding more components.

**Threats**

There are advanced programs which are simple to learn is out already.

### High Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HR1 | Microcontroller | Implemented |
| HR2 | Switch | Implemented |
| HR3 | Four LED | Implemented |
| HR4 | Software | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| LLR1 | STM32 | Implemented |
| LLR2 | Switch | Implemented |
| LLR3 | LED | Implemented |

## Design

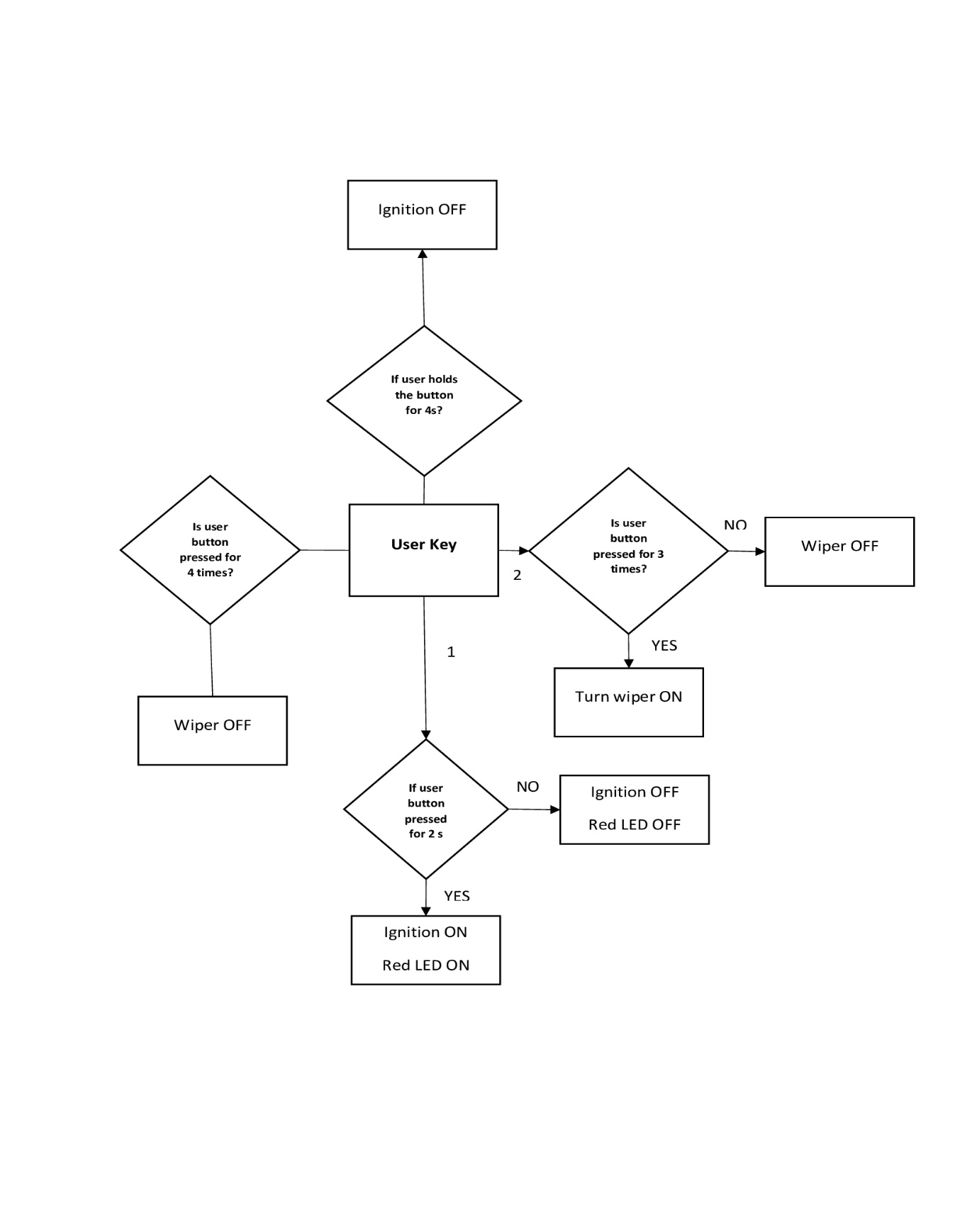


Figure 13 Structure Diagram

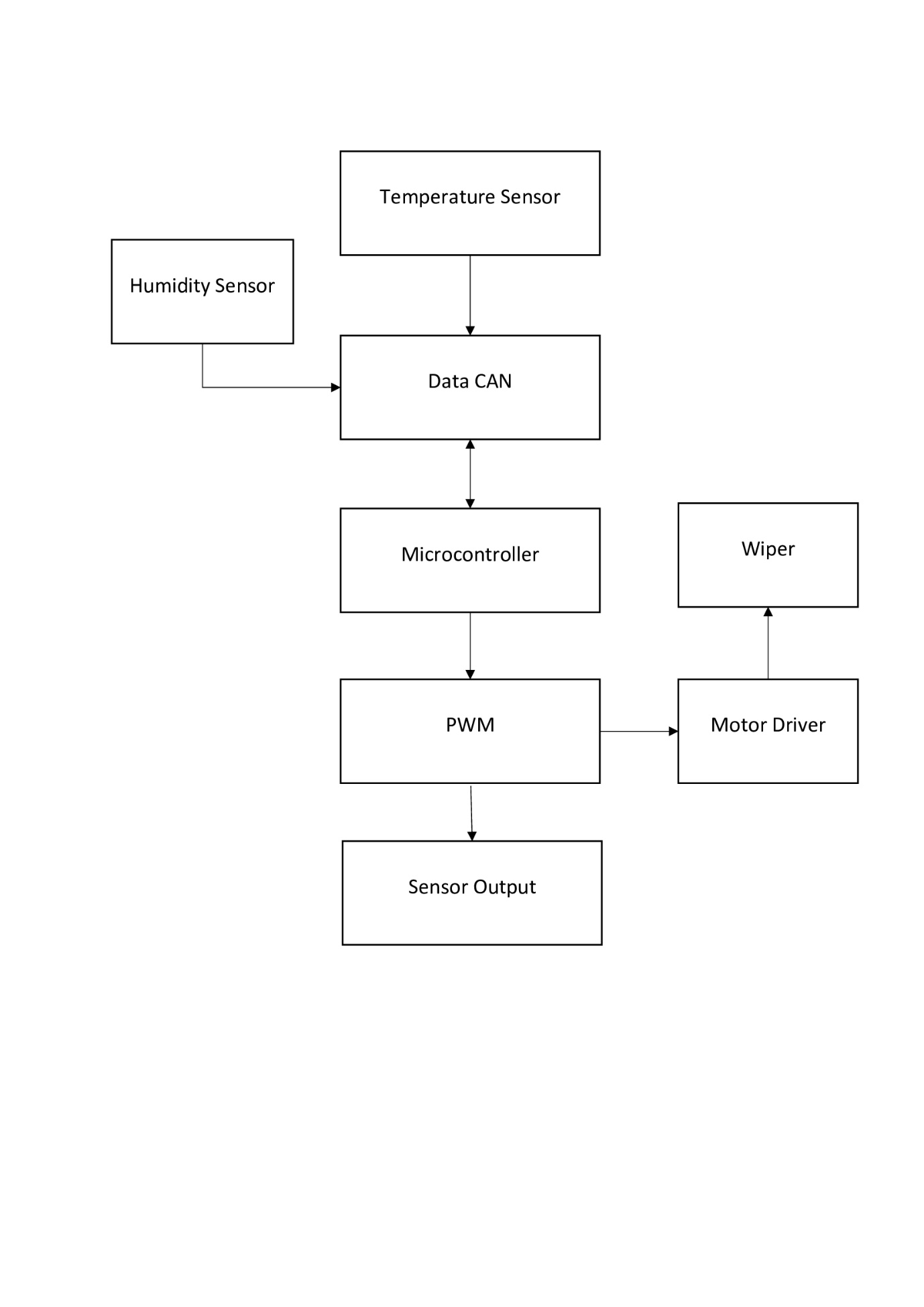


Figure 14 Behaviour Diagram

## Test Plan

### High Level Test Plan

| ID | Description | Output | Type of Test |
| --- | --- | --- | --- |
| HLTP\_1 | Press and hold the button to put the Ignition key position in ACC mode | System Enters ACC State | Requirement Based |
| HLTP\_2 | Different wiper frequencies to be set (1Hz, 4Hz & 8Hz) | Responds Based on Input | Requirement Based |
| HLTP\_3 | Hold the button to put the system in Idle state | Enters Idle State | Requirement Based |

### Low Level Test Plan

| ID | Description | Output | HLTP ID | Type of Test |
| --- | --- | --- | --- | --- |
| LLTP\_1 | Hold the button for 2 sec to bring the ignition key position at ACC mode | Red LED-ON | HLTP\_1 | Requirement Based |
| LLTP\_2 | Hold the button for 2 sec to go back to the Idle state | Red LED-OFF | HLTP\_1, HLTP\_3 | Requirement Based |
| LLTP\_3 | Press the button one time to set frequency to 1Hz | Blue LED-ON | HLTP\_2 | Requirement Based |
| LLTP\_4 | Press the button second time to set frequency to 4Hz | Green LED-ON | HLTP\_2 | Requirement Based |
| LLTP\_5 | Press the button third time to set frequency to 8Hz | Orange LED-ON | HLTP\_2 | Requirement Based |
| LLTP\_6 | Press the button fourth time to turn OFF the wiper action | All LED OFF except Red | HLTP\_2 | Requirement Based |
| LLTP\_7 | Hold the button for 2 sec to bring ignition key position at Lock state | Red LED-OFF | HLTP\_3 | Requirement Based |

## Implementation and Summary

### Git Link:

Link: <https://github.com/GENESIS-2022/MasteringMCU-Team32>

### Individual Contribution and Highlights

1. Wiper System using C Programming
2. Source code management using GitHub

**Role in Project Team**

1. Programmer: Done Programming for Wiper System
2. Integrator: Integrated all the codes
3. Tester: Writing Testcases and testing the integrated code

# Miniproject 7 – Jaguar Project[Team]

## Modules

1. Automotive Systems
2. Git

## Requirements

In this Jaguar project we have taken following features and I have contributed to Parking System Feature

1. Parking System
2. Headlight Control
3. Sideview Mirror Control
4. Wiper Control System

| **S.NO** | **Function** | **Description** |
| --- | --- | --- |
| 1 | Rain sensor | The IR sensor in the front windscreen sends the signal and if it is reflected by the rain drop then front and rear wipers are turned ON. The wipers speed are in accordance with the car speed. |
| 2 | Single wipe | The wiper is turned ON once and set back to its initial position. If it is held then it wipes until its held. |
| 3 | Manual speed modes | Low and High speed modes are chosen according to the users by rotating the switching so that the gap between each wipes varies. |
| 4 | Wash and wipe | When this mode is turned ON, the washing liquid is sprayed and wiper runs continuously until its held ON. |
| 5 | Rear windscreen wipe with car speed | When switched ON to INT mode the frequency of the rear wiper depends on the car’s speed. |
| 6 | Rear windscreen continuous automatic wipe | If rear side switch is turned to ON state so its wipes continuously until this mode is switched OFF. |
| 7 | User based control | If this mode is turned ON, it wipes the rear windscreen only when this mode is held. |

## Design

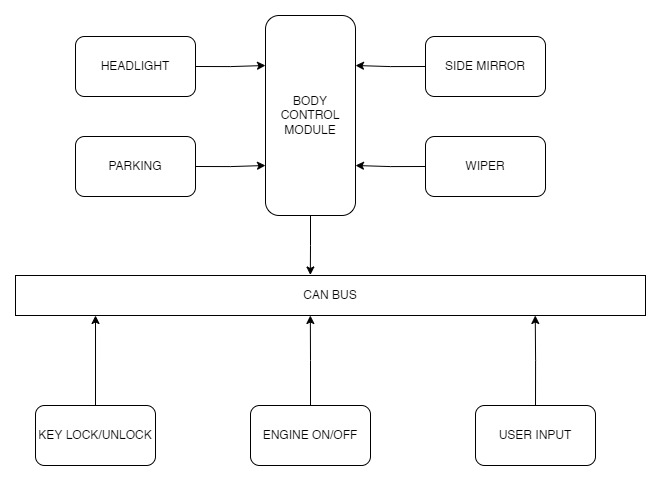


Figure 15 Structure Diagram

## Implementation and Summary

### Git Link:

Link: <https://github.com/alrichroshan/Team_Jaguar-XF>

### Individual Contribution and Highlights

* 1. Wiper Control System Case Study
  2. Source code management using GitHub

**Role in Project Team**

1. Designer: Done Designing for Project
2. Researcher: Done case study for Wiper Control System

# Miniproject 8 – EV Bike [Team]

## Modules

1. Matlab
2. Matlab Script

## Requirements

**Motor Performance:**

1. Our Arrow M1 has a Mid Drive IPM motor which can produce 7.2 kW power and 40 Nm torque. We find these figures to be a nice balance of drivability and efficiency.
2. Arrow M1 has an acceleration time from 0 to 60 km/hr of 6.5 seconds.
3. Top speed of our Arrow M1 is 100 km/hr

**Battery Performance:**

1. We are using a Lithium polymer battery to reduce weight and thereby increase fuel efficiency, performance and handling.
2. A range of 220 km is class leading due to our battery being the biggest at 4.6 kWh.
3. Charging times of our Arrow M1 is higher than the competition at 7.15 hours but we make up for it in the range section.
4. We also offer fast charging.

**Braking Performance:**

1. Our Arrow M1 also uses combi braking system and use disc brakes for both front and back wheels.
2. Braking performance is on par with the competition.

**Wheel Performance:**

1. Our Arrow M1 uses Alloy wheels at 12 inches diameter.
2. We use a 90 section, 90 profile tire for a balance between grip, efficiency and ride quality.

**Suspension Performance:**

1. We use Mono shocks for rear and single fork for front.

**Dimensions:**

1. Our kerb weight is 110 kg which is just 2 kg heavier than the Ather 450X while having a substantially bigger battery and more powerful motor.
2. Length, Height and Weight are all comparable to the competition.
3. Wheelbase is 1370 mm is the longest in the segment.
4. With a seat height of 782 mm it is accessible for a wide range of people in terms of height.

## Implementation and Summary

Submission: Submitted in GEALearn

### Individual Contribution and Highlights

1. Done in Matlab Script

**Role in Project Team**

1. Done Matlab scripting for EV Bike
2. Researcher: Done case study for EV Bike

# Miniproject 9 – Wiper Control System[Individual]

## Modules

1. Autosar
2. Git

## Requirements

| **S.NO** | **Function** | **Description** |
| --- | --- | --- |
| 1 | Rain sensor | The IR sensor in the front windscreen sends the signal and if it is reflected by the rain drop then front and rear wipers are turned ON. The wipers speed are in accordance with the car speed. |
| 2 | Single wipe | The wiper is turned ON once and set back to its initial position. If it is held then it wipes until its held. |
| 3 | Manual speed modes | Low and High speed modes are chosen according to the users by rotating the switching so that the gap between each wipes varies. |
| 4 | Wash and wipe | When this mode is turned ON, the washing liquid is sprayed and wiper runs continuously until its held ON. |
| 5 | Rear windscreen wipe with car speed | When switched ON to INT mode the frequency of the rear wiper depends on the car’s speed. |
| 6 | Rear windscreen continuous automatic wipe | If rear side switch is turned to ON state so its wipes continuously until this mode is switched OFF. |
| 7 | User based control | If this mode is turned ON, it wipes the rear windscreen only when this mode is held. |

## Design

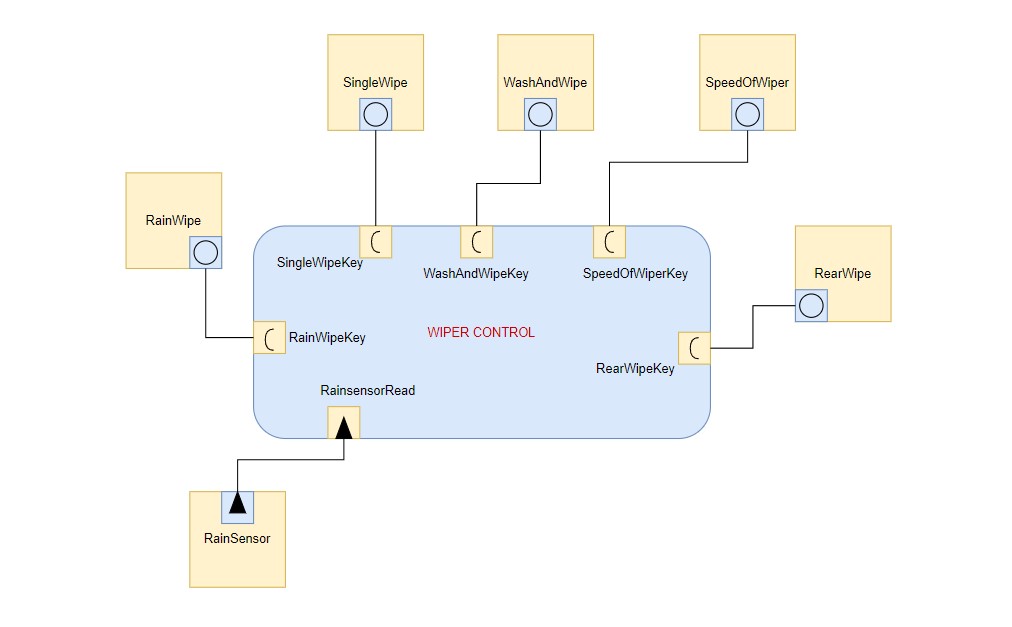


Figure 16 VFB Diagram

## Implementation and Summary

### Git Link:

Link: <https://github.com/muthupbalag1310/WiperControl_40020526_DPS.git>

### Individual Contribution and Highlights

1. Wiper Control System Case Study
2. Source code management using GitHub
3. AtomicSwComponent
4. SWCInternalBehavior
5. SWCImplementation