Details

Ver. Rel. No.	Release Date	Prepared By	Reviewed By	To Be Approved	Remarks/Revision Details
1.0	21/02/2022	Nayan Balkrishna Farde 40021043			

Contents

List of Figures

Mini Project – 1: Tourism Management System [Individual]

Modules:

1. C Programming
2. Git

Requirements

4W's and 1 H's

Who:

- This application can be useful for whom who are planning to go for tour so as grab the best tourism packages for them.

Where:

- The people who are finding best platform to book a ticket for tourism packages.

Who:

- This application can be useful for whom who are planning to go for tour so as grab the best tourism packages for them

When:

- It can be used when someone is thinking to go for the tour.

How:

- The user can do many things in the system in the system, he/she can add Book a package, check current tickets, cancel ticket, change user password. The system will give you a specific function that will help you to manage your business. The system will store your data as a text file.

High Level Requirements

ID	Description	Status
HLR_1	User shall be able to sign up for the new account	Implemented
HLR_2	User shall be able to login/logout of their account	Implemented
HLR_3	User shall be able to Book Tour Package and check Tickets	Implemented
HLR_4	Admin should be able to cancel tickets | Successful	Implemented
HLR_5	Admin should be able to change their password	Implemented

Low Level Requirements

ID	Description	Status
LLR_1	When the system is accessed then the system will give the option to sign up or login to their account	Implemented
LLR_2	When they login should give appropriate menu for them	Implemented
LLR_3	If user logged in, it should show the user's previously booked tickets	Implemented
LLR_4	The system should save the new user login data and their booked tickets	Implemented

Design

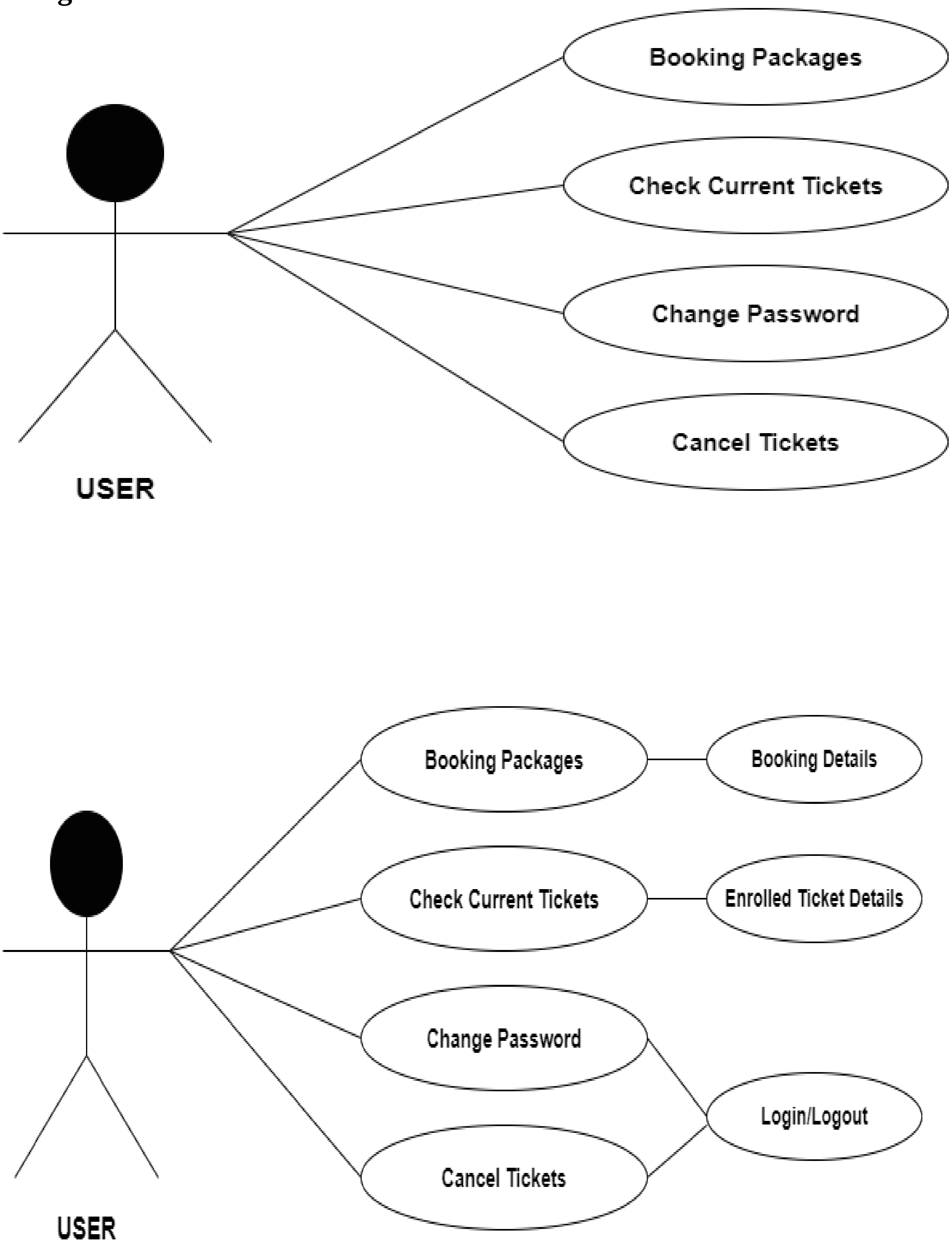


Figure 1.High Level & Low Level Behavioural Diagram

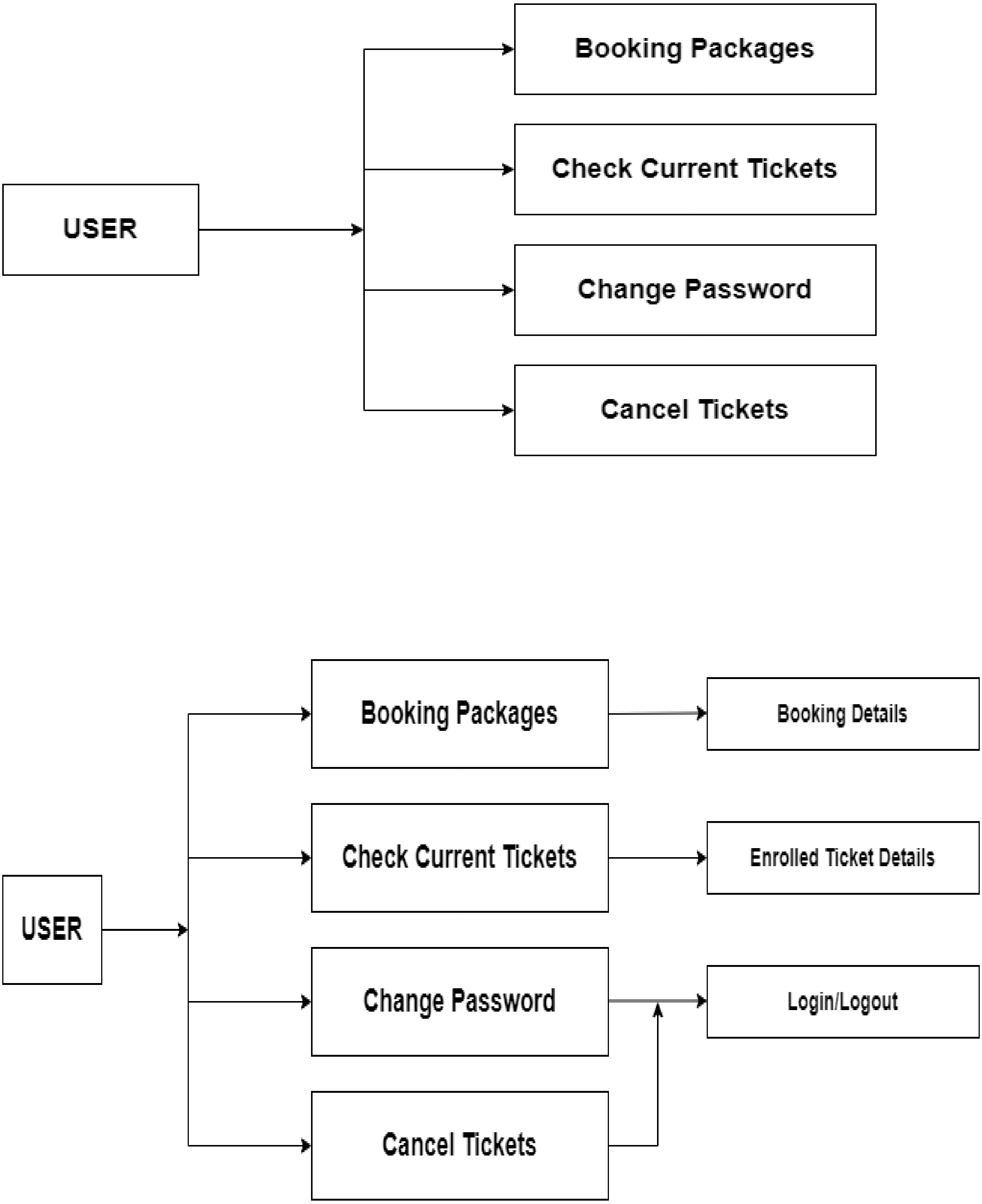


Figure 2. High Level & Low Level Structural Diagram

Test Plan

High Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HL_1	Change Password	Type Old Password	SUCCESS	SUCCESS	Requirement Based
HL_2	Ticket Booking	Show list of packages	SUCCESS	SUCCESS	Requirement Based
HL_3	Ticket Cancellation	Yes / No	SUCCESS	SUCCESS	Requirement Based

Low Level Test Plan

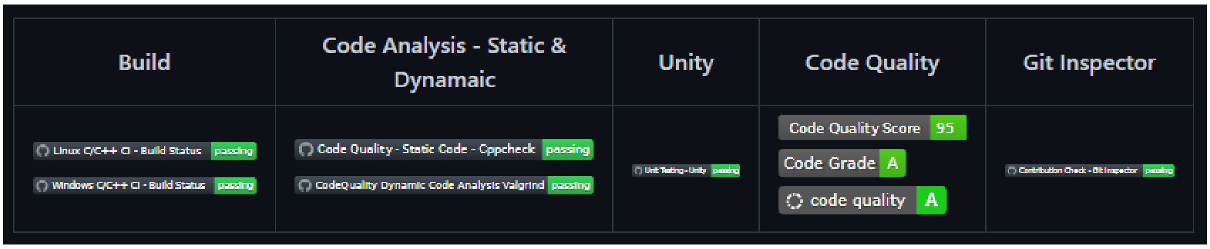
ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LL_1	Booking Details	Select booking package	Package Details	Package Details	Requirement Based
LL_2	Ticket Status	Login credentials	Confirmed / Not Confirmed	Confirmed / Not Confirmed	Requirement Based
LL_3	Login / Logout	Username / Password	Success	Success	Requirement Based

Implementation and Summary

Git Link:

Link: Nayan349/M1_Application_TourismManagementSystem (github.com)

Git Dashboard



Build	Code Analysis - Static & Dynamaic	Unity	Code Quality	Git Inspector
Linux C/C++ CI - Build Status passing Windows C/C++ CI - Build Status passing	Code Quality - Static Code - Cppcheck passing CodeQuality Dynamic Code Analysis Valgrind passing	Unit Testing - Unity passing	Code Quality Score 95 Code Grade A code quality A	Contribution Check - Git Inspector passing

Figure 3 Git Dashboard

Mini Project 2 – Obstacle Detection System [Individual]

Modules

1. C Programming
2. Embedded System
3. Simul IDE
4. Git

Requirements

4W's and 1 H's

What:

- I have made a setup based on a microcontroller in which object detection and real time distance is sensed by an ultrasonic sensor and displays measured distance on an LCD display.

Where:

- It measures accurate distance using a non-contact technology.

Who:

- The importance of the project is to detect and calculate accurate distance from any obstacle that we want to measure. Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology, medical applications and anti-collision safety systems

When:

- This will be useful to user when they need assistance in dark light and whenever we want to measure the particular distance from the moving object.

How:

- Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor if obstacle detected. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object

High Level Requirements

ID	Description	Status
HLR_1	Interfacing Ultrasonic Sensor with controller	Implemented
HLR_2	Interfacing LCD display with controller	Implemented

ID	Description	Status
HLR_3	Installing required software on the PC/Laptop	Implemented

Low Level Requirements

ID	Description	Status
LLR_1	Setting the range up to 80 cm	Implemented
LLR_2	The high-level signal is sent to 10 microseconds using Trigger	Implemented

Design

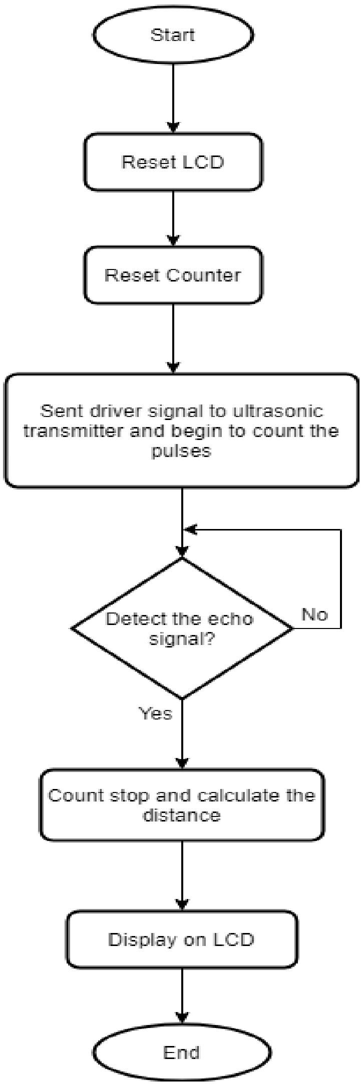

Start
Reset LCD
Reset Counter
Sent driver signal to ultrasonic transmitter and begin to count the pulses
Detect the echo signal?
No
Yes
Count stop and calculate the distance
Display on LCD
End

Figure 4 Behavioural Diagram

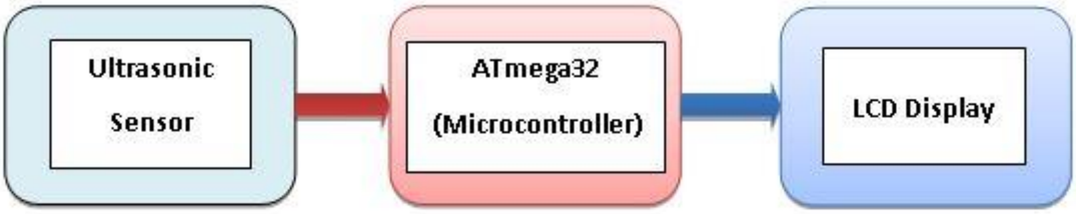
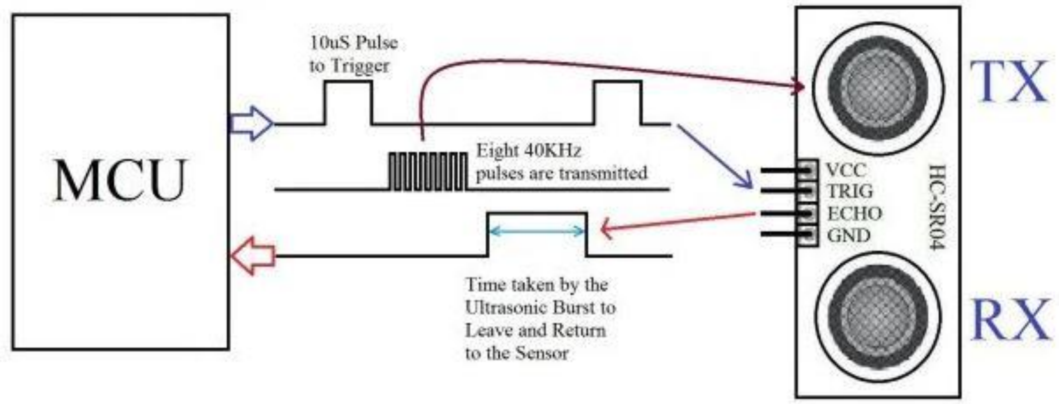


Figure 6 Block Diagram


MCU
10uS Pulse to Trigger
Eight 40KHz pulses are transmitted
Time taken by the Ultrasonic Burst to Leave and Return to the Sensor
TX
RX
VCC
TRIG
ECHO
GND
HC-SR04

Working of HC-SR04 Ultrasonic Sensor

Figure 5 Block Diagram

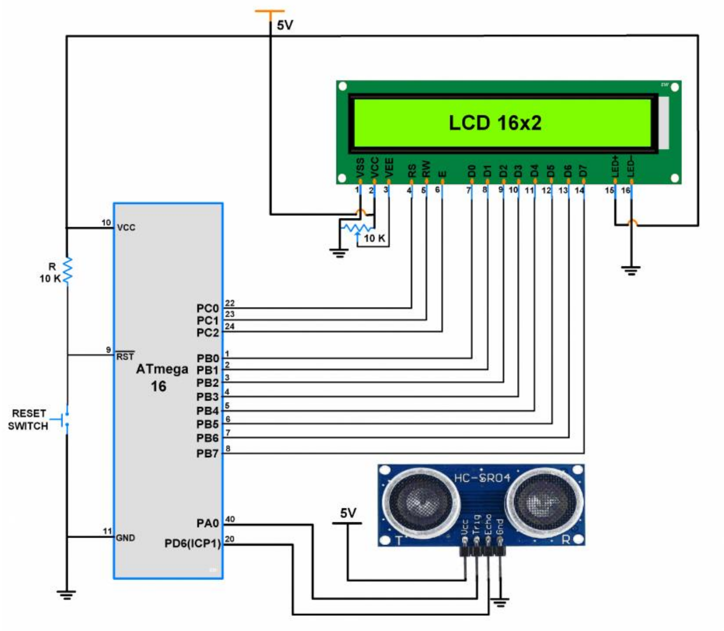


Figure 6 Simulation

Test Plan

High Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HL01	Sensor Working	Obstacle	Obstacle Detected	As Expected	Requirement Based
HL02	High accuracy	Object in the range	Accurate distance from object on Display	As Expected	Scenario based
HL03	Measuring time lapses between the sending and receiving of the ultrasonic pulse	Object in the range	Display	As Expected	Requirement Based

Low Level Test Plan

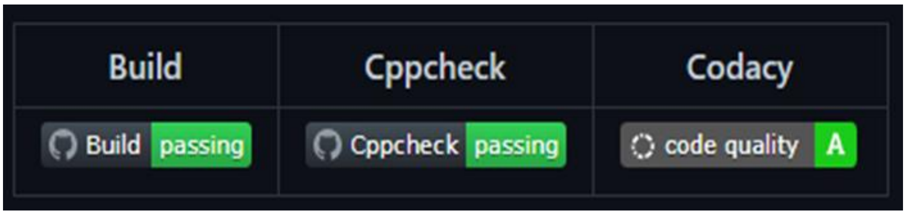
ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LL01	Detection of clear objects	Obstacle	Display	As Expected	Scenario based
LL02	Provide multiple range measurements per second	Moving Obstacle	Display	As Expected	Requirement Based

Implementation and Summary

Git Link:

Link GENESIS2021Q1/Applied_SDLC-Dec_Team_48: Details (github.com)

Git Dashboard



Build	Cppcheck	Codacy
Build passing	Cppcheck passing	code quality A

Figure 7 Git Dashboard

Mini Project 3 – Scientific Calculator [Team]

Modules

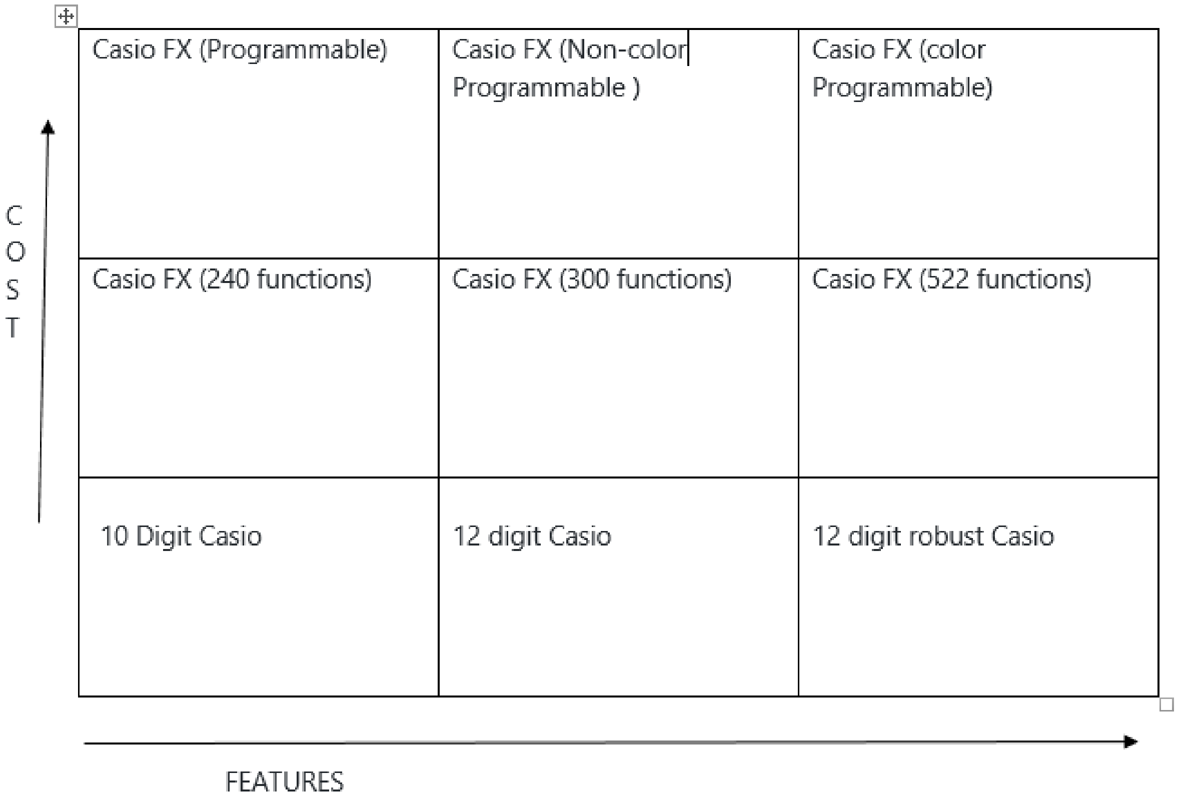
1. SDLC
2. Git

Requirements

Features

- It can do all operations (Addition, Subtraction, Multiplication, Division).
- Logarithmic operations, Exponential operations are also available.
- Power functions, Factorial and Conversions which are helpful for students are added.
- Basic Trigonometric Operations are also available.
- Basic Matrix Operations are also available.
- It has double precision.

Cost v/s features



Casio FX (Programmable)	Casio FX (Non-color Programmable)	Casio FX (color Programmable)
Casio FX (240 functions)	Casio FX (300 functions)	Casio FX (522 functions)
10 Digit Casio	12 digit Casio	12 digit robust Casio

SWOT ANALYSIS

Strengths

• User Friendly

• All basic operations

• Double Precision

• Trigonometric operations

• Matrix Operations

Weakness

• Limited Operations

• Memory Wastage

Opportunities

• It can be expanded by adding additional features like Inverse Trigonometric operations, Equations etc.

Threats

• There are other programmable calculators which may affect our product marketing.

4W's and 1'H

Why:

- For Easy and faster Calculations.

Where:

- Used in Provisional Stores and Students.

What:

- Scientific Calculator.

When:

- For Complex Calculations.

How:

- Operates by using User Inputs.

High Level Requirements

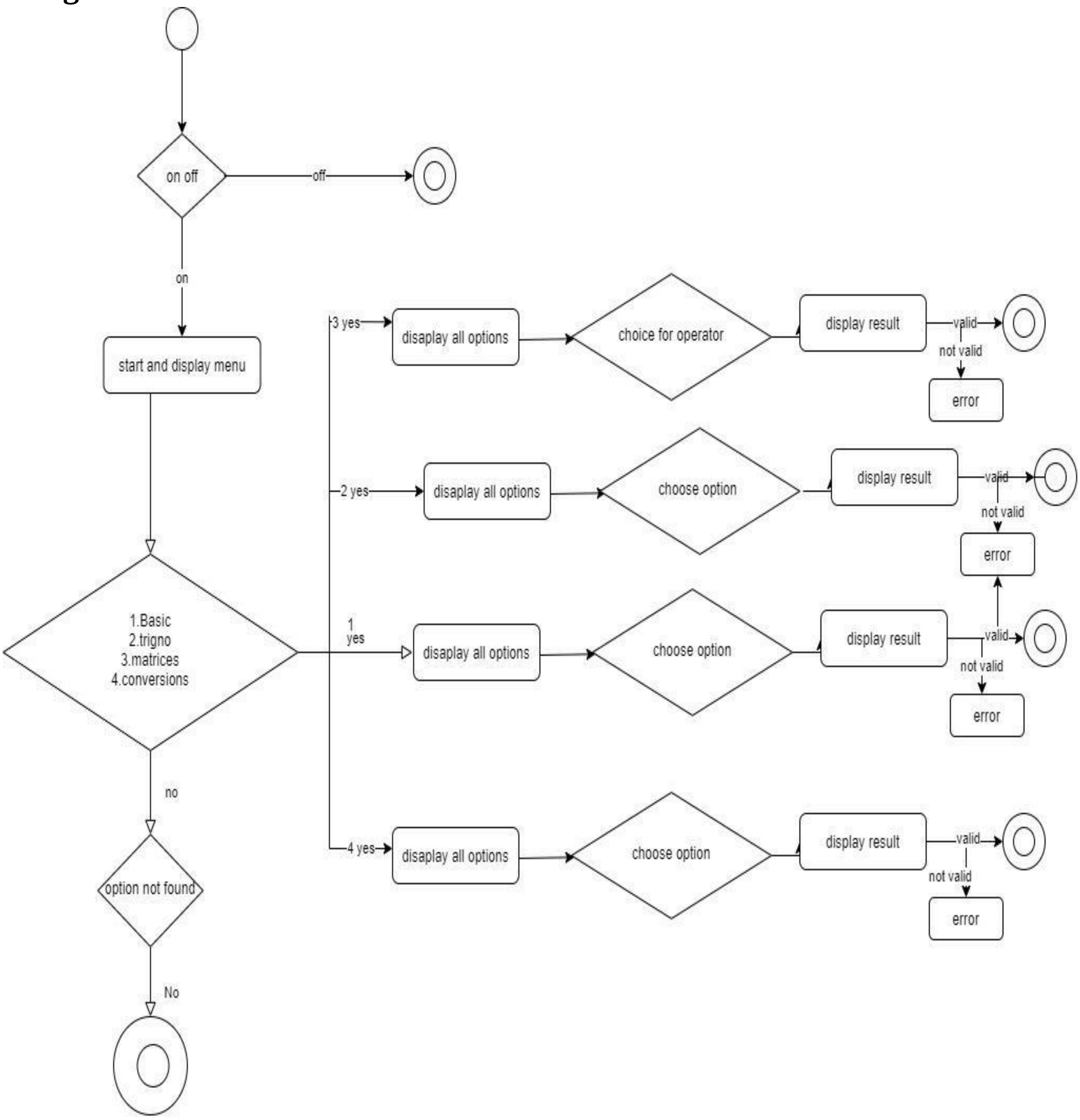
ID	Description	Status
HLR1	Basic Operations	Implemented
HLR2	Trigonometric Operations	Implemented
HLR3	Matrix Operations	Implemented
HLR4	Conversions	Implemented
HRL5	Advanced operations	Implemented

Low Level Requirements

ID	Description	Status
LLR1	Addition	Implemented
LLR2	Subtraction	Implemented
LLR3	Multiplication	Implemented
LLR4	Division	Implemented
LLR5	Sine	Implemented
LLR6	Cos	Implemented
LLR7	Tan	HLR2
LLR8	Cosec	HLR2
LLR9	sec	HLR2
LLR10	Cot	HLR2
LLR11	matrix addition	HLR3

ID	Description	Status
LLR12	matrix subtraction	HLR3
LLR13	matrix multiplication	HLR3
LLR14	Binary - Decimal Conversion	HLR4
LLR15	Decimal - Binary Conversion	HLR4
LLR16	Decimal - Octal Conversion	HLR4
LLR17	Octal - Decimal Conversion	HLR4
LLR18	Length unit Conversion	HLR4
LLR19	Temperature unit Conversion	HLR4
LLR20	Current AC-DC Conversion	HLR4
LLR21	Log	HLR5
LLR22	Exponential	HLR5
LLR23	Modulus (remainder)	HLR5
LLR24	Factorial	HLR5
LLR25	Square root	HLR5
LLR26	Cube root	HLR5
LLR27	LCM	HLR5
LLR28	GCD	HLR5
LLR29	Permutation	HLR5
LLR30	Combination	HLR5

Design


on off
off
on
start and display menu
1.Basic
2.trigno
3.matrices
4.conversions
no
option not found
No
3 yes
2 yes
1
yes
4 yes
disaplay all options
disaplay all options
disaplay all options
disaplay all options
choice for operator
choose option
choose option
choose option
display result
display result
display result
display result
valid
valid
valid
valid
not valid
not valid
not valid
not valid
error
error
error
error

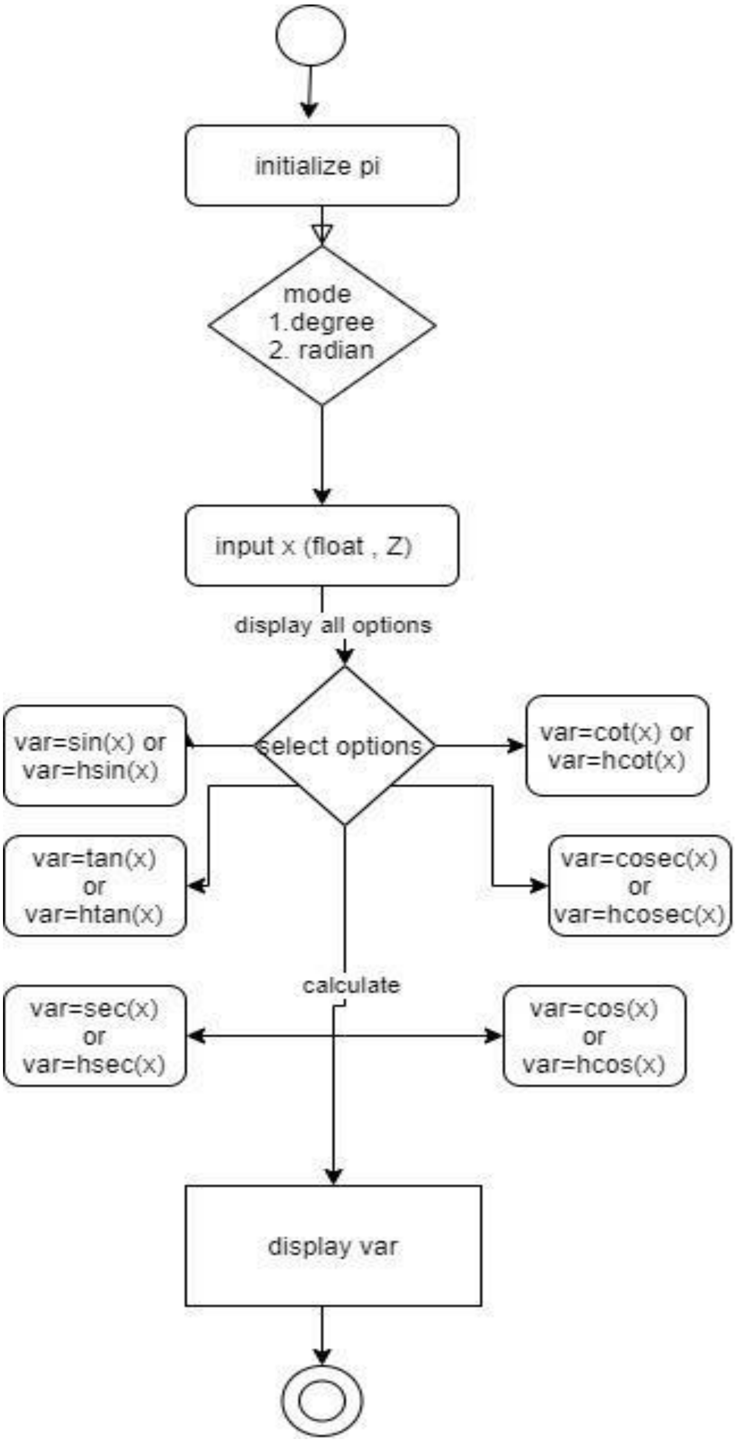


Figure 8 Behavioural Diagram

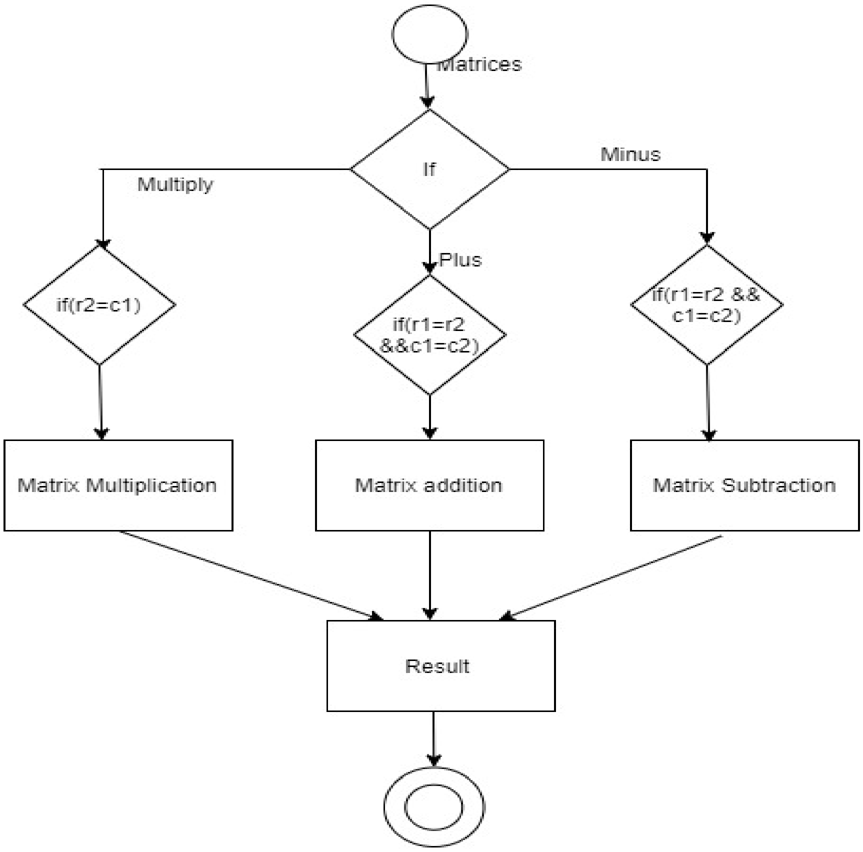


Figure 9 Behavioural Diagram of Matrix Operation

Test Plan

High Level Test Plan for Basic & Matrix Operation

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLR01	Check the number dived by zero	Given number is divided by zero	Error	Error	Requirement Based
HLR02	Check the base of log	Given number is other than 2,10	Error	Error	Requirement Based

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLR03	Check the number in square root	Given number is negative not positive	Error	Error	Requirement Based
HLR04	Check the number in cube root	Given number is negative	Negative number	Negative number	Requirement Based
HLR05	Check divided by big number	Given number is greater	Less than zero	Less than zero	Requirement Based
HLR06	Check divided by smaller number	Given number is smaller	Greater than zero	Greater than zero	Requirement Based
HLR07	Check divided by negative number	Given number is negative	Negative number	Negative number	Requirement Based
HLR08	R1!=R2 &C1!=C2	Invalid	Invalid	LLR8	Requirement Based
HLR09	R1=R2 &C1=C2	Do Matrix Addition	Do Matrix Addition	LLR8	Requirement Based
HLR10	R1!=R2 &C1!=C2	Invalid	Invalid	LLR9	Requirement Based
HLR11	R1=R2 &C1=C2	Do Matrix Subtraction	Do Matrix Addition	LLR9	Requirement Based
HLR12	R2!=C1	Invalid	Invalid	LLR10	Requirement Based

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLR13	R2=C1	Do Matrix Multiplication	Do Matrix Addition	LLR10	Requirement Based

Low Level Test Plan

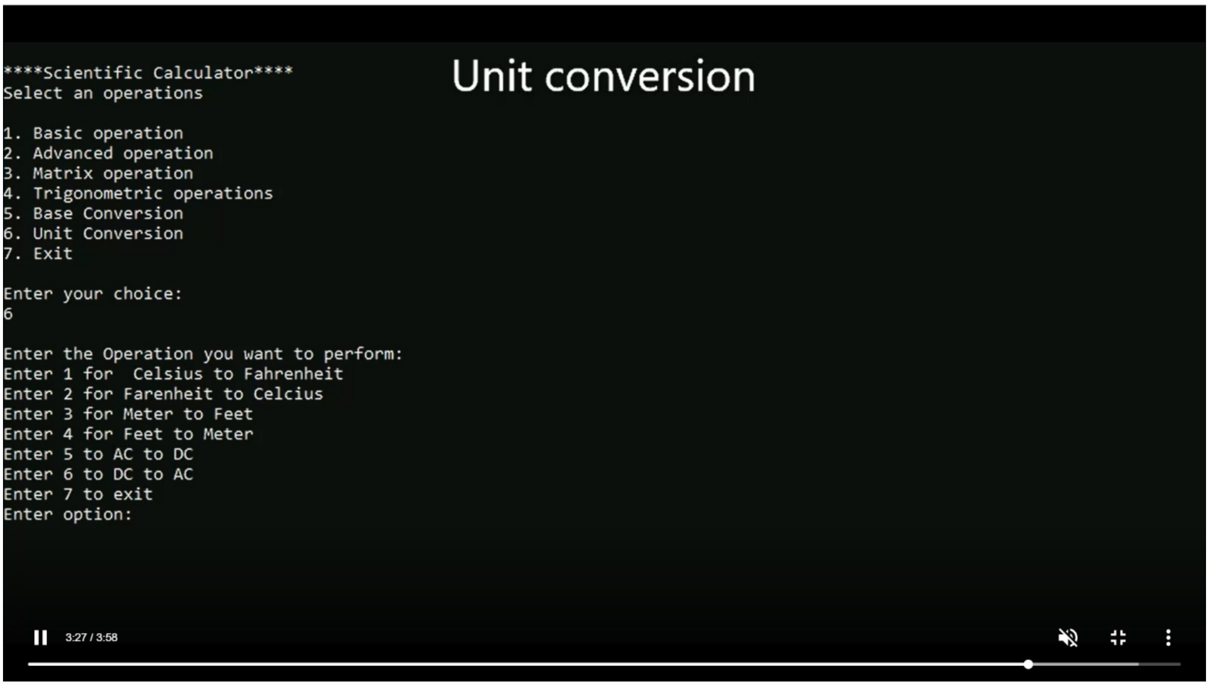
ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LLR1	sin(30)	-0.988032	Positive Test case	-0.988032	Requirement Based
LLR2	cos(30)	0.1542	Positive Test case	0.1542	Requirement Based
LLR3	tan(30)	-6.405331	Positive Test case	-6.405331	Requirement Based
LLR4	cot(30)	-0.15612	Positive Test case	-0.15612	Requirement Based
LLR5	sec(30)	6.4829	Positive Test case	6.4829	Requirement Based
LLR6	cosec(30)	-1.012113	Positive Test case	-1.012113	Requirement Based
LLR7	sin(60)	-0.304811	Positive Test case	-0.304811	Requirement Based
LLR8	cos(60)	-0.9524	Positive Test case	-0.9524	Requirement Based
LLR9	tan(60)	0.32004	Positive Test case	0.32004	Requirement Based
LLR10	cot(60)	3.124606	Positive Test case	3.124606	Requirement Based

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LLR11	sec(60)	-1.04996	Positive Test case	-1.04996	Requirement Based
LLR12	cosec(60)	-3.280726	Positive Test case	-3.280726	Requirement Based

Implementation and Summary

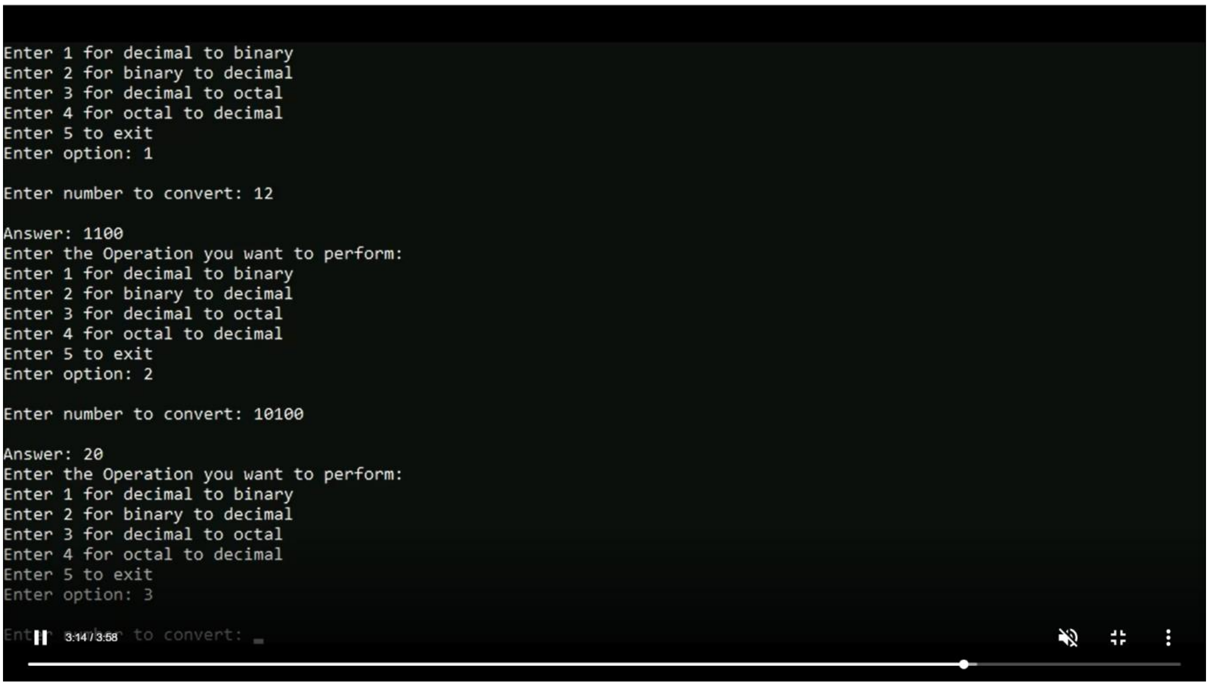
Git Link:

Link: https://github.com/GENESIS2021Q1/Applied_SDLC-Dec_Team_47

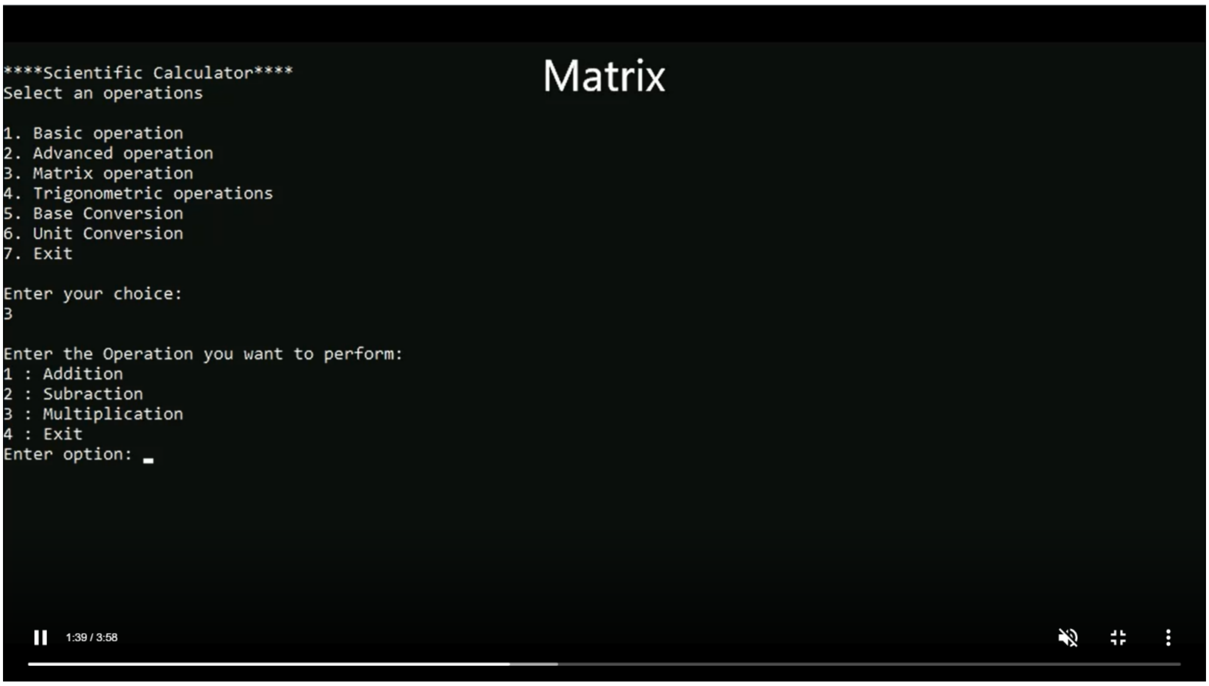


Unit conversion

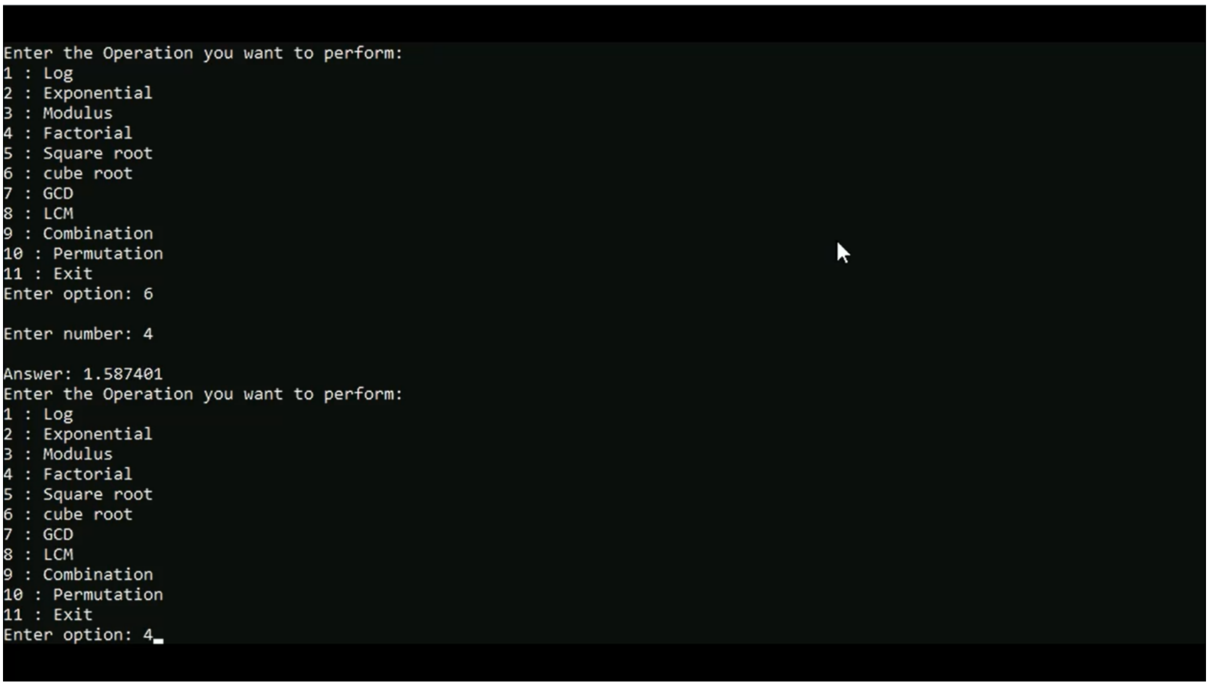
```
****Scientific Calculator****
Select an operations

1. Basic operation
2. Advanced operation
3. Matrix operation
4. Trigonometric operations
5. Base Conversion
6. Unit Conversion
7. Exit

Enter your choice:
6

Enter the Operation you want to perform:
Enter 1 for  Celsius to Fahrenheit
Enter 2 for Farenheit to Celcius
Enter 3 for Meter to Feet
Enter 4 for Feet to Meter
Enter 5 to AC to DC
Enter 6 to DC to AC
Enter 7 to exit
Enter option:
```



```
Enter 1 for decimal to binary
Enter 2 for binary to decimal
Enter 3 for decimal to octal
Enter 4 for octal to decimal
Enter 5 to exit
Enter option: 1

Enter number to convert: 12

Answer: 1100
Enter the Operation you want to perform:
Enter 1 for decimal to binary
Enter 2 for binary to decimal
Enter 3 for decimal to octal
Enter 4 for octal to decimal
Enter 5 to exit
Enter option: 2

Enter number to convert: 10100

Answer: 20
Enter the Operation you want to perform:
Enter 1 for decimal to binary
Enter 2 for binary to decimal
Enter 3 for decimal to octal
Enter 4 for octal to decimal
Enter 5 to exit
Enter option: 3

Enter number to convert: _
```

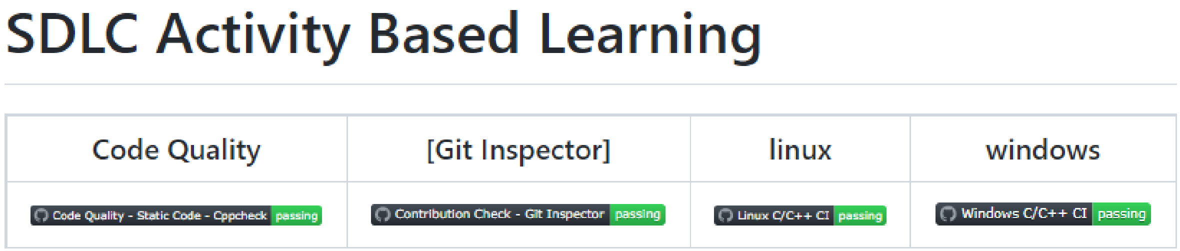


```
****Scientific Calculator****
Select an operations

1. Basic operation
2. Advanced operation
3. Matrix operation
4. Trigonometric operations
5. Base Conversion
6. Unit Conversion
7. Exit

Enter your choice:
3

Enter the Operation you want to perform:
1 : Addition
2 : Subraction
3 : Multiplication
4 : Exit
Enter option: _
```



```
Enter the Operation you want to perform:
1 : Log
2 : Exponential
3 : Modulus
4 : Factorial
5 : Square root
6 : cube root
7 : GCD
8 : LCM
9 : Combination
10 : Permutation
11 : Exit
Enter option: 6

Enter number: 4

Answer: 1.587401
Enter the Operation you want to perform:
1 : Log
2 : Exponential
3 : Modulus
4 : Factorial
5 : Square root
6 : cube root
7 : GCD
8 : LCM
9 : Combination
10 : Permutation
11 : Exit
Enter option: 4_
```

Git Dashboard



SDLC Activity Based Learning

Code Quality	[Git Inspector]	linux	windows
Code Quality - Static Code - Cppcheck passing	Contribution Check - Git Inspector passing	Linux C/C++ CI passing	Windows C/C++ CI passing

Challenges Faced and How Was It Overcome

1. Running the make file as its resolved by defining its correct path(.out for Linux and -lm for math functions)
2. Synchronizing the VS code to GitHub , colleague help to resolve the issue
3. Making the function call in correct path
4. Open git log while committing, thus went to GitHub desktop and pulled origin and then pushed origin.
5. Test case code for the boundary problem. Added code with the help of internet

Mini Project 4 – Calendar Automation [Team]

Modules

1. Python
2. Git

Requirements

High Level Requirements

ID	Feature	MATLAB v.0 Status	Python v.0 Status
HR01	GUI	Implemented	Implemented
HR02	Master calendar	Implemented	Implemented
HR03	Faculty calendar	Implemented	Implemented
HR04	Faculty load sheet	Implemented	Implemented
HR05	Showing Available Open Slots based on faculty and modules	Not Available	Not Available
HR06	Output file generated across different computers (windows + Linux)	Not Available	Implemented
HR07	Visualizing data to create Meaningful Insights	Not Available	Not Available
HR08	Calculate Individual Faculty Load	Implemented	Implemented

Low Level Requirements

ID	Feature	High Level ID	MATLAB v.0 Status	Python v.0 Status
LR01	GUI should allow user to login using credentials	HR01	Not Available	Not Available

ID	Feature	High Level ID	MATLAB v.0 Status	Python v.0 Status
LR02	Input Files Based on Different Initiatives and Timelines	HR01	Implemented	Not Available
LR03	GUI should get Base Calendar as Input	HR01	Implemented	Implemented
LR04	GUI should get Month and Initiative as Input	HR01	Implemented	Implemented
LR05	GUI should be able to show Conflicts/Warnings	HR01	Implemented	Not Implemented
LR06	Master Calendar: display Month wise	HR02	Implemented	Implemented
LR07	Master Calendar: display Initiative wise	HR02	Implemented	Not Available
LR08	Master Calendar: Differentiate Initiatives (Colour Codes/Numbers)	HR02	Implemented	Implemented
LR09	Master Calendar: Appending	HR02	Implemented	Not Available
LR10	Master Calendar: Course code correction	HR02	Implemented	Not Available

Link for template standard input template :

https://docs.google.com/spreadsheets/d/1EWYp_1iyK2wLMfKGJOiTJAk5WexZusCP/edit?usp=sharing&ouid=113003694561146884677&rtpof=true&sd=true

- Using the template above, training schedule can be added monthwise and initiatives wise

- The name of the input excel sheet MUST be named as "Test_vector"(as shown in template)
- Along with the Test_vector sheet, "Key" sheet MUST be present under the columns assigned as in the template
- The "Key" sheet must contain all times the 6 fixed initiatives with their respective codes and total list of course code and course title in order to refer for corrections while writing to output files
- Appending additional slots for existing courses is possible by adding just the additional slots in the input file for the same course

Requirements for updating Master calendar using Master calendar as input

Link for template

2 Slots format - M/A : https://docs.google.com/spreadsheets/d/1jtKnXV12VE1fH20CGDo4B3uNWRTAhQCWz-hHUDWUe3I/edit?usp=sharing

4 Slots format - M1/M2/A1/A2 : https://docs.google.com/spreadsheets/d/1jVheSPZkOtfNKRNoc_858nwk2UaHCe0gExTNZfZ8vxA/edit?usp=sharing

- Any of the two templates can be used for updating Master calendar month wise on to the drive
- The blocked slots must have the corresponding initiative code in the cell according to the key as shown in the sample data in the template
- The name of the sheet must be the name of the month to be updated
- The "Key" sheet must be present with the fixed list of initiatives and initiative code

App deployment

- The app is deployed on heroku servers.
- To add/modify new features, you will be required to install HEROKU CLI link
- After installation, open terminal in working directory and enter the following commands:
  - "heroku git:clone -a gea calendar"

  - login using heroku credentials
- After pulling and making changes, enter the following commands to push app and deploy on server
  - Git add.
  - git commit -m "commit message"
  - git push Heroku master

Additional features for V1 to do

- Update key sheet by appending new initiatives/courses list
- Check for duplicate course entries in input file
- Using built in libraries to identify number of days in month, current year and highlight weekend and holidays
- Function to remove a course schedule
- Read multiple months data in one sheet as input file (currently takes data one by one month)
- Calculate individual faculty load

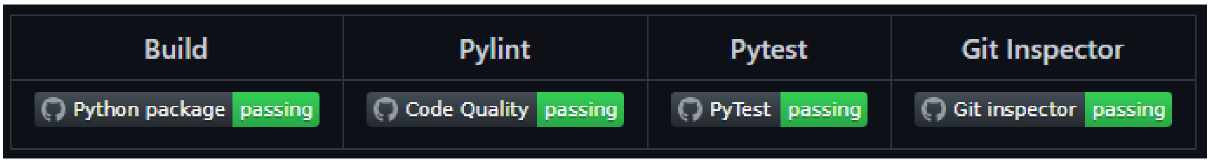
Git Link:

Implementation and Summary

Git Link:

Link: Usharani8/Oopswithpython_Calendar_Automation_Team-47 (github.com)

Git Dashboard



Build	Pylint	Pytest	Git Inspector
Python package passing	Code Quality passing	PyTest passing	Git inspector passing

Figure 10 Git Dashboard

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

Mini Project 5 – BMW Project [Team]

Modules

1. MATLAB
2. Git

Requirements

We have implemented following features

1. Adaptive Cruise Control System
2. Anti-Lock Braking System
3. Interior Light Control System
4. Door Locking system
5. Suspension System
6. HVAC System
7. Power Window

Individual Feature Requirements

Door Locking System

High Level Requirements: -

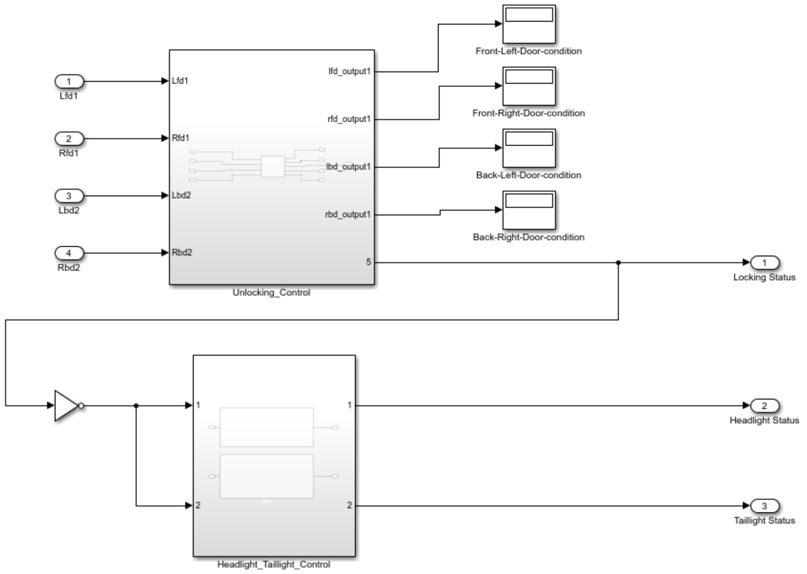
ID	Description
HLR1	The lock button on the key should lock all the doors of the car.
HLR2	The unlock button on the key should unlock all the doors of the car.
HLR3	After locking exterior lights of the car should get pop up for 3 seconds
HLR4	Exterior lights should get pop up after unlocking the car.

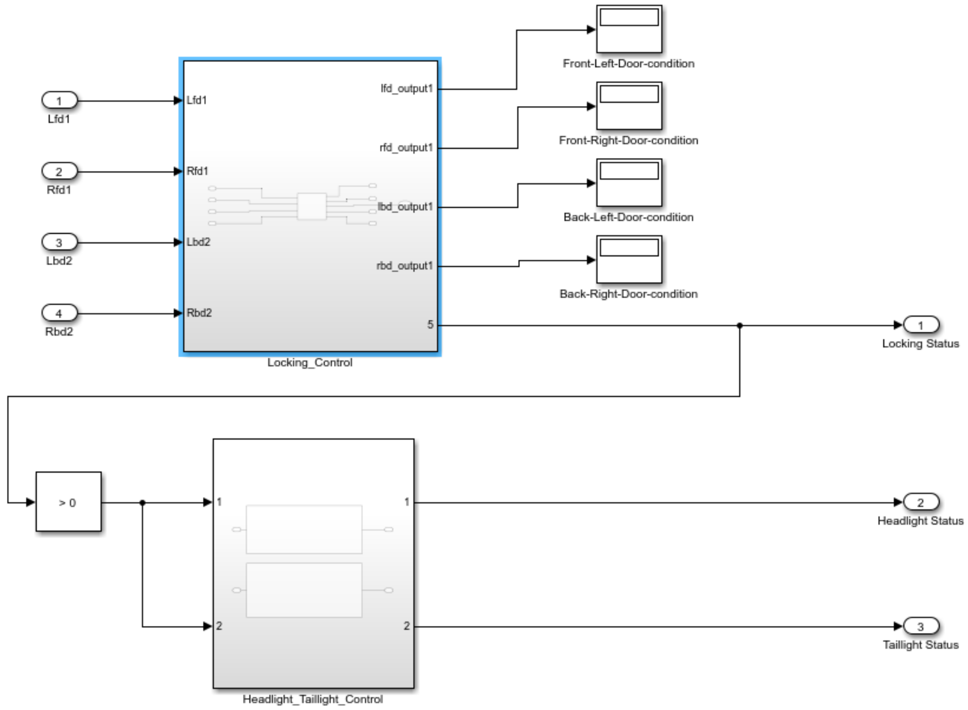
Low Level Requirements: -

ID	Description
LLR1	Status of the door should have to be visible on the display
LLR2	Locking and the exterior light pop up should be in synchronised manner.
LLR3	Unlocking and the exterior light pop up should be in synchronised manner.
LLR4	If any door is opened it should indicate on the display.

Design

Locking & Unlocking System:

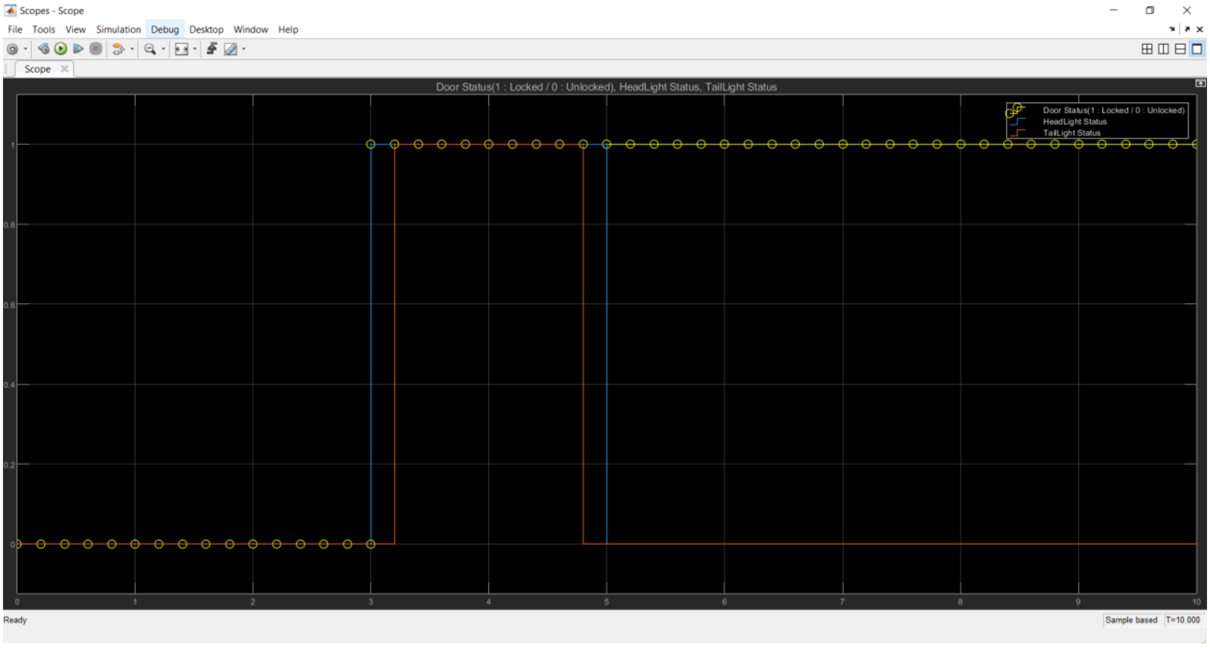




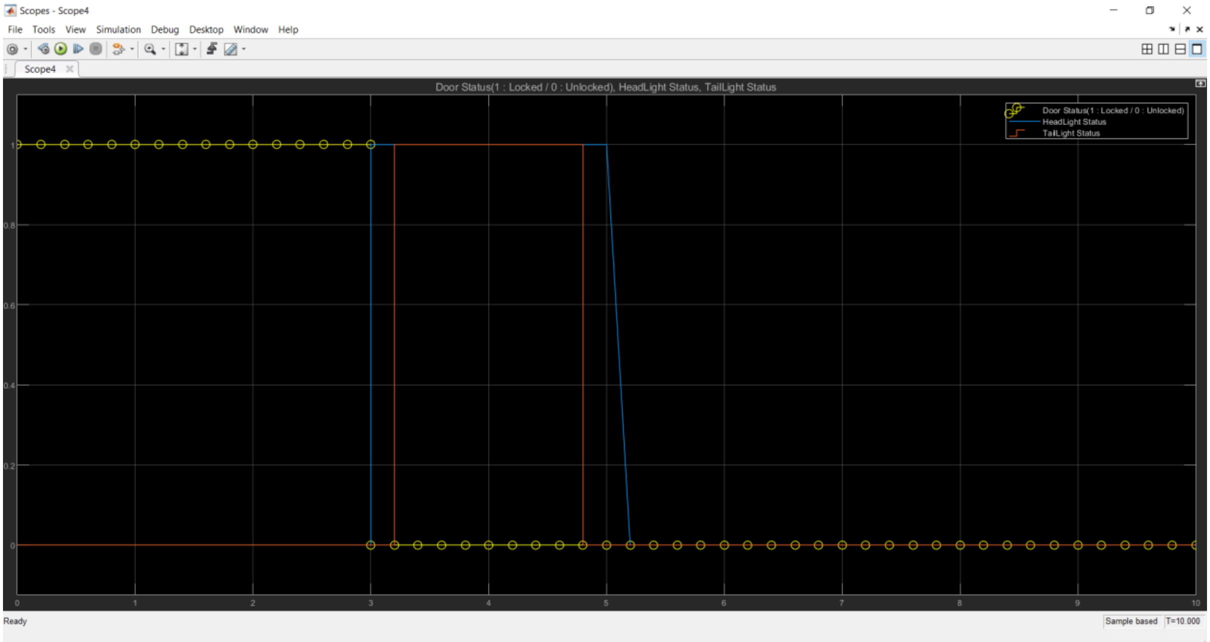
OUTPUT

Running the Simulation in ABS Mode

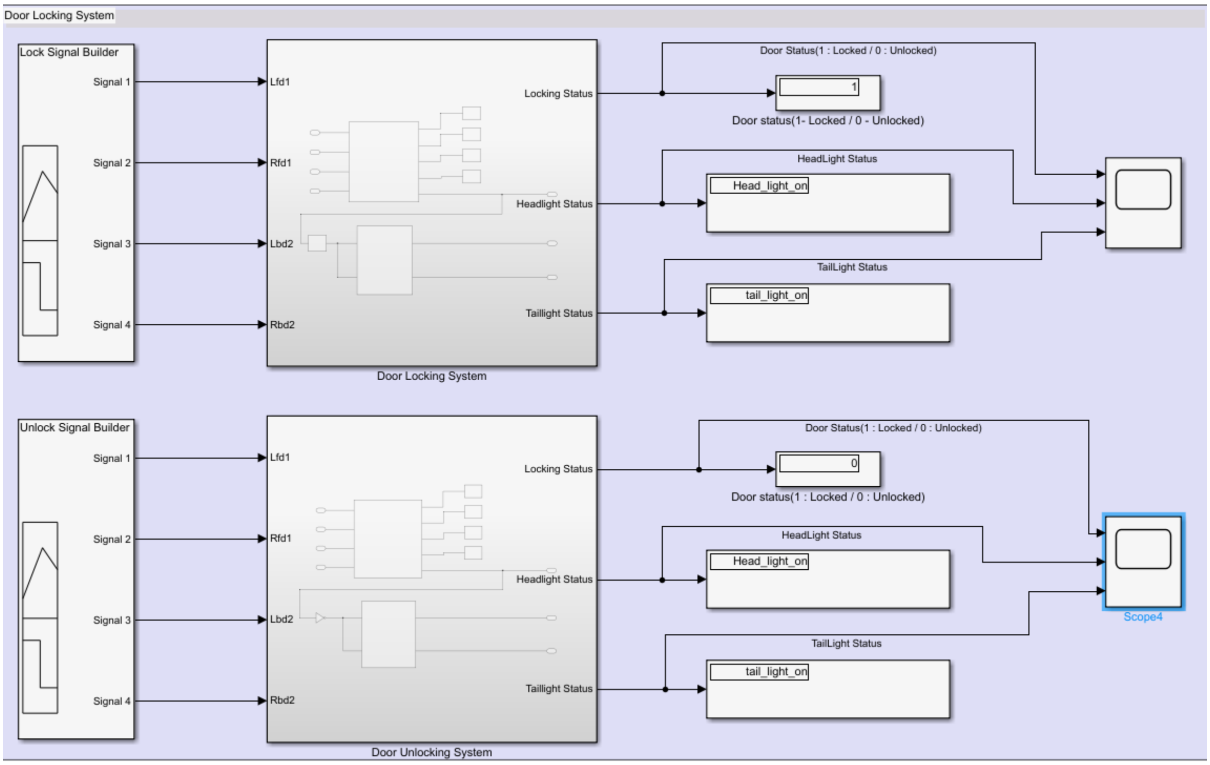
- Vehicle Locking System:


Scopes - Scope
File Tools View Simulation Debug Desktop Window Help
Scope
Door Status(1 : Locked / 0 : Unlocked), HeadLight Status, TailLight Status
Door Status(1 : Locked / 0 : Unlocked)
HeadLight Status
TailLight Status
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
Ready
Sample based
T=10.000

- Vehicle unlocking system:


Scopes - Scope4
File Tools View Simulation Debug Desktop Window Help
Scope4
Door Status(1 : Locked / 0 : Unlocked), HeadLight Status, TailLight Status
Door Status(1 : Locked / 0 : Unlocked)
HeadLight Status
TailLight Status
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
Ready
Sample based
T=10.000

- <u>Overall output with exterior light status:</u>



Mini Project 6 – Wiper Control [Team]

Modules

1. C Programming
2. STM32

Requirements

4W's and 1'H

Who:

Users who drives the vehicles can use this.

What:

This project is concerned is about automatic wiper system in vehicles.

When:

When there is a change in the weather the wipers work automatically.

Where:

This projects helps the users to achieve the clear path when there is a change of weather.

How:

The wiper system is controlled using rain sensor, temperature sensor and SMT32 microcontroller

High Level Requirements

ID	Description	Status
HLR_1	Press and hold the button to put the Ignition key position in ACC mode	Implemented
HLR_2	Different wiper frequencies to be set (1Hz, 4Hz & 8Hz)	Implemented
HLR_3	Hold the button to put the system in Idle state	Implemented

Low Level Requirements

ID	Description	HLTP ID	Status
LLR_1	Hold the button for 2 sec to bring the ignition key position at ACC mode	HLR_1	Implemented
LLR_2	Hold the button for 2 sec to go back to the Idle state	HLR_1, HLR_3	Implemented
LLR_3	Press the button one time to set frequency to 1Hz	HLR_2	Implemented
LLR_4	Press the button second time to set frequency to 4Hz	HLR_2	Implemented
LLR_5	Press the button third time to set frequency to 8Hz	HLR_2	Implemented
LLR_6	Press the button fourth time to turn OFF the wiper action	HLR_2	Implemented
LLR_7	Hold the button for 2 sec to bring ignition key position at Lock state	HLR_3	Implemented

Design

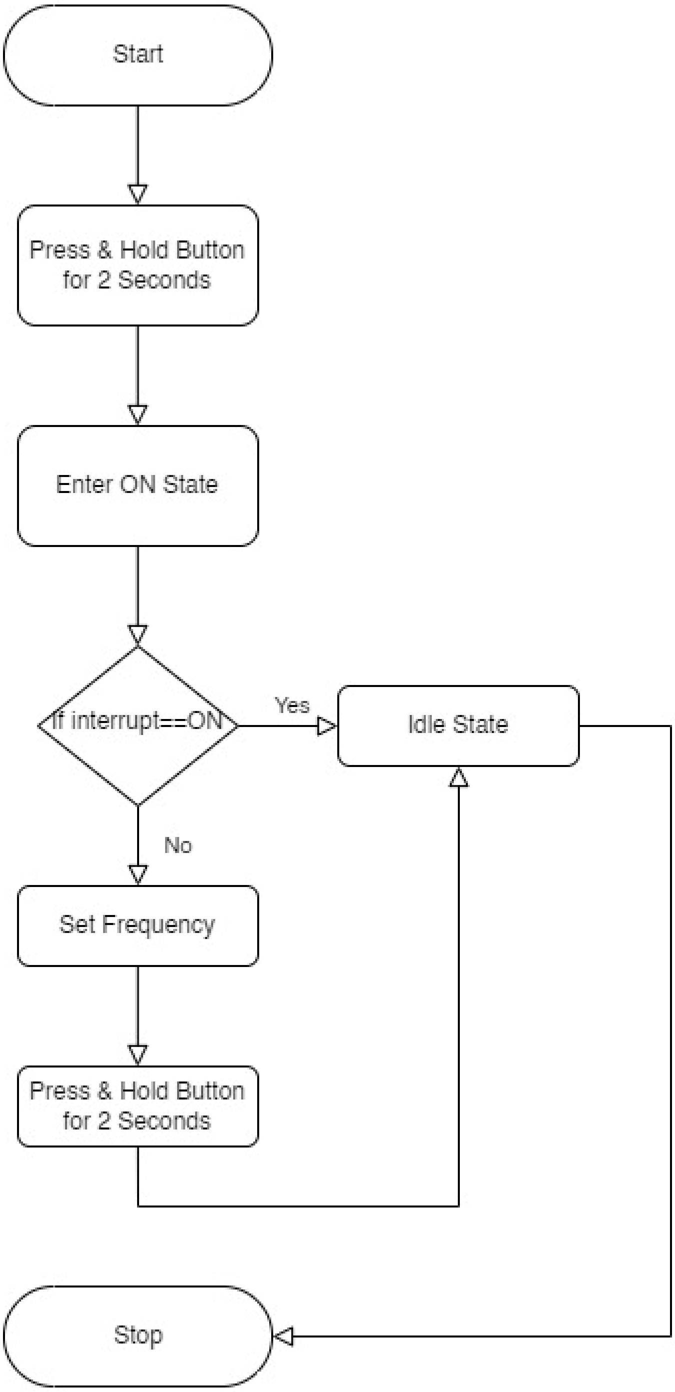


Figure 11 Structural Diagram

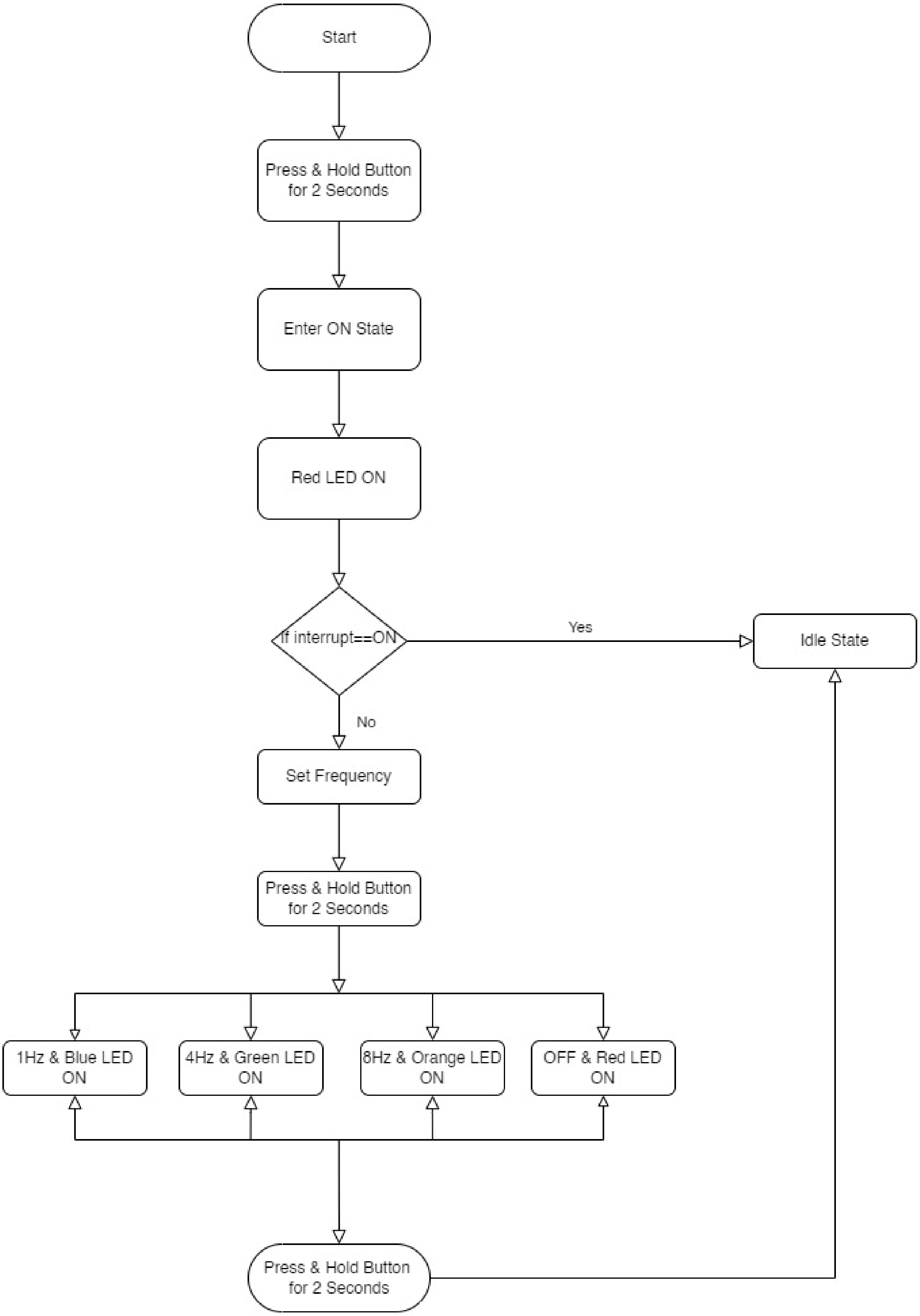


Figure 12 Behavioural Diagram

Test Plan

High Level Test Plan

TEST PLAN ID	Description	Expected Input	Expected Output	Status
HLR01	Check Red LED on & wiper work	Press and held 2sec	Red LED On & Wiper works	pass
HLR02	Press button 1 more time	press button	Blue LED ON	Pass
HLR03	Hold the button to put the system in Idle state	Enters Idle State	Requirement Based	
HLR03	Check the frequency of blue LED	1HZ	Wiper receive 1HZ	Pass
HLR04	Press the button again 1 more time	press button	Green LED ON	Pass
HLR05	Check the frequency of Green LED	4HZ	wiper receive 4HZ	Pass
HLR06	Press the button 1 more time again	Press Button	Orange LED ON	Pass
HLR07	Check the frequency of Orange LED	8HZ	Wiper receive 8HZ	Pass
HLR08	Check the frequency of LED & check wiper work	5,6,7HZ	Wiper receive 5HZ for blue,6HZ for green & 7HZ for orange	Pass

Low Level Test Plan

LLR ID	Description	HLR ID	Expected Input	Expected Output	Status
LLR01	Run the system	HLR01	Check the LEDs & wiper	LEDs off & wiper off	Pass
LLR02	Press the button for 2sec	HLR01	Press and held 2sec	Red LED will be ON	Pass
LLR03	Press the button for 1sec	HLR01	Press and held 1sec	Red LED will be OFF	Pass
LLR04	Press the button for 3sec	HLR01	Press and held more then 3sec	Red LED will be OFF	Pass
LLR05	Press the button for 2sec	HLR01	Press and held 2sec	Red LED will be OFF	Fail
LLR06	After red LED ON wiper is also ON	HLR01	Wiper ON	Wiper will start working	Pass
LLR07	After red LED ON wiper is also ON	HLR01	Wiper OFF	Wiper will stop working	Fail
LLR08	Press & check the blue LED	HLR02	press button	Blue LED ON	Pass
LLR09	Check the frequency of wiper working	HLR03	1HZ	Wiper gets 1HZ & work at 1HZ speed	Pass

LLR ID	Description	HLR ID	Expected Input	Expected Output	Status
LLR10	Check the frequency after some time	HLR03	1HZ	Wiper work at 1HZ range only	Pass
LLR11	Check the frequency after some time	HLR03	1HZ	Wiper working more then 1HZ	Fail
LLR12	Press the button & Green LED ON	HLR04	Press button	Green LED ON	Pass
LLR13	Press the button & Green LED ON	HLR04	Press button	Green LED OFF	Fail
LLR14	Check the frequency wiper receiving	HLR05	4HZ	Wiper should receive & work at 4HZ	Pass
LLR15	Check the frequency after some time	HLR05	4HZ	Wiper working at 4HZ	Pass
LLR16	Check the frequency after some time	HLR05	4HZ	Wiper working more then 4HZ	Fail
LLR17	Press the button & Orange LED ON	HLR06	Press button	Orange LED ON	Pass

LLR ID	Description	HLR ID	Expected Input	Expected Output	Status
LLR18	Press the button & orange LED ON	HLR06	Press button	Orange LED OFF	Fail
LLR19	Check the frequency wiper receiving	HLR07	8HZ	Wiper works at same 8HZ level	Pass
LLR20	Check the frequency wiper receiving	HLR07	8HZ	Wiper works mare then 8HZ level	Fail
LLR21	Check the frequency for each LEDs	HLR08	5,6,7HZ	New frequency set for Blue,Green,Orange	Pass
LLR22	Repeat steps from LLR08 to LLR21	HLR09	same i/p	Same O/P	Same status
LLR23	Press & held button for 2sec & Red LED OFF	HLR09	Press button for 2sec	Red LED OFF	Pass
LLR24	Press & held button for 2sec & Red LED OFF	HLR09	Press button for 2sec	Still ON Red LED	Fail
LLR25	Check the wiper after Red LED OFF	HLR09	Press button for 2sec	Wiper stops working	Pass

LLR ID	Description	HLR ID	Expected Input	Expected Output	Status
LLR26	Check the other LED's too	HLR09	Press button for 2sec	Blue,Green,Orange LED's ON	Pass
LLR27	Press the button again & again check the LED's	HLR10	Press button again & again	Blue,Green,Orange LED's ON	Pass
LLR28	Press the button again & again check the LED's	HLR10	Press button again & again	Blue,Green,Orange LED's ON	Fail

Implementation and Summary

Git Link:

Link: GENESIS-2022/MasteringMCU-Team33: Details (github.com)

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

Mini Project 7 – Automotive BMW Project [Team]

Modules

1. Automotive Systems
2. Git

Requirements

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

1. Power Window Control System
2. Mirror Control System
3. Door Locking System
4. Interior Light Control System

Power Window Control System

High Level Requirement	Description
HLR1	The Front Right(FR) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
HLR2	The Front Left(FL) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
HLR3	The Back Right(BR) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
HLR4	The Back Left(BL) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
HLR5	The master control on driver side door should be able to grant or deny access to open or close other windows in the car.
HLR6	The auto-lower & auto-raised feature is triggered if the driver's window lower switch is held down for more than 1 second.

High Level Requirement	Description
HLR7	After pressing the door lock button on the key, if any window is opened should get automatically closed.
HLR8	The windows should get automatically raised according to the humidity and cooling status of the AC as per need.
HLR9	If the seatbelt is not wore by the driver while the car is above 20 km/h, access to the power windows control should get removed.
HLR10	While raining, by gathering wiper control data, windows should get raised automatically.
HLR11	Due to anti-pinch feature, if the system senses any obstacle in the path of the glass the window glass should be stopped moving further up.

UML Diagram

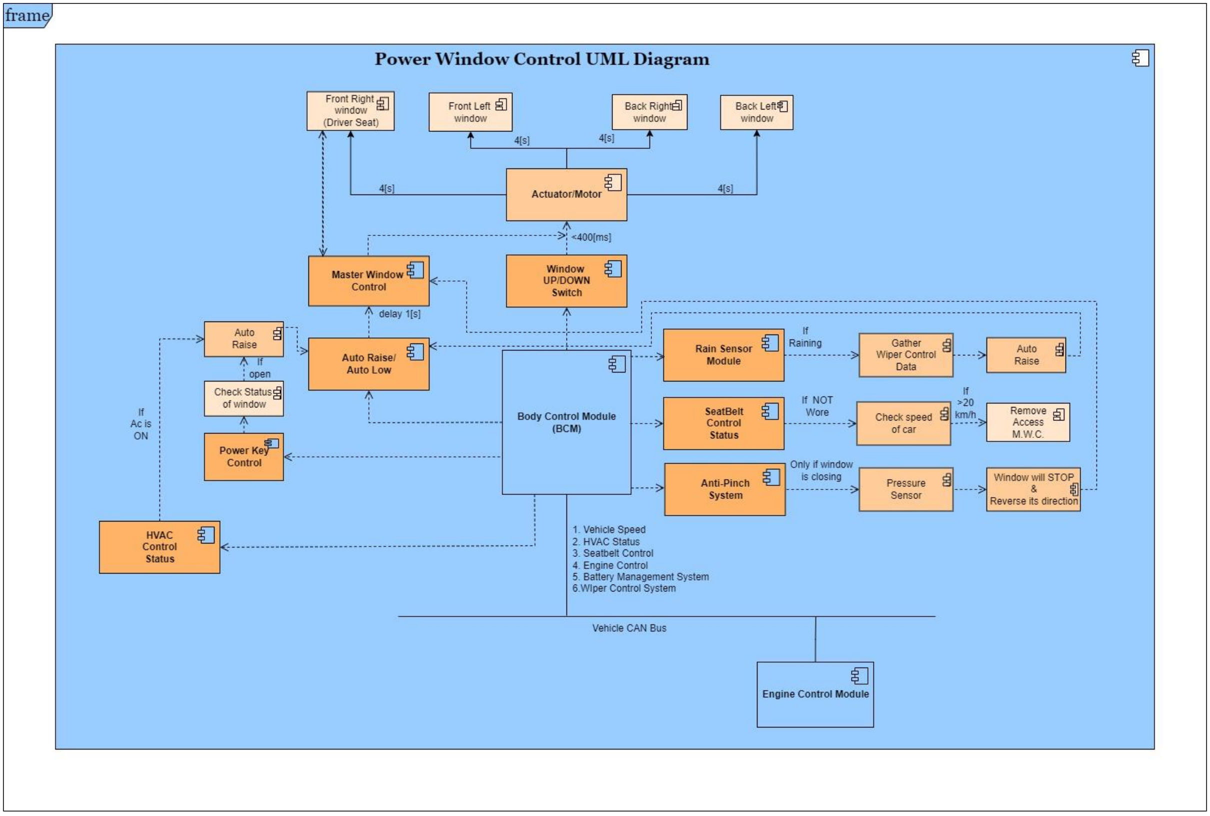


Figure 13 Structure Diagram

Implementation and Summary

Git Link:

Link: Nayan349/Automotive_BMW_Project (github.com)

Individual Contribution and Highlights

1. Power Window Case Study
2. Source code management using GitHub

Role in Project Team

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

Mini Project 8 – Golf Cart [Team]

Modules

1. MATLAB
2. MATLAB Script

HISTORY

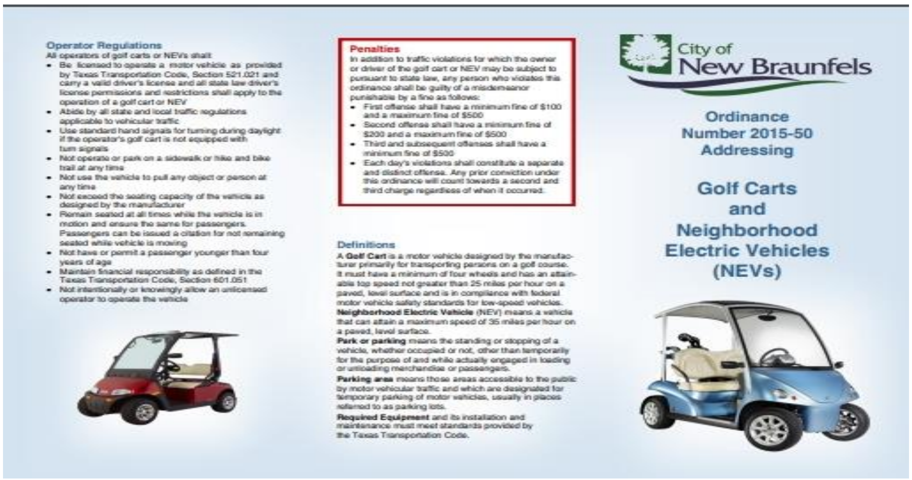
Reportedly, the first use of a motorized cart on a golf course was by JK Wadley of Texarkana, Texas/Arkansas, who saw a three-wheeled electric cart being used in Los Angeles to transport senior citizens to the grocery store. Later, he purchased a cart and found that it worked poorly on a golf course.The first electric golf cart was custom-made in 1932, but did not gain widespread acceptance. In the 1930s until the 1950s the most widespread use of golf carts was for those with disabilities who could not walk far. By the mid 1950s the golf cart had gained wide acceptance with US golfers, with several manufactures (e.g. Victor Adding Machine Co. and Sears Roebuck) producing various models. Most were electric.

ABSTRACT

In this paper, the Design & Calculations of an Electric golf cart is done, the calculations and process to develop a golf cart chassis, electrical transmission using transaxle, braking, steering geometry, suspension system, solar panel of a long six seat golf cart, battery calculations and controller which monitors the electric system is explained. All the materials used for the calculation are industry standard being utilized currently. Priority of this design was to keep the cart as light weight as possible and rigid with good performance on grassy and concrete terrain's used in either golf course or in hospitality education sectors. A unique tubular chassis is used in the design comprising of circular & rectangular shape. The focus of design was long mileage from the batteries used and how much, adding solar panels can contribute to the existing range of the electric vehicle. A 23 miles range from the battery pack using 6 deep cycle lead acid batteries was achieved and increased12 miles from the solar panels under the constraint assumed for its use (direct sunlight for over 6-7 hours)

INTRODUCTION

This projects aim is to Design an electric golf cart with six people seating capacity which runs on batteries as well as solar panels. The need for a golf cart isn't just using it in the golf course and is nowadays being used in large universities spread over vast acres of land and in expensive hotels and latest complexes for comfort traveling i.e Hospitality industry and providing low cost of charging with increasing mileage as technology advances. In this project the aim has been to put forth an industry standard design for an electric golf cart that can run exceedingly well on grassy / muddy terrains as well as wet conditions and concrete roads. During designing all the constraints and cost considerations and all technical aspects were set according to the current advancements in the industry in a thoroughly researched manner. A battery electric vehicle (BEV) has far fewer moving parts than a conventional gasoline-powered vehicle. There's no need for a liquid fuels or oil changes. There's no transmission or timing belt to fail you. And most of the maintenance cost associated with ic engine are eliminated.



Operator Regulations

All operators of golf carts or NEVs shall:

- Be licensed to operate a motor vehicle as provided by Texas Transportation Code, Section 521.021 and carry a valid driver's license and all state law driver's license permissions and restrictions shall apply to the operation of a golf cart or NEV
- Abide by all state and local traffic regulations applicable to vehicular traffic
- Use standard hand signals for turning during daylight if the operator's golf cart is not equipped with turn signals
- Not operate or park on a sidewalk or hike and bike trail at any time
- Not use the vehicle to pull any object or person at any time
- Not exceed the seating capacity of the vehicle as designed by the manufacturer
- Remain seated at all times while the vehicle is in motion and ensure the same for passengers. Passengers can be issued a citation for not remaining seated while vehicle is moving
- Not have or permit a passenger younger than four years of age
- Maintain financial responsibility as defined in the Texas Transportation Code, Section 601.051
- Not intentionally or knowingly allow an unlicensed operator to operate the vehicle

Penalties

In addition to traffic violations for which the owner or driver of the golf cart or NEV may be subject to pursuant to state law, any person who violates this ordinance shall be guilty of a misdemeanor punishable by a fine as follows:

- First offense shall have a minimum fine of $100 and a maximum fine of $500
- Second offense shall have a minimum fine of $200 and a maximum fine of $500
- Third and subsequent offenses shall have a minimum fine of $500
- Each day's violations shall constitute a separate and distinct offense. Any prior conviction under this ordinance will count towards a second and third charge regardless of when it occurred.

Definitions

A **Golf Cart** is a motor vehicle designed by the manufacturer primarily for transporting persons on a golf course. It must have a minimum of four wheels and has an attainable top speed not greater than 25 miles per hour on a paved, level surface and is in compliance with federal motor vehicle safety standards for low-speed vehicles.

Neighborhood Electric Vehicle (NEV) means a vehicle that can attain a maximum speed of 35 miles per hour on a paved, level surface.

Park or parking means the standing or stopping of a vehicle, whether occupied or not, other than temporarily for the purpose of and while actually engaged in loading or unloading merchandise or passengers.

Parking area means those areas accessible to the public by motor vehicular traffic and which are designated for temporary parking of motor vehicles, usually in places referred to as parking lots.

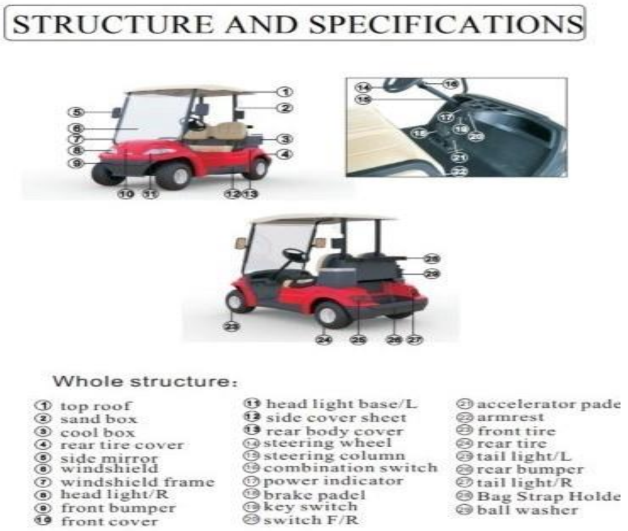
Required Equipment and its installation and maintenance must meet standards provided by the Texas Transportation Code.

City of New Braunfels

Ordinance Number 2015-50 Addressing

Golf Carts and Neighborhood Electric Vehicles (NEVs)

REQUIREMENTS



Range

Vehicle Model	**Buggy**
Colour	**Wide range of Colour options**
Usage/Application	**Eco-friendly drive within enclosed campuses.**
Seating Capacity	**4-12 persons**
Running Distance	**40 Km/Charge**
Maximum Speed	**11KM/Hour**
Brand	**Maini**
Material	**Powder coated tubular steel Chassis, Body frame - dent proof ABS**
Number Of Battery	**6 no, 8 Volt each, 150AH**
Voltage	**48V**

Motor Rated	**48 Volt AC Motor**
Power	**4 Kw Continuous**
Speed	**11KM/Hour**
Seater	**2-14 seater**
Capacity Kg	**200-1100 KG**

BATTERY

Brand	**Trojan**
Capacity	**@20Hr - 170Ah**
Model Name/Number	**Trojan Motive T-875 with Bayonet Cap**
Voltage	**8V**
Battery Type	**Deep-Cycle Flooded/Wet Lead-Acid Battery**
Dimensions	**10.27 x 7.10 x 11.14 Inches**
Weight	**29 Kg**
Material	**Polypropylene**
Application / Usage	**Golf cart, Low Speed Electric Vehicle**

EZGO marathon Vs YAMAHA UMAX rally 2+2:

Motor Specifications:

Component	**EZGO marathon**	**Y- UMAX rally 2+2**
Engine Type	350cc twin cylinder unit 18 cubic inches	402cc low-emission single cylinder 60 degree incline OHV
Fuel Tank Capacity	Twin cylinder unit	5.2US GAL (20 LITERS)
Top speed	12-14 Nm	15 mph (24.1 km/hr)
Minimum Turning Radius	4.24m	3.98m
Maximum Forward Speed	12mph (19.3 km/h)	15mph (24.1 km/h)

Battery Specifications:

Component	**EZGO marathon**	**Y- UMAX rally 2+2**
Battery Type	Works on gas cylinder	Lithium-ion
Range	25-30 miles	35 miles
Battery Charging Time	Works on gas cylinder	4 Hours

Battery Capacity	Works on gas cylinder	2 kWh
No of Cells	Works on gas cylinder	20,500

Dimensions:

Component	EZGO marathon	Y- UMAX rally 2+2
Overall Length	125.5 in (317.3cm)	134.37 in (341.3cm)
Overall Width	50.3 in (127.6cm)	53.54 in (136cm)
Overall Height	72.7 in (182.4cm)	77.20 in (196.1cm)
Wheelbase	72.2 in (182.1cm)	77.28 in (196.3cm)
Front Wheel Tread	38.5 in (96.8cm)	41.26 in (104.8cm)
Rear Wheel Tread	37.2 in (94.1cm)	41.26 in (104.8cm)
Minimum Ground Clearance	5.1 in (12.7cm)	5.94 in (15.1cm)

Wheel Specifications:

Wheel Type	EZGO marathon	Y- UMAX rally 2+2
Front Wheel Size	10 inches	12 inches
Rear Wheel Size	11 inches	12 inches
Front Tyre Size	22*9-11*10.5 - ply monitor k272	23*10.5-12*4- ply monitor k389

Dimensions & Chassis Specifications:

Component	EZGO marathon	Y- UMAX rally 2+2
Kerb Weight	125 kg	108 kg
Overall Length	1,859 mm	1,800 mm
Overall Width	712 mm	700 mm
Overall Height	1,160 mm	1,250 mm

Wheelbase	1,359 mm	1,278 mm
Ground Clearance	165 mm	160 mm
Seat Height	792 mm	765 mm
Chassis Type	Tubular	Precision Machined Hybrid Chassis

Implementation and Summary

Submission: Submitted in GEA Learn

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

Mini Project 9 – Power Window Control [Individual]

Modules

1. AutoSAR
2. Git

Requirements

S.NO	Description
1	The Front Right(FR) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
2	The Front Left(FL) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
3	The Back Right(BR) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
4	The Back Left(BL) window glass should get raised or lowed after the switch of the door is pulled up or pushed.
5	The master control on driver side door should be able to grant or deny access to open or close other windows in the car.
6	The auto-lower & auto-raised feature is triggered if the driver's window lower switch is held down for more than 1 second.
7	After pressing the door lock button on the key, if any window is opened should get automatically closed.
8	The windows should get automatically raised according to the humidity and cooling status of the AC as per need.
9	If the seatbelt is not wore by the driver while the car is above 20 km/h, access to the power windows control should get removed.

S.NO	Description
10	While raining, by gathering wiper control data, windows should get raised automatically.
11	Due to anti-pinch feature, if the system senses any obstacle in the path of the glass the window glass should be stopped moving further up.

Implementation and Summary

Git Link:

Link: Automotive_BMW_Project/40021043_Nayan_Farde_TRN at main · Nayan349/Automotive_BMW_Project (github.com)

Individual Contribution and Highlights

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