GENESIS - Learning Outcome & Mini-project Summary Report

L&T Technology Services

Details

Ver. Rel. No.	Release Date	Prepared By	Reviewed By	To Be Approved	Remarks/Revision Details
1.0	18/02/2022	Vidya Prasad K A 40020555	C Programming on Multiple Platforms		
1.0	18/02/2022	Vidya Prasad K A 40020555	Essentials of Embedded System		
1.0	18/02/2022	Vidya Prasad K A 40020555	Applied SDLC and Software Testing		
1.0	18/02/2022	Vidya Prasad K A 40020555	OOPS with Python		
1.0	18/02/2022	Vidya Prasad K A 40020555	Applied Model Based Design Module		
1.0	18/02/2022	Vidya Prasad K A 40020555	Mastering Microcontrollers with Embedded Driver Development Module		
1.0	18/02/2022	Vidya Prasad K A 40020555	Overview of Automotive Systems		
1.0	18/02/2022	Vidya Prasad K A 40020555	Applied Control Systems and Vehicle Dynamics		
1.0	18/02/2022	Vidya Prasad K A 40020555	Classic Autosar Basic to Intermediate		

Contents

List of Figures

Miniproject – 1: ATM Banking [Individual]

Modules:

1. C Programming
2. Git

Requirements

4W's and 1 H's

Why:

1. To save time from the traditional ways of banking.
2. It is fast and convenient.

Where:

1. This can be used in all the ATMs that are kept in public places.

Who:

1. It can be used by anyone who has a bank account.

When:

1. It can be used 24*7 which is the major advantage.

How:

1. Individuals who has a bank account, and knows how to perform the actions can do this easily.

High Level Requirements

ID	Description	Status
HLR_1	The user can check the total balance	Implemented
HLR_2	The user can deposit money	Implemented
HLR_3	The user can withdraw money	Implemented

Low Level Requirements

ID	Description	Status
LLR_1	Check Balance	Implemented
LLR_2	Deposit money	Implemented
LLR_3	Withdraw money	Implemented

Design

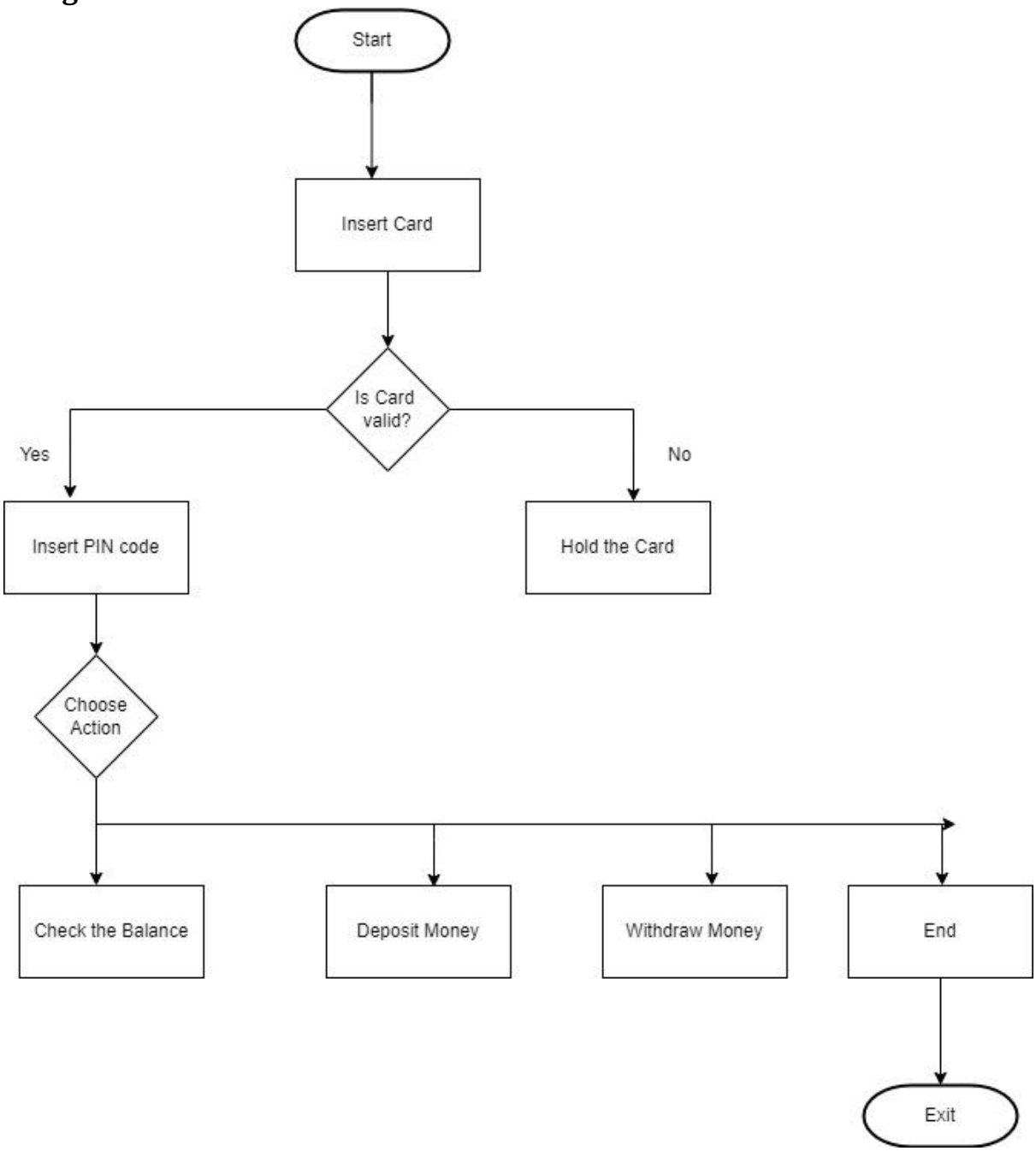

Figure 1 Behavior Diagram

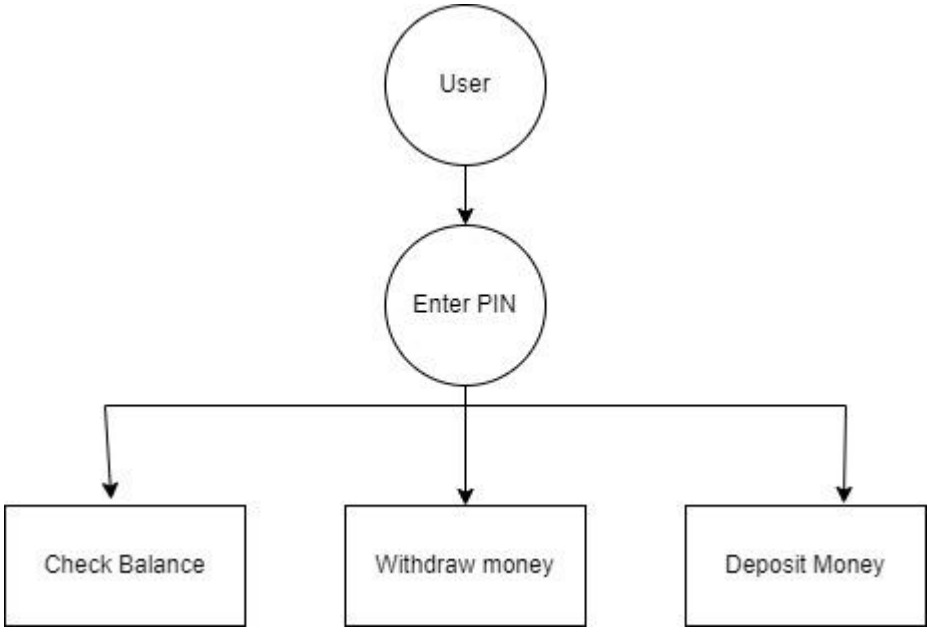

Figure 2 Structure Diagram

Test Plan

High Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLTP_1	check total balance	Choice	Success	Success	Requirement Based
HLTP_2	deposit money	Choice	Success	Success	Requirement Based
HLTP_3	withdraw money	Choice	Success	Success	Requirement Based

Low Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LLTP_1	Total balance	1500	1500	1500	Requirement based

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LLTP_2	Deposit money	1444	16444	16444	Requirement Based
LLTP_3	Withdraw money	1000	15444	15444	Requirement Based

Implementation and Summary

Git Link:

Link: https://github.com/VidyaPrasad008/M1_Application_ATM_Banking.git

Git Dashboard

Build	Code Analysis - Static & Dynamaic	Unity	Code Quality	Git Inspector
C/C++ CI passing Windows C/C++ CI - Build Status passing	Code Quality - Static Code - Cppcheck passing	Unit Testing - Unity passing	Code Quality Score 91 Code Grade A code quality B	Contribution Check - Git Inspector passing

Figure 3 Git Dashboard

Summary

Git Inspector Summary

```
Author                    Commits    Insertions      Deletions    % of changes
VidyaPrasad008                141          5392            767          100.00

Below are the number of rows from each author that have survived and are still
intact in the current revision:

Author                    Rows      Stability          Age       % in comments
VidyaPrasad008            4695           87.1          0.2               11.54
```

Figure 4 Git Inspector Summary

Miniproject 2 – Embedded Calculator [Individual]

Modules

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

Requirements

4W's and 1 H's

Why:

1. It is easy to do and saves a lot of time
2. It gives the exact answer as there won't be any error in calculation

Where:

1. It can be used in places like shops, offices, etc.,

Who:

1. This function can be performed by anyone like students, office employees, retail shop owners etc.,
2. It can also be used for personal purposes.

When:

1. It can be used daily or whenever there is a need for calculation.

How:

1. The user can carry the calculator handy with him by a device or use it through integrations with phone or laptop.

High Level Requirements

ID	Description	Status
HLR_1	Control Unit	Implemented

ID	Description	Status
HLR_2	Input Unit	Implemented
HLR_3	Output Unit	Implemented
HLR_4	Software Design	Implemented

Low Level Requirements

ID	Description	HLR ID	Status
LLR_1	Atmega 328 Microcontroller	HLR_1	Implemented
LLR_2	4*4 Keypad Interface	HLR_2	Implemented
LLR_3	16*2 LCD Interface	HLR_3	Implemented
LLR_4	Simulide	HLR_4	Implemented

Design

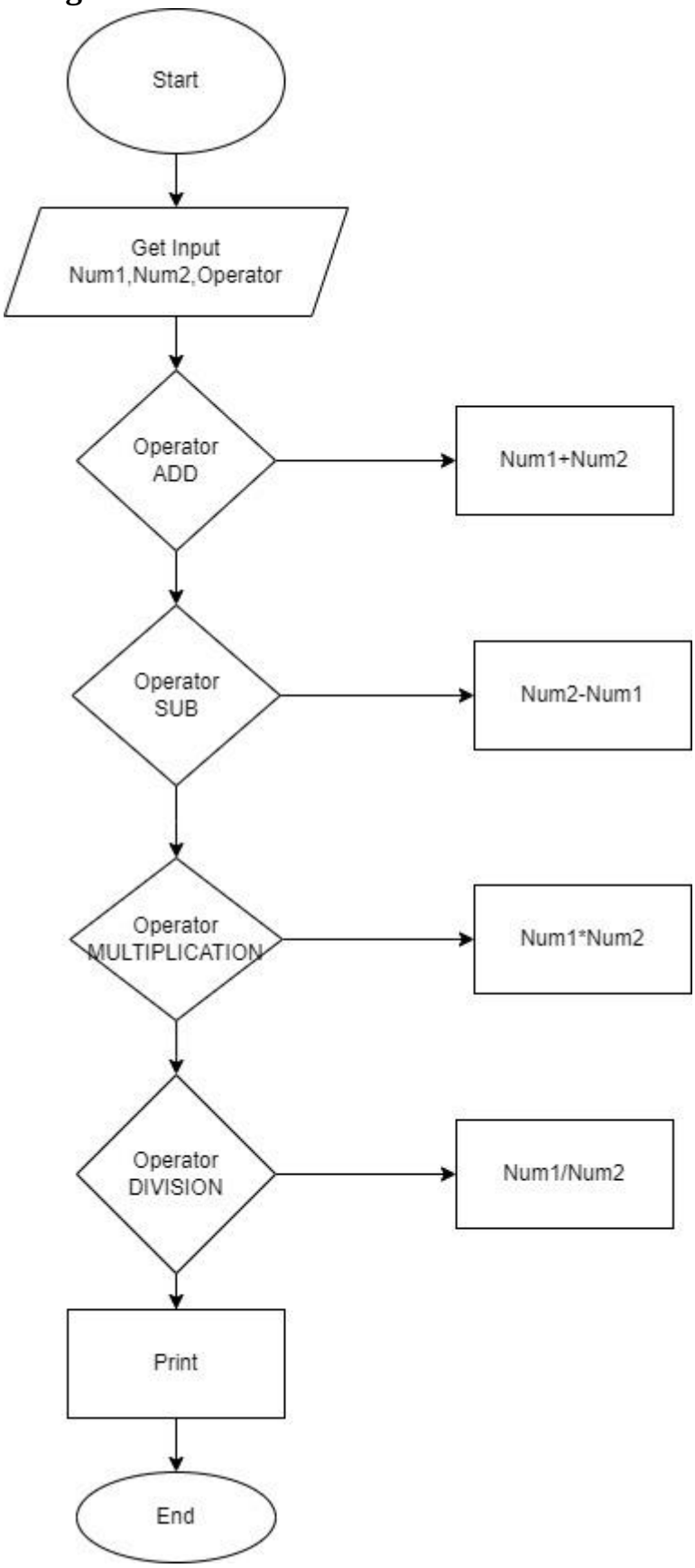

Figure 5 Behavior Diagram

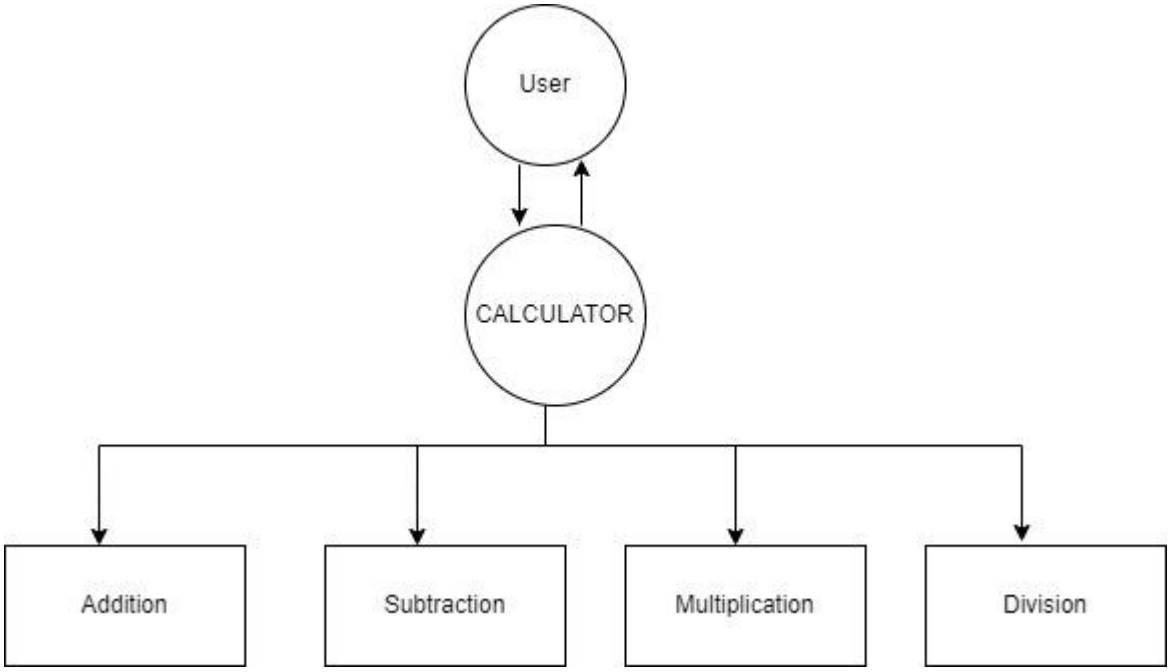

Figure 6 Structure Diagram

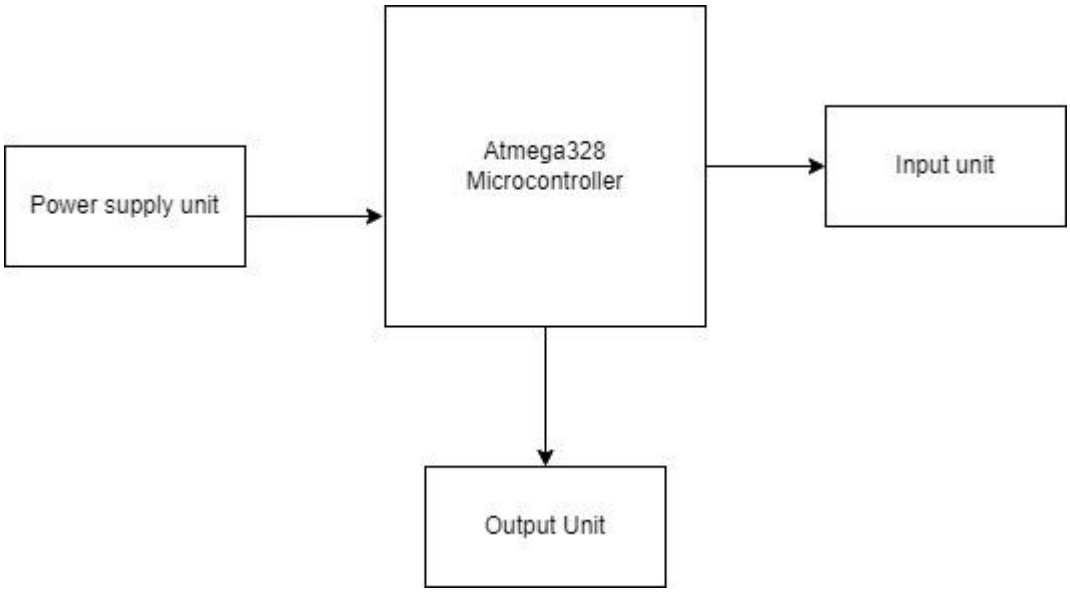

Figure 7Block Diagram

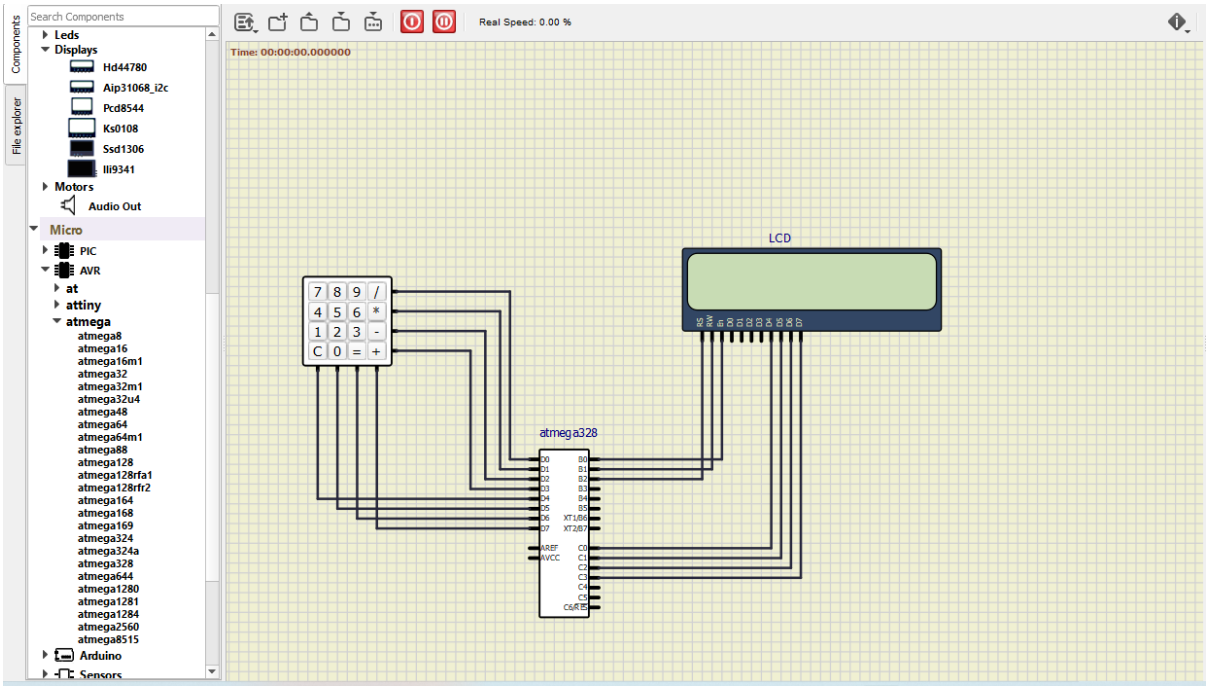

Figure 8 Simulation

Test Plan

High Level Test Plan

Test ID	Description	Exp I/P	Exp O/P	Actual Output	Type of Test
HLTP_1	Power ON	Power	Display ON	SUCCESS	Requirement Based
HLTP_2	User Input	Input Value	Return Output to the User	SUCCESS	Requirement Based
HLTP_3	Return Output from Input	Inputed Value by User	Shows Output in Display	SUCCESS	Requirement Based

Low Level Test Plan

Test ID	Description	Exp I/P	Exp O/P	Actual Output	Type of Test
LLTP_1	Addition	(40, 50)	90	90	Requirement Based
LLTP_2	Subtraction	(70, 20)	50	50	Requirement Based
LLTP_3	Multiplication	(5, 5)	25	25	Requirement Based
LLTP_4	Division	(24, 2)	12	12	Requirement Based

Implementation and Summary

Git Link:

Link: https://github.com/VidyaPrasad008/M1-Embedded_ScientificCalculator.git

Git Dashboard

Build	Cppcheck	Codacy
CI passing	Cppcheck passing	code quality B Code Grade A

Figure 9 Git Dashboard

Miniproject 3 – Patient Management System [Team]

Modules

1. SDLC
2. Git

Requirements

4W's and 1 H's

Who:

- COVID vaccinations are providing to the general public by all government and private Hospitals, small clinics and Dispensaries in the city.

Where:

- Vaccination Management System, a Medical Solution for Clinics, Hospitals & Doctors.
- Allow your patients to book appointments, request video consultations & e-consults.
- Hospitals can now remind patients about vaccination updates with the Vaccination Management System.
- This management system will be very useful in all the hospitals.

When:

- Currently, all people are advised to take the vaccination ever since the Vaccination drive started in February 2021.

Where:

- Vaccination Management System, a Medical Solution for Clinics, Hospitals & Doctors.
- Allow your patients to book appointments, request video consultations & e-consults.
- Hospitals can now remind patients about vaccination updates with the Vaccination Management System.
- This management system will be very useful in all the hospitals.

How:

- On getting due dose of COVID-19 vaccine, the beneficiary will receive SMS on their registered mobile number.
- After all doses of vaccine are administered, a QR code based certificate will also be sent to the registered mobile number of the beneficiary.

High Level Requirements

ID	Description	Category	Status
HR01	End user can read the patient's record	Technical	Implemented
HR02	End user can add new patient's record	Technical	Implemented
HR03	End user can save the patient's records in file	Technical	Implemented
HR04	Even failure occurs, the data will not be lost	Scenario	Implemented
HR05	End user can delete patient's record	Technical	Implemented
HR06	End user can read the data from file	Technical	Implemented
HR07	End user can update the patient's record	Technical	Implemented
HR08	Storage of data occurs while closing the system	Scenario	Implemented

Low Level Requirements

ID	Description	Category	Status
LLR01	Reading patient data should be in 2 ways. First being by searching by id of a patient. By printing all the records available	HLR01	Implemented
LLR02	New record shall be added by providing all the asked information. It should be unique and validated. file or else patient record should not be accepted	HLR02	Implemented
LLR03	User shall be able to save the files, if file already exists then file and should not overwrite it and if file does not exists then it should create a new file	HLR03	Implemented
LLR04	If opening the file fails, then the system should prompt the message "Unable to access file" and should not end the program execution	HLR04	Implemented
LLR05	User need to search by id for the patient record to be deleted, if no such record is available then "No Record Found" Message should be displayed	HLR05,06	Implemented
LLR07	If opening the file fails, then the system should prompt the message "Unable to access file" and should not end the program execution	HLR07	Implemented
LLR08	When user Log off the system perform check and save data to file. If new data in inserted add it to file. If New data is not inserted do not add anything to file	HLR08	Implemented

Design

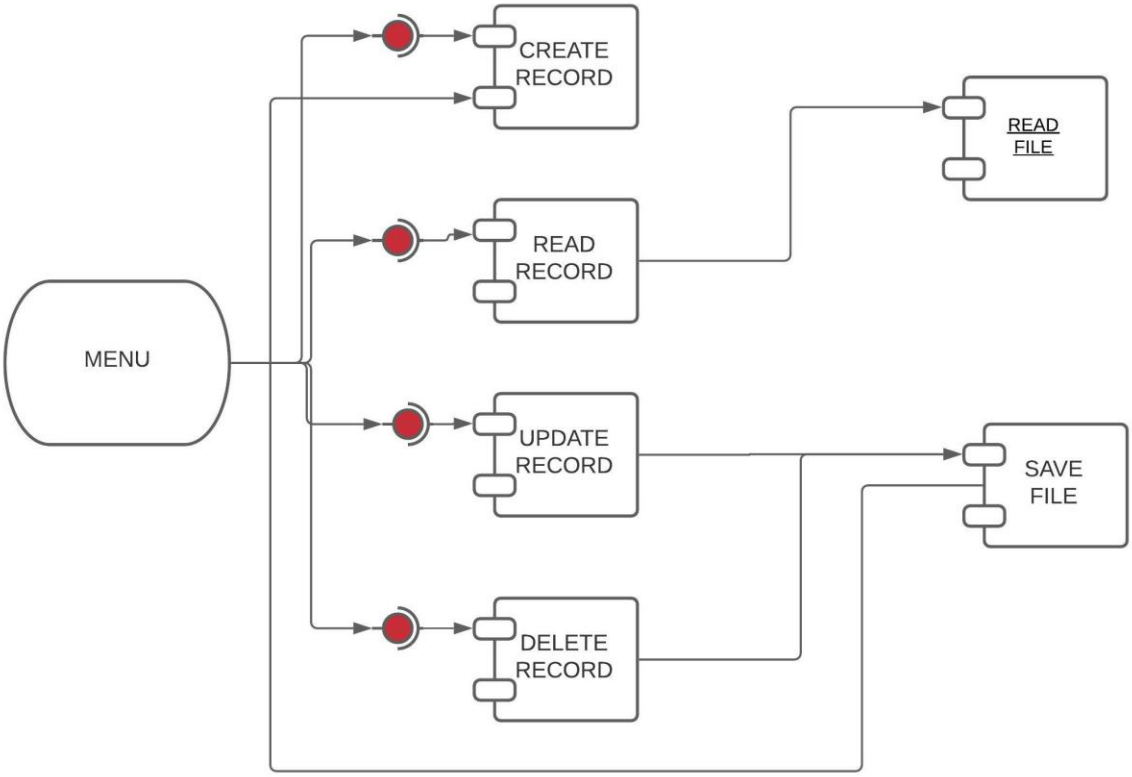

Figure 10Component Diagram

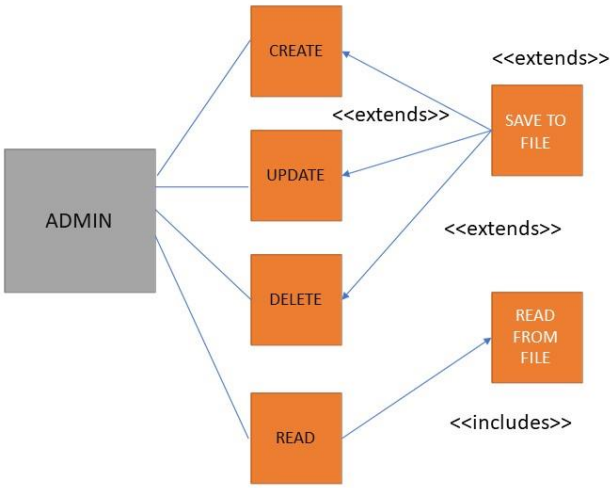

Figure 11Usecase Diagram

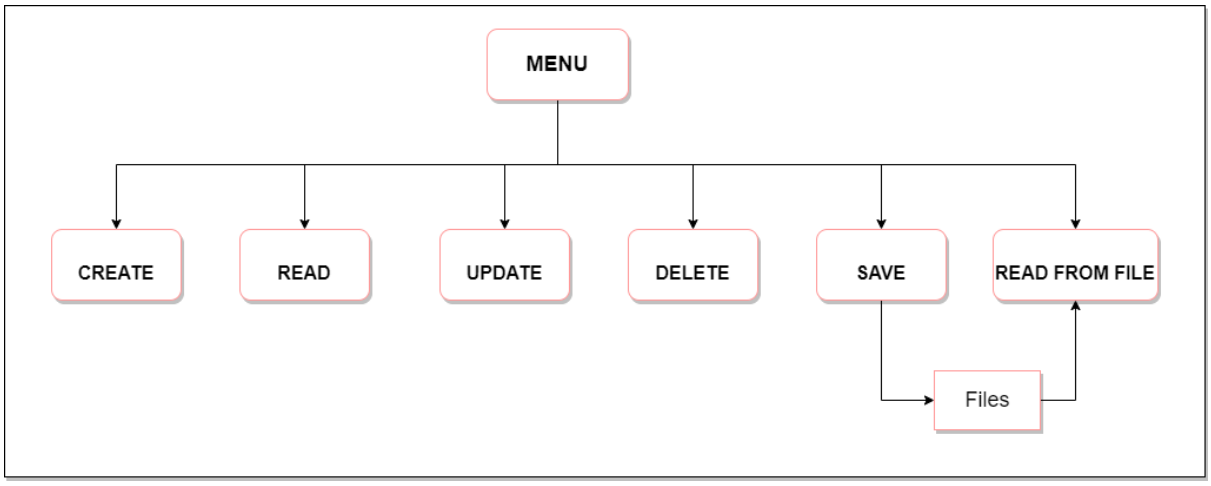

Figure 12High level Diagram

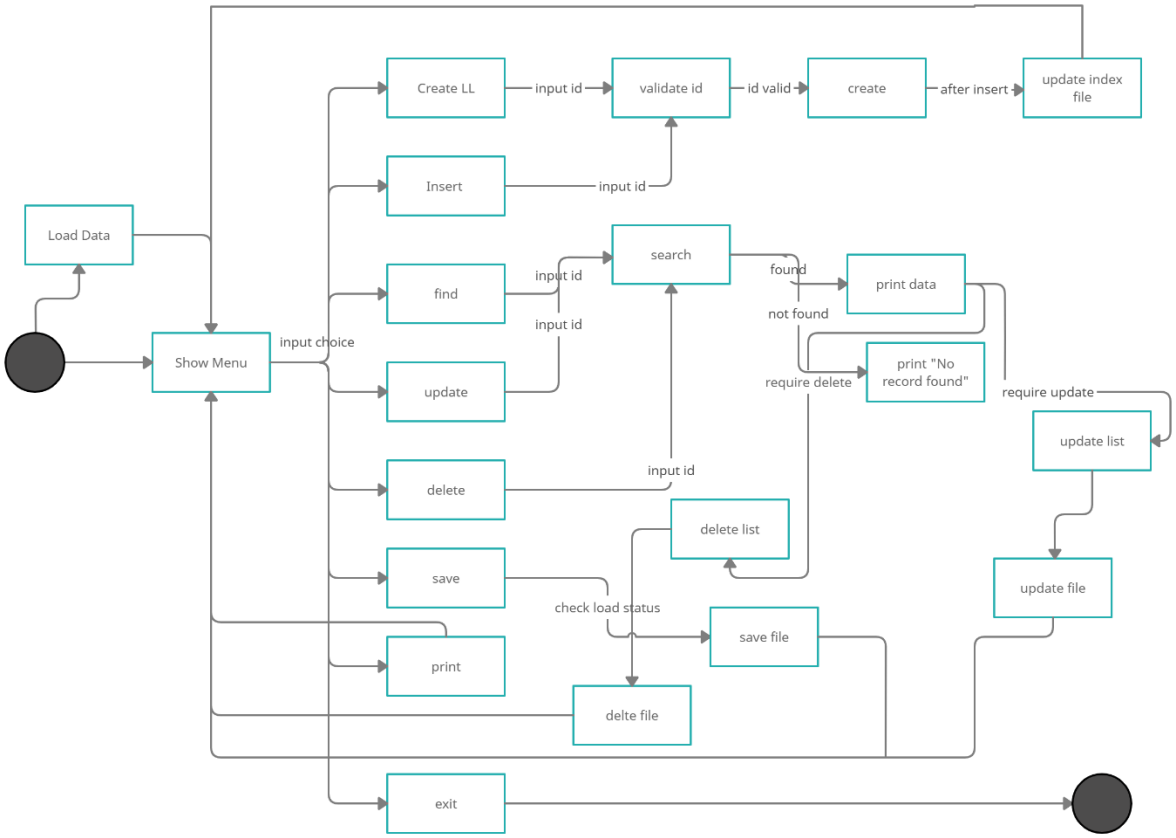

Figure 13State Diagram

Test Plan

High Level Test Plan

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
H_01	Check if Linked List is created or not	(1). NULL Pointer (2). Unique id (3). First name (4). Last name (5). Height (6). Weight (7). Age (8). Insurance Status (9). vaccine code	Pointer to head node	PASS	Requirement based
H_01_01	Check LL initialized from a file	(1). Head Pointer (2). File	LL should be initilized from a file	PASS	Scenario/Technical

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
		Pointer			
H_02	Check Insertion of new data in list	(1). Head Pointer (2). Unique id (3). First name (4). Last name (5). Height (6). Weight (7). Age (8). Insurance Status (9). vaccine code	SUCCESS	SUCCESS	Requirement based
H_02_01	Check if during insertion id gets stored in file	(1). File name (2). file mode (3). File Pointer	SUCCESS	SUCCESS	Requirement based

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
H_02_02	Check if during insertion no head exists	(1). File name (2). file mode (3). File Pointer	NO_HEAD_EXISTS	NO_HEAD_EXISTS	Technical
H_03	Check if records are displayed properly	(1). Head Pointer	SUCCESS	SUCCESS	Requirement based
H_03_01	Check if records in file are displayed properly	(1). File Pointer	SUCCESS	SUCCESS	Technical
H_04	Check if search by Patient id is working correct	(1). Head pointer (2). Id (3). Result Pointer (4). Flag	SUCCESS	SUCCESS	Requirement based
H_05	Check if record is updated properly	(1). Head pointer (2). Id (3).	SUCCESS	SUCCESS	Requirement based

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
		Field to be updated (4). Flag			
H_05_01	Check if record is also updated in File	(1). File Pointer (2). Id	SUCCESS	SUCCESS	Technical
H_06	Deleting Record	(1). Head pointer (2). Id	Pointer to Head node	PASS	Requirement based
H_06_01	If record is only present in List, then delete from List	(1). Head pointer (2). Id	Pointer to head node	PASS	Technical
H_06_02	If record is only present in File, then delete from File	(1). File pointers (2). Id	SUCCESS	SUCCESS	Technical
H_06_03	If record is deleted, then Index	(1). File pointers (2). Id	SUCCESS	SUCCESS	Technical

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
	from Index File should also be deleted				
H_07	When required list can be stored in file	(1). File pointers (2). Head Pointer (3). Flag	SUCCESS	SUCCESS	Requirement based
H_08	When program shuts down records should be saved in File	(1). File pointers (2). Head Pointer	SUCCESS	SUCCESS	Requirement based
H_08_01	When program Shuts down all allocated Memory Locations should be freed	(1). Head Pointer	No Memory Leaks	FAIL	Technical

Low Level Test Plan

Test ID	HLT ID	Description	Exp IN	Exp OUT	Actual Out	Type Of Test
L_01	H_02	During insertion check if ID is unique in INDEX.DAT file	(1). File Pointer (2). ID	SUCCESS	SUCCESS	Requirement based
L_01_02	H_02	Id f during insertion id alrady exists, do not allow insertion	(1). File Pointer (2). ID	ID_ALREADY_EXISTS	ID_ALREADY_EXISTS	Scenario based
L_03	H_02,H_01, H_06,H_07	Check if file is properly opened during program execution	(1). File Name (2). File Mode (3). File Pointer	SUCCESS	SUCCESS	Technical
L_04	H_07,H_08	if data is loaded from	(1). File pointers	SUCCESS	SUCCESS	Technical

Test ID	HLT ID	Description	Exp IN	Exp OUT	Actual Out	Type Of Test
		file during startup then writing of file should begin from the start of file	(2). Head Pointer (3). Flag			
L_05	H_06	If there is only one node in list then deletion from beginning algorithm should work	(1). File pointers	SUCCESS	SUCCESS	Technical
L_06	H_06	If first node is being deleted then deletion from beginning algorit	(1). File pointers	SUCCESS	SUCCESS	Technical

Test ID	HLT ID	Description	Exp IN	Exp OUT	Actual Out	Type Of Test
		hm should work				

Implementation and Summary

Git Link:

Link: https://github.com/GENESIS2021Q1/Applied_SDLC-Dec_Team_3.git

Individual Contribution and Highlights

Summary

- Better Vaccine Management
- Ease burden on staff
- Timely Patient Care
- Organization

Role in Project Team

1. Worked on implementing main code file and design part

Miniproject 4 – Attendance Automation[Team]

Modules

1. Python
2. Git

Requirements

4W's and 1H:

Who

- User may easily get attendance history of a particular student

What

- An automatic attendance system is an educational ERP system that records the student's attendance in an institution.
- Unlike the conventional attendance system, the automatic attendance software enables the faculty to record, store, and monitor students' attendance history & manage the classroom efficiently.

When

- Attendance management systems have been in existence for as long as one can remember.
- Although the tools and processes we use to monitor them have changed, the crux of the concept remains the same.
- In simple terms, attendance management is the process of keeping track of all the employees in an organisation, monitoring their work times, having all leave requests and calendars collated in one place for easy management, and tracking daily expenses.

Where

- An online attendance application also makes it easier for the HR team to manage employee's leave.
- All you need is sign into the system to see how much leave you have left.
- Attendance software also facilitates the management of all matters related to attendance.

How

- The system helps the faculty to easily find out defaulters in a single click.

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

ID	Feature	High Level ID	Status
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	SUCCESS	SUCCESS	

ID	HLTP ID	Description	Expected I/P	Actual O/P	Type Of Test
		Attendance Status			
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/sunilkora31/oops_with_python_Attendance-Automation_team-15.git

Git Dashboard

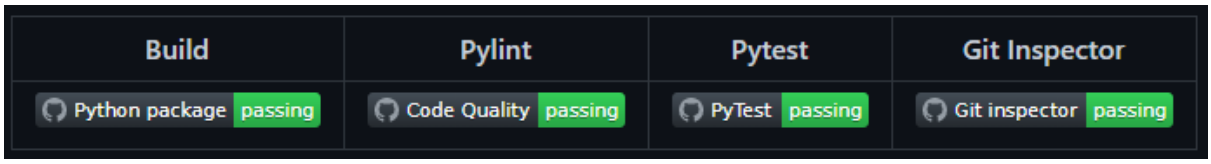

Figure 14 Git Dashboard

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
3. Tester: Writing Testcases and testing the integrated code

Miniproject 5 – Kia Project[Team]

Modules

1. Matlab
2. Git

Requirements

We have implemented following features

1. Adaptive Cruise Control System
2. Anti Lock Braking System
3. Automatic Transmission Control System
4. Door Locking system
5. Engine Braking System
6. Lane Assist System
7. Power Window

Door Locking System:

Power door locks (also known as electric door locks or central locking) allow the driver or front passenger to simultaneously lock or unlock all the doors of an automobile or truck, by pressing a button or flipping a switch.

Design

This project was implemented using Matlab.

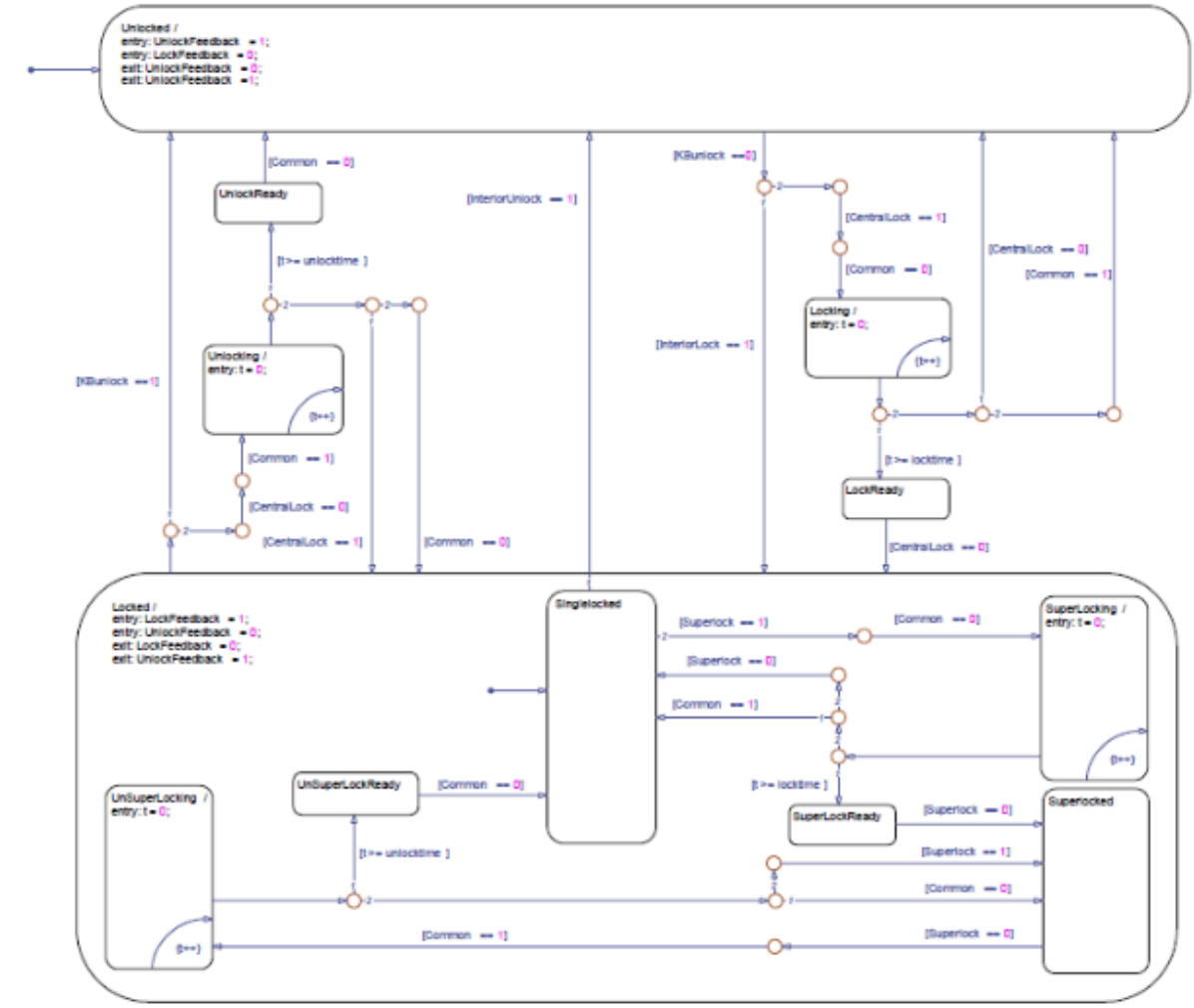

Figure 15Simulation

Implementation and Summary

Git Link:

Link: https://github.com/karthikeyans99/Genesis2021_Applied_Mbd-Kia_Project_Team.git

Individual Contribution and Highlights

1. Door Locking System Case Study
2. Designed and implemented using matlab and the file is in github.

Role in Project Team

1. Designer: Done Designing for Door Locking system using matlab.

Miniproject 6 – Wiper Control[Team]

Modules

1. C Programming
2. STM32

Requirements

4W's and 1'H

Who:

A wiper speed control system for an automotive wiper controls the operational speed of a wiper in accordance with rain conditions.

What:

Vehicles are now available with driver-programmable intelligent windscreen wipers that detect the presence and amount of rain using a rain sensor.

When:

Whenever the water hit a dedicated sensor that located on windscreen, it will send a signal to move on the wiper motor. Once water is not detected by sensor, the wiper will automatically stop. This will help the driver to give more concentration and reduce the car accident probability.

Where:

It is located underneath the dashboard, above the brake and accelerator pedal, and is responsible for the complete operation of the windshield wiper system.

How:

Windshield wipers are controlled by the stalk on the right side of your steering wheel. Simply moving the stalk down will turn your windshield wipers on. Moving the stalk down will turn your wipers on.

High Level Requirements

ID	Description	Status
HLR_1	Ignition at ACC - Red LED ON	Pass
HLR_2	Wiper ON - LED'S ON in Blue, Green and Orange	Pass

ID	Description	Status
HLR_3	Wiper OFF - LED'S OFF in Blue, Green and Orange	Pass
HLR_4	Ignition at lock - Red LED OFF	Pass

Low Level Requirements

ID	Description	Status
LLR_1	Pressing button for two seconds - Red LED ON	Pass
LLR_2	Wiping at 1Hz - Blue LED ON	Pass
LLR_3	Wiping at 4Hz - Green LED ON	Pass
LLR_4	Wiping at 8Hz - Orange LED ON	Pass
LLR_5	Pressing button for two seconds - Red LED OFF	Pass

Design

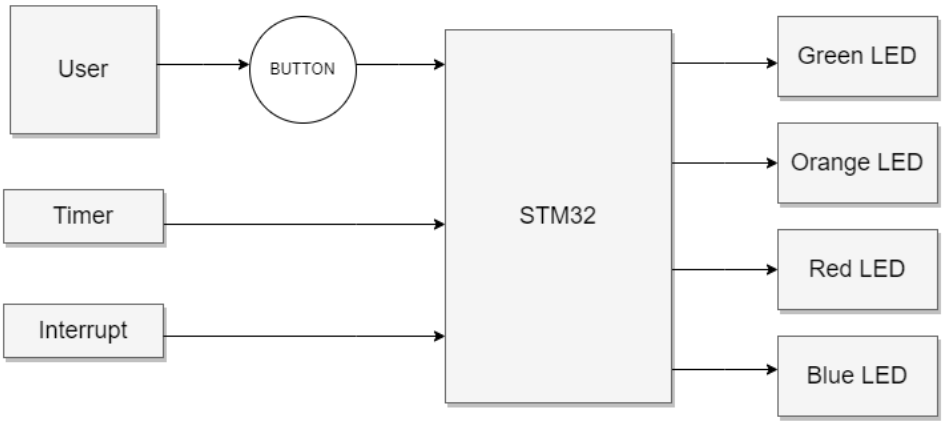

Figure 16Structure Diagram

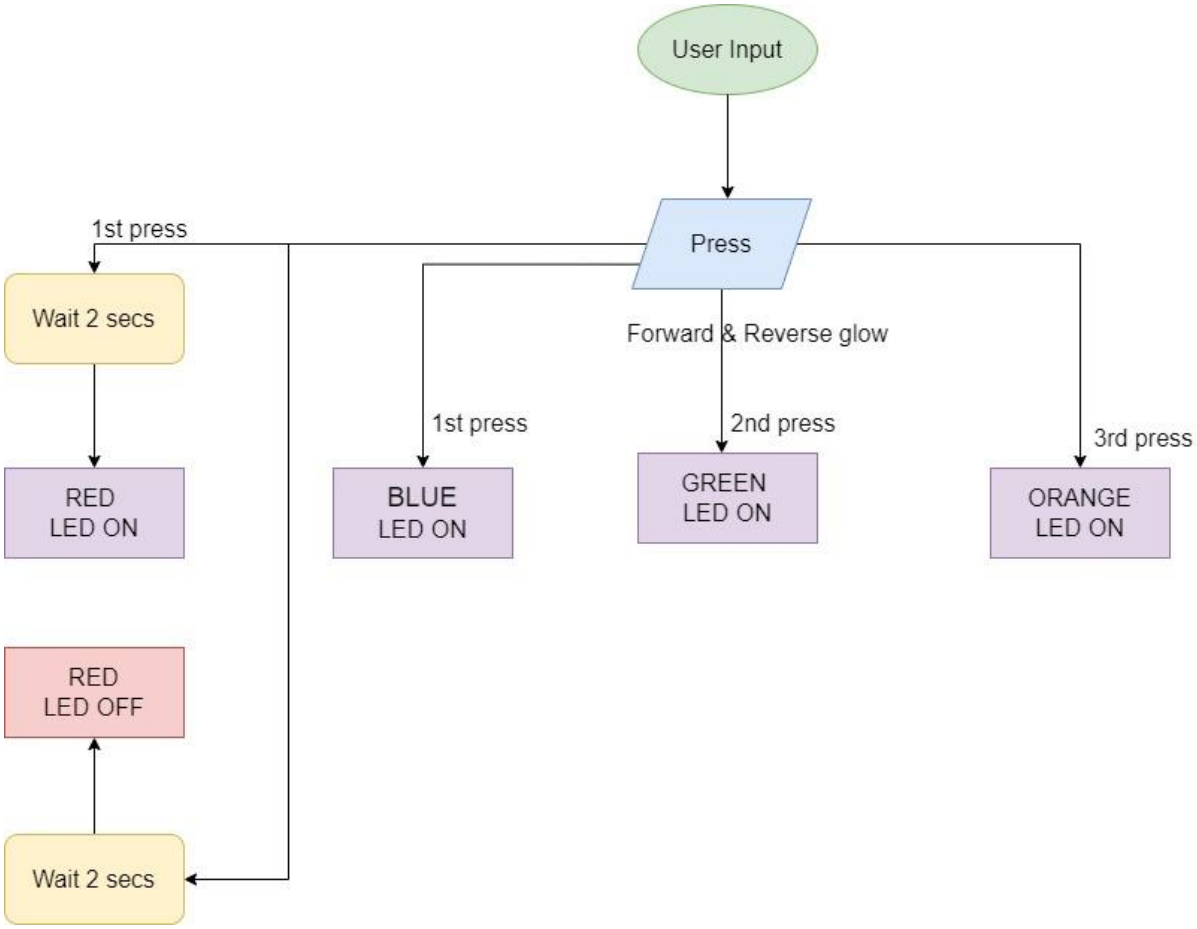

Figure 17 Behavior Diagram

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-Team65.git

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 – Jeep Compass Project[Team]

Modules

1. Automotive Systems
2. Git

Requirements

Door System is a type of door opening, typically hinged on its front edge, but sometimes attached by other mechanisms such as tracks, for entering and exiting a vehicle. Doors most often integrate side windows for visibility from inside the car and can be locked to secure the vehicle. The door system available in this car are,

1. Power Door Lock
2. Passive Keyless Entry
3. Automatic Unlock Doors on Exit
4. Power Window System

High Level Requirements:

S. No.	Feature	Description
HLR_1	Power Door Lock	Driver or front passenger can lock or unlock the doors.
HLR_2	Passive Keyless Entry	Allows you to unlock and lock the doors to a vehicle without using a key.
HLR_3	Automatic Unlock Door on Exit	The door locks will unlocked automatically.
HLR_4	Power Window System	All windows can accessed by switching on the passenger door trim panel.

Low Level Requirements:

S. No.	Feature	Description
LLR_1	Power Door Lock	It automatically locks doors at certain speeds.

S. No.	Feature	Description
LLR_1.1	Power Door Lock	To secure all the doors at the same time.
LLR_2	Passive Keyless Entry	Doors will be lock or unlock by using a radio frequency remote keyless system.
LLR_2.1	Passive Keyless Entry	If the vehicle is unlocked by Passive Entry and no door is opened within 60 seconds, the vehicle will re-lock.
LLR_3	Automatic Unlock Door on Exit	If the vehicle is unlocked by Passive Entry and no door is opened within 60 seconds, the vehicle will re-lock.
LLR_4	Power Window System	The windows will be open and close within 4 seconds.

Design

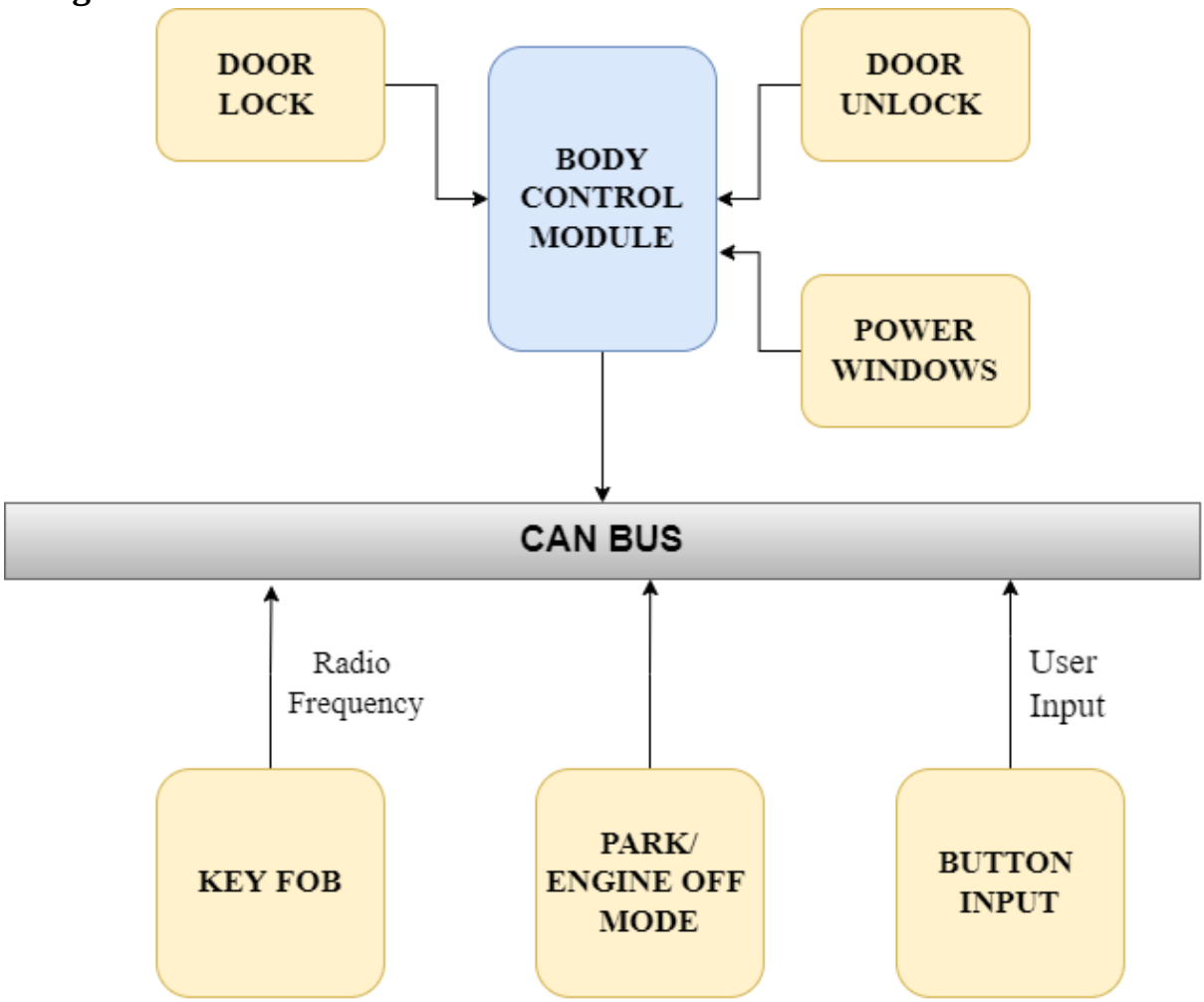

Figure 18Structure Diagram

Implementation and Summary

Git Link:

Link: https://github.com/karthikeyans99/Jeep_Compass_Team.git

Individual Contribution and Highlights

3. Door System Case Study
4. Source code management using GitHub

Role in Project Team

2. Designer: Done Designing for Project
3. Researcher: Done case study for Door Locking System

Miniproject 8 – EV Car [Team]

Modules

1. Matlab
2. Matlab Script

Requirements

Motor Specifications:

Component	Tata Nexon	Hyundai Kona
Top speed	120kmh	165kmh
Acceleration(0-100kmph)	9.14s	7.9s
Engine type	PMSM	PMSM
Max motor performance	127bhp 245Nm	134 bhp 395 Nm
Driving Range	312 kms	450kms
Battery	30.2 kWh, Lithium-Ion Polymer, 320V Battery Placed Under Floor Pan	39.2 kWh, Lithium-Ion Polymer, 327V Battery Placed Under Floor Pan

Safety Specifications:

Component	Tata Nexon	Hyundai Kona
Overspeed Warning	1 beep over 80kmph, Continuous beeps over 120kmph	1 beep over 80kmph, Continuous beeps over 120kmph
Emergency Brake Light Flashing	No	Yes
NCAP Rating	5 Star (Global NCAP)	5 Star (Euro NCAP)
Airbags	2 Airbags (Driver, Passenger)	6 Airbags (Driver, Passenger, 2 Curtain, Driver Side, Front Passenger Side)

Component	Tata Nexon	Hyundai Kona
Middle Rear Head Rest	No	Yes

Battery Performance:

1. Both cars use the same Lithium-ion battery type as it is the industry standard right now.
2. Hyundai Kona has a massive 452 km lead in terms of range which is more than double of what the Tata Nexon.
3. Battery charging times are longer in the Hyundai Kona due to its larger 39.1 kWh battery compared to the 37.4 kWh.
4. Hyundai Kona offer fast charging.

Braking Performance:

1. Hyundai Kona has ESP, not in Tata Nexon.
2. Huyndai Kona has TCS, not in Tata Nexon.

Suspension Performance:

1. Hyundai kona has Independent MacPherson strut with coil spring and Tata nexon has McPherson Strut Type front suspension.
2. Hyundai kona has Twist beam with dual path Strut and Tata nexon has multi-Link rear suspension.

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 Car
2. Researcher: Done case study for EV

Miniproject 9 – Power Door Locking System[Individual]

Modules

1. Autosar
2. Git

Requirements

A power door lock switch is located on each of the front door trim panels. Use this switch to lock or unlock the doors, liftgate and fuel door. If you push the power door lock switch while the ignition is in the ON/RUN position, and any door or the liftgate is open, the power locks will not operate. This prevents you from accidentally locking the key fob in the vehicle. Placing the ignition in the OFF position or closing the doors and liftgate will allow the locks to operate. If the driver door is open, and the ignition is in the RUN position, a chime will sound as a reminder to remove the key.

High Level Requirements:

HLR_1	Power Door Lock	Driver or front passenger can lock or unlock the doors.

Low Level Requirements:

LLR_1	Power Door Lock	It automatically lock doors at certain speeds.
LLR_1.1	Power Door Lock	To secure all the doors at the same time.

Design

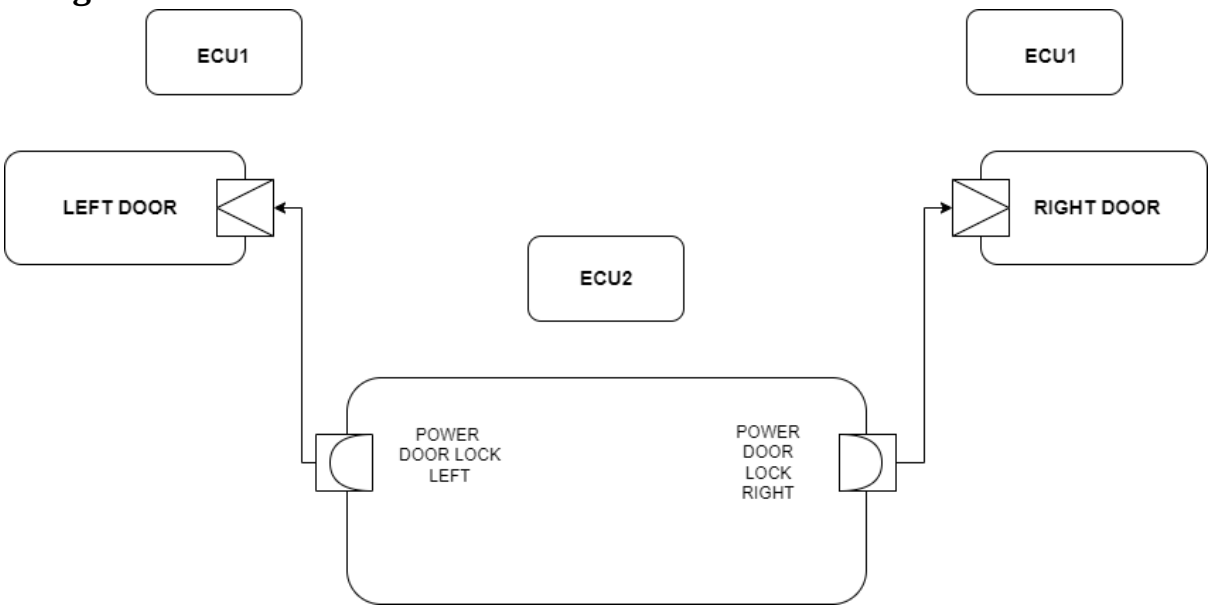

Figure 19VFB Diagram

Implementation and Summary

Git Link:

Link: https://github.com/VidyaPrasad008/PowerDoorLockSystem_40020555_DPS.git

Individual Contribution and Highlights

1. Power door lock System Case Study
2. Source code management using GitHub
3. AtomicSwComponent
4. SWCInternalBehavior
5. SWCImplementation