



**slington college**  
(इरिलिङ्टन कलेज)

**Module Code & Module Title**

**CS6P05NI Final Year Project**

**Assessment Weightage & Type**

**40% Final Project Report**

**Semester**

**2023 Spring**

**PROJECT TITLE: Smart Garage**

**Student Name: Aadarsha Muni Shakya**

**London Met ID: 20049438**

**College ID: NP01NT4S210023**

**Internal Supervisor: Monil Adhikari**

**External Supervisor: Suryansh Mathema**

**Assignment Due Date: April 19, 2023**

**Assignment Submission Date: April 19, 2023**

**Word Count (Where Required): 8233**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## **Acknowledgement**

With the assistance of ongoing direction and encouragement throughout the project's conception, development, and conclusion. This IoT system would not have operated successfully, in my opinion, if the supervisors had not provided vital feedback.

The information and contemplation that I was able to get throughout this time period assisted me in adhering to a suitable project development approach and implementing project solutions through an iterative process. I am also grateful to Islington College for giving the resources and assistance needed to complete this project.

**Abstract**

This report illustrates the development of Smart Garage while submitting in the Interim report. In development front end application development is done using Flutter, Android studio, VS code and dart programming language. This project is developed to make life easier by opening the shutter without human interactions. After verification and authentication, the garage shutter will open automatically. Also, admin console can open and close the shutter without the need of authentication and verification.

## Table of Contents

Acknowledgement.....	2
Abstract.....	3
Table of Contents.....	4
Table of Figures .....	9
Table of Tables .....	13
Chapter 1: Introduction.....	14
1.1. Project Description .....	14
1.2. Current Scenario.....	15
1.3. Problem Domain and Project as a Solution .....	16
1.4. Aim and Objectives.....	18
1.4.1. Aim.....	18
1.4.2. Objectives .....	18
1.5. Structure of the Report .....	19
1.5.1. Background .....	19
1.5.2. Development .....	19
1.5.3. Testing and Analysis .....	19
1.5.4. Conclusion.....	19
Chapter 2: Background .....	20
2.1. About the End Users.....	20
2.1.1. End-Users .....	20
2.1.2. Admin .....	20
2.2. Understanding the Solution .....	20
2.2.1. Solution for Users.....	21

2.2.2. Solution for Admin .....	21
2.3. Similar Projects .....	22
2.3.1. Project 1: .....	22
2.3.2 Project 2: .....	23
2.3.3 Project 3: .....	24
2.4. Comparisons .....	25
Chapter 3: Development .....	26
3.1. Considered Methodologies .....	26
3.1.1 Waterfall Methodology .....	26
3.2. Selected Methodology .....	28
3.3. Phases of Methodology .....	28
3.4. Survey Results .....	30
3.4.1. Pre-Survey Results .....	31
3.4.2. Post-Survey Results .....	31
3.5. Requirement Analysis .....	32
3.5.1. Feature Requitelements .....	32
3.5.2. Software Requirements .....	32
3.5.3. Hardware Requirements .....	33
3.6. Design .....	34
3.6.1. Application Design .....	34
3.6.2. System Design .....	36
3.6.3. Feature Design .....	39
3.7. Implementation .....	54
3.7.1. IoT System .....	54

3.7.2. Mobile Application .....	58
Chapter 4: Testing and Analysis .....	67
4.1. Test Plan .....	67
4.1.1. Unit Testing, Test Plan .....	67
4.1.2 System Testing, Test Plan .....	67
4.2. Unit Testing.....	68
4.2.1. Test case1: Authentication .....	68
4.2.2. Test case 2: Verification .....	71
4.2.3. Test case 3: Clicking Picture of The Numberplate .....	72
4.2.4. Test case 4: OCR in a Loop .....	73
4.2.5. Test case 5: Admin Password Validation .....	77
4.2.6. Test case 6: Admin Console Login.....	79
4.2.7. Test case 7: Open Button.....	81
4.2.8. Test case 8: Navigation Button .....	84
4.2.9. Test case 9: Get Data Button .....	86
4.2.10. Test case 10: Clear Button.....	89
4.2.11. Test case 11: Logout Button .....	91
4.3. System Testing .....	93
4.3.1. Test case 12: Verification and Authentication Shutter Opening .....	93
4.3.2. Test case 13: Mobile App.....	95
4.4. Critical Analysis .....	100
4.4.1. Analysing Failed Test Cases.....	101
4.4.2. Test Summary.....	104
Chapter 5: Conclusion.....	105

5.1 Legal, Social and Ethical Issues .....	106
5.1.1. Legal Issues .....	106
5.1.2. Social Issues .....	107
5.1.3. Ethical Issues .....	108
5.2. Advantages.....	109
5.2.1. Increased Security .....	109
5.2.2. Remote Access .....	109
5.2.3. Automated operation .....	109
5.2.4. Activity Monitoring .....	109
5.2.5. Improved convenience .....	109
5.3. Limitations .....	110
5.3.1. Cost.....	110
5.3.2. Reliability.....	110
5.3.3. Complexity .....	110
5.3.4. Privacy concerns .....	110
5.3.5. Compatibility.....	110
5.4. Future Work.....	111
5.4.1. Two factor authentication .....	111
5.4.2. Cloud Storage .....	111
5.4.3. Door locking system.....	112
5.4.4. Notification System .....	112
5.4.5. Power Management .....	113
Chapter 6: References .....	114
Chapter 7: Appendix.....	117

7.1. Appendix 1: Pre-Survey .....	117
7.1.1. Pre-Survey form .....	117
7.1.2. Pre-Survey Result .....	120
7.2. Appendix 2: Post-Survey .....	124
7.2.1. Post-Survey form .....	124
7.2.2. Post-Survey Result .....	127
7.3. Appendix 3: Sample Code .....	132
7.3.1. Backend .....	132
7.3.2. Frontend .....	135
7.4. Appendix 4: Designs .....	144
7.4.1. Gantt chart .....	144
7.4.2. Work Breakdown Structure .....	144
7.4.3. Hardware Architecture .....	145
7.4.4. Block Diagram .....	146



## Table of Figures

Figure 1: Graph of IoT's Current Scenario (Hasan, 2022) .....	15
Figure 2: Waterfall Methodology (Adobe Communcations Team, 2022) .....	26
Figure 3: Login Page Design .....	34
Figure 4: Home Page Design .....	35
Figure 5: Log Info Page Design.....	35
Figure 6: System Diagram.....	36
Figure 7: Activity Diagram for Admin Console .....	37
Figure 8: Activity Diagram for IoT System .....	38
Figure 9: Data Flow Diagram for Verification.....	39
Figure 10: Communication Diagram for Verification .....	39
Figure 11: Sequence Diagram for Verification.....	40
Figure 12: Data Flow Diagram for Authentication.....	41
Figure 13: Communication Diagram for Authentication .....	41
Figure 14: Sequence Diagram for Authentication.....	42
Figure 15: Data Flow Diagram for Open Button .....	43
Figure 16: Communication Diagram for Open Button .....	43
Figure 17: Sequence Diagram for Open Button .....	44
Figure 18: Data Flow Diagram for Get data Button .....	45
Figure 19: Communication Diagram for Get data Button.....	45
Figure 20: Sequence Diagram for Get data Button .....	46
Figure 21: Data Flow Diagram for Admin Login .....	47
Figure 22: Communication Diagram for Admin Login.....	47
Figure 23: Sequence Diagram for Admin Login .....	48

Figure 24: Data Flow Diagram for Log/Home page Button.....	49
Figure 25: Communication Diagram for Log/Home page Button.....	49
Figure 26: Sequence Diagram for Log/Home page Button.....	50
Figure 27: Data Flow Diagram for Clear Button .....	51
Figure 28: Communication Diagram for Clear Button.....	51
Figure 29: Sequence Diagram for Clear Button .....	52
Figure 30: Data Flow Diagram for Logout Button .....	53
Figure 31: Communication Diagram for Logout Button .....	53
Figure 32: Sequence Diagram for Logout Button .....	53
Figure 33: Opening json file and extracting values.....	54
Figure 34: JSON File.....	54
Figure 35: Command for fswebcam to click Picture .....	54
Figure 36: Use of Tesseract OCR .....	55
Figure 37: Authentication and verification backend code .....	56
Figure 38: Writing to json file.....	56
Figure 39: Rest API.....	57
Figure 40: Main Page .....	58
Figure 41: Login Page 1 .....	59
Figure 42: Login Page 2 .....	60
Figure 43: Login Page 3.....	61
Figure 44: Home Page 1 .....	62
Figure 45: Home Page 2 .....	63
Figure 46: Home Page 3 .....	63
Figure 47: Log Info Page 1 .....	64

Figure 48: Log Info Page 2.....	65
Figure 49: Log Info Page 3.....	66
Figure 50: Authentication Successful .....	69
Figure 51: Authentication backend code .....	69
Figure 52: OCR backend code .....	70
Figure 53: Verification Successful .....	71
Figure 54: Verification backend code .....	71
Figure 55: Picture clicked by USB cam .....	72
Figure 56: Command for OCR in loop .....	73
Figure 57: Valid images for OCR .....	73
Figure 58: OCR value of 1.jpg.....	74
Figure 59: OCR value of 2.jpg.....	74
Figure 60: OCR value of 3.jpg.....	75
Figure 61: OCR value of 4.jpg.....	75
Figure 62: OCR value of 5.jpg.....	76
Figure 63: Admin Login Page.....	77
Figure 64: Entering incorrect password.....	78
Figure 65: Password validation .....	78
Figure 66: Admin Login page .....	79
Figure 67: Entering Correct password.....	80
Figure 68: Home page is displayed.....	80
Figure 69: Open Button in Home page.....	81
Figure 70: Frontend post request successful .....	82
Figure 71: Backend Post request successful .....	83

Figure 72: Navigation button in App bad of the Home Page .....	84
Figure 73: Log Info Page displayed.....	85
Figure 74: Get button In Log Info Page .....	86
Figure 75: Data from Json file stored in Backend in displayed .....	87
Figure 76: Json file in the backend.....	87
Figure 77: Get Request successful .....	88
Figure 78: Data displayed in Log info page .....	89
Figure 79: Clear button clearing data .....	90
Figure 80: Logout Button in the app bar .....	91
Figure 81: Logged out of the admin console .....	92
Figure 82: Running the backend code.....	93
Figure 83: Dictionary file updated.....	94
Figure 84: Entering password in the admin login .....	96
Figure 85: Home displayed .....	96
Figure 86: Post request to open Shutter.....	97
Figure 87: Navigating to Log info page.....	97
Figure 88: Get request to display contents of backend .....	98
Figure 89: Clear button.....	98
Figure 90: Logged out .....	99
Figure 91: Code for OCR .....	101
Figure 92: Inaccurate Numberplate picture clicking .....	102
Figure 93: Output according to the Image .....	102
Figure 94: Authentication not being successful .....	103

**Table of Tables**

Table 1: Comparison Table .....	25
Table 2: Test 1 Authentication.....	68
Table 3: Test 2 Verification.....	71
Table 4: Test 3 Clicking Pictures .....	72
Table 5: Test 4 OCR in Loop.....	73
Table 6: Test 5 Admin password Validation .....	77
Table 7: Test 6 Admin Console Login .....	79
Table 8: Test 7 Open Button .....	81
Table 9: Test 8 Navigation Button .....	84
Table 10: Test 9 Get Data Button.....	86
Table 11: Test 10 Clear Button .....	89
Table 12: Test 11 Logout Button .....	91
Table 13: Test 12 Verification and Authentication Shutter Opening .....	93
Table 14: Test 13 Mobile App .....	95

## **Chapter 1: Introduction**

### **1.1. Project Description**

A smart garage is an interesting piece of technology that can make your life easier. It essentially allows you to open and close your garage doors without difficulty. How does it accomplish this? It has these small sensors positioned throughout your garage that detect when your automobile is nearby. When you drive in, the garage door automatically opens, and when you drive out, it closes automatically. Don't worry, the sensors are really intelligent; they can even determine if it's your car or not, so you won't have to worry about unwanted access.

Once the vehicle has been authorized, the smart garage will automatically open the door using motors or other mechanical systems. The door will then close automatically once the vehicle has passed through.

Smart garage door openers are just one part of a bigger category of smart home automation devices. They can be connected to other smart home systems, allowing homeowners to remotely control and monitor their garage doors using a smartphone app or other internet-connected device.

## 1.2. Current Scenario

Now a days Internet of Thing (IoT) devices has taken over most platforms. From complex devices like cars to simple device like a light bulb can be found using the concept of IoT. According to a survey conducted about this project, mostly everyone was familiar with IoT domain, and they use IoT gadgets. For survey click [here](#).

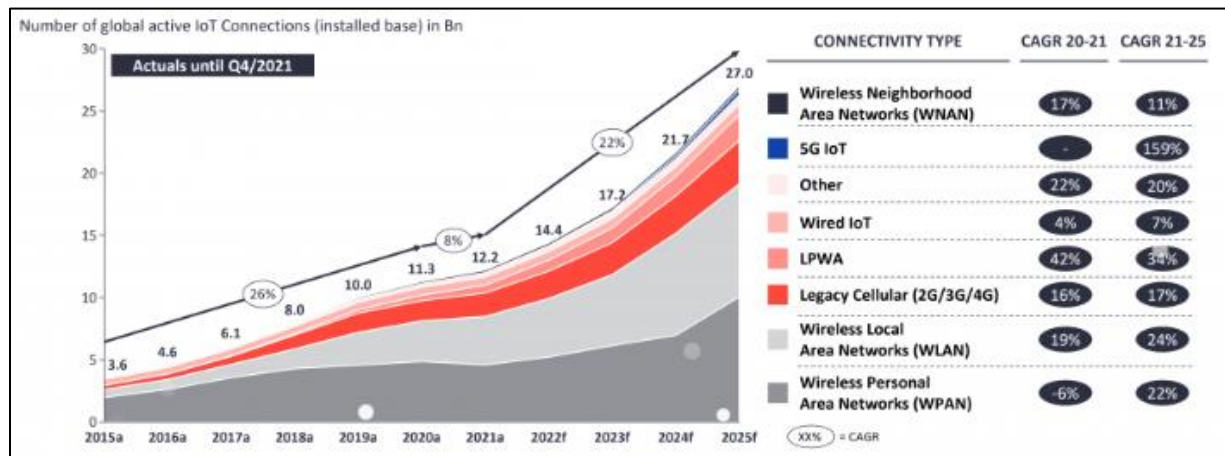


Figure 1: Graph of IoT's Current Scenario (Hasan, 2022)

According to a survey done by IOT ANALYTICS the use of IoT devices is growing by 18% in 2022 and is expected to grow more in the future. (Hasan, 2022)

Since the use of IoT devices is increasing this project is developed. Not many concepts of smart garage are implemented which verifies and authenticates vehicles before opening the shutter.

### 1.3. Problem Domain and Project as a Solution

The concept of garage doors started with a basic mechanism using a pulley and a lever or switch to trigger a motor for opening the door. Old garages that use manual mechanisms to open and close the door can be inconvenient and time-consuming to use. In addition, these old mechanisms may be prone to breakdowns and require regular maintenance. They can also be less secure than modern smart garage systems, as they can be more easily accessed by unauthorized individuals. Finally, these old garage systems may lack the convenience and flexibility of modern smart garage systems, which allow the owner to remotely control the door and monitor access to their garage using their smartphone or other internet-connected device.

Smarter garage systems have been developed as a result of developments in sensor technology. These current systems are outfitted with sensors that can detect the presence of a car and will automatically open or close the garage door without the owner's intervention. This cutting-edge technology not only saves time but also provides users with more convenience. There will be no more manually opening and closing the garage door.

While automated garage systems have numerous advantages, security remains a major worry. Unauthorized individuals may acquire access to the garage if the authentication and verification processes are not adequately developed or applied. Smart garage systems must use advanced authentication techniques like as numberplate recognition authentication and verification to ensure that only authorized users are permitted access. This will help to keep the garage safe and provide the owner with peace of mind that only authorized personnel will have access to their stuff.

Furthermore, smart garage systems must have strong security mechanisms in place to protect against hacking and other types of cyber-attacks. These safeguards are critical in maintaining the garage's security and preventing the owner's personal information from being compromised. Smart garage systems can create a secure and safe environment for the owner and their goods by adopting powerful security features.



In general, while smart garage systems have various advantages over traditional garage systems, adequate security measures must be implemented to ensure that they provide a secure and secured environment for the owner's goods and vehicle.

Smart garages provide an alternative to manually opening and closing garage doors. The owner can use this technology to open the door without getting out of the automobile, saving time and effort. To guarantee that only authorized individuals have access to the garage, a load cell sensor is used to detect the presence of a car, and an ultrasonic sensor and USB camera combo is used to authenticate the individual. The door opens automatically after the car and individual have been authenticated.

## 1.4. Aim and Objectives

### 1.4.1. Aim

Allowing vehicles inside the garage after **verifying and authenticating** them using IoT devices and programs.

### 1.4.2. Objectives

- To verify vehicles using a load cell sensor when weight of the vehicle outside the garage is detect.
- To authenticate vehicles using camera to detect numberplate and allow vehicles with permission.
- Properly code to execute commands for opening and closing the garage door.
- Develop Admin console for opening shutter without verifying and authenticating.
- Recording data of incoming vehicles and storing it in cloud platform.
- Displaying the recorded data in the mobile application.

## **1.5. Structure of the Report**

### **1.5.1. Background**

A project's background section provides a full overview of the project, including its intended audience, the solution provided to address a specific issue, the project's technical and functional requirements, and a rundown of its features. This part may also include information about similar initiatives in the same domain, as well as a comparative study to assist readers understand the project better.

### **1.5.2. Development**

The approach of choosing a development methodology appropriate for this project is described in this section. The approach chosen will depend on how simple it will be to apply and whether it will work for the particular project.

The survey analysis for the project before and after development, performed using the chosen methodology, is documented in the following phase. The project's requirements analysis, design, and construction are all recorded in accordance with the methodology's rules.

### **1.5.3. Testing and Analysis**

To test the project's outcomes, various testing, including unit testing and system testing, is done in this section. Critical analysis is also provided in light of the results.

### **1.5.4. Conclusion**

This section of the report details the limitations of the developed system, work that still needs to be done, and potential future improvements to this project. All of the restrictions and upcoming work for the project have been justified in the proper manner.

## **Chapter 2: Background**

### **2.1. About the End Users**

#### **2.1.1. End-Users**

This project serves end users that require automatic shutter opening via authentication and verification. The fundamental advantage of this technology for users is that the shutters open and close automatically, eliminating the need for manual operation. This is especially useful when the user is distracted and unable to manually manipulate the shutters, such as when their hands are engaged or if they have mobility limitations. When the end-user's license plate is registered, the vehicle is validated and verified, and the shutter opens.

#### **2.1.2. Admin**

The system administrator can easily regulate the shutters' opening and closing using their mobile smartphone. The admin console allows the user to manage the shutters as well as monitor data collected by the system's backend. This data includes vital information such as the vehicle's license plate, the success rate of the shutter opening, and the date and time of each incident.

### **2.2. Understanding the Solution**

Smart garage systems have numerous advantages, but security is a primary worry. Unauthorized access is a concern in the absence of adequate authentication and verification methods. Modern techniques such as number plate recognition are employed to ensure that only authorized users can use the system. In addition, smart garages allow users to open and close doors with little physical effort, while load cell and ultrasonic sensors, as well as USB cameras, are utilized to authenticate and identify vehicles and individuals. These safeguards ensure that only authorized individuals obtain entry and that the owner's vehicle and valuables remain safe. To create a safe and secure workplace, adequate security measures must be implemented.

**2.2.1. Solution for Users**

The goal of this technology is to increase the security of smart garages. Only users who have successfully completed the authentication and verification process will be allowed to open the shutter, preventing unauthorized individuals from gaining access. Furthermore, the system may collect data from sensors to monitor activity, making it more convenient and versatile than typical garages.

**2.2.2. Solution for Admin**

The administrator will be able to open the shutter using an admin console without the need for authentication or verification. This functionality can be used to grant access to unregistered users or visitors. It is important to note, however, that this function may offer a security concern because anyone with access to the admin console could open the shutter without sufficient authentication.

## **2.3. Similar Projects**

### **2.3.1. Project 1:**

The major goal of this project is to give clients a simple way to remotely operate their garage doors, locking and unlocking them. To do this, the project employs cutting-edge technology such as wireless keypads, radio receivers, remote chain keys, and solenoid deadbolts, which all operate in tandem to provide consumers with a smooth and hassle-free experience.

Furthermore, the project incorporates an automatic door locking system that helps to improve security by closing and locking the door after a certain length of time. This function ensures that the garage is secure even if the user forgets to close and lock the door manually.

Moreover, the project features a number plate recognition system that permits data to be stored on vehicles that enter and exit the garage. This system keeps track of the vehicles parked in the garage and can be valuable in the event of a security breach or an accident.

In general, the smart garage project strives to provide clients with a simple, secure, and effective way to manage their garage doors. The adoption of new technology makes opening and closing the garage door easier, while also ensuring that the garage is secure at all times. (Gomathy, 2022)

### **2.3.2 Project 2:**

The garage door in this project may be controlled via a webpage. Two buttons are displayed on the user interface for opening and closing the garage door. When the user presses one of these buttons, the microcontroller receives a signal and performs the required action to open or close the garage door.

There is also a display that shows the current status of the process. If the garage door is about to close, a notice will show on the user interface, and the door will be closed soon. The microcontroller monitors the user interface for signals every 30 seconds to guarantee that any modifications made by the user are instantly reflected.

The smart garage concept employs two infrared sensors to identify whether the garage door is now open or closed. When the sensors detect the position of the garage door, the information is communicated to the user interface, which is updated to reflect the status of the garage door.

The user interface also displays a log with useful information, such as the date and time of each garage door operation, as well as whether the door was opened or closed. This log keeps track of the smart garage's activity and usage, as well as a record of its previous actions. (Projects Factory, 2021)

### 2.3.3 Project 3:

This project entails building a smart garage door opener driven by a Raspberry Pi. The system works by employing a web server built with Flask, a Python microframework. This web server can accept commands from a webpage and then send them to the Raspberry Pi through the network, allowing for remote control of garage doors.

A Raspberry Pi board with Raspbian installed, connection wires, and a relay module are required to finish the project. Connecting the Raspberry Pi to the internet is also required. When all of the components are in place, connect the garage door opener mechanism to the relay's output.

A Python script is required to interface with the Raspberry Pi GPIOs and set up the web server in order to complete the project. The script stores the pin number, name, and pin condition in a dictionary and configures the pin as output, initially setting it to low. The pin state is read by the main function and stored in a variable before being delivered to the HTML page. The HTML website is built using templates, and the head section can be changed to fit individual preferences. The page provides a button for opening and closing the garage door.

In summary, this project demonstrates how IoT technology may be utilized to remotely operate objects. It also gives a useful use case for Raspberry Pi and Flask development. (Jain, 2019)



## 2.4. Comparisons

Table 1: Comparison Table

S.N.	Features	Project 1	Project 2	Project 3	This Project
1	Microprocessor	✗	✗	✓	✓
2	Microcontroller	✓	✓	✗	✗
3	Data Storing	✓	✓	✗	✓
4	Data Displaying	✗	✓	✗	✓
5	Automation	✗	✗	✗	✓
6	Web UI	✓	✓	✓	✗
7	App UI	✗	✗	✗	✓
8	Authentication	✗	✗	✗	✓
9	Verification	✗	✗	✗	✓
10	Locking	✓	✗	✗	✗

## Chapter 3: Development

### 3.1. Considered Methodologies

#### 3.1.1 Waterfall Methodology

The waterfall methodology is a project management approach that emphasizes a straight line from start to finish. This methodology, which is frequently used by engineers, is designed to rely on careful planning, detailed documentation, and sequential execution. (Adobe Communications Team, 2022)

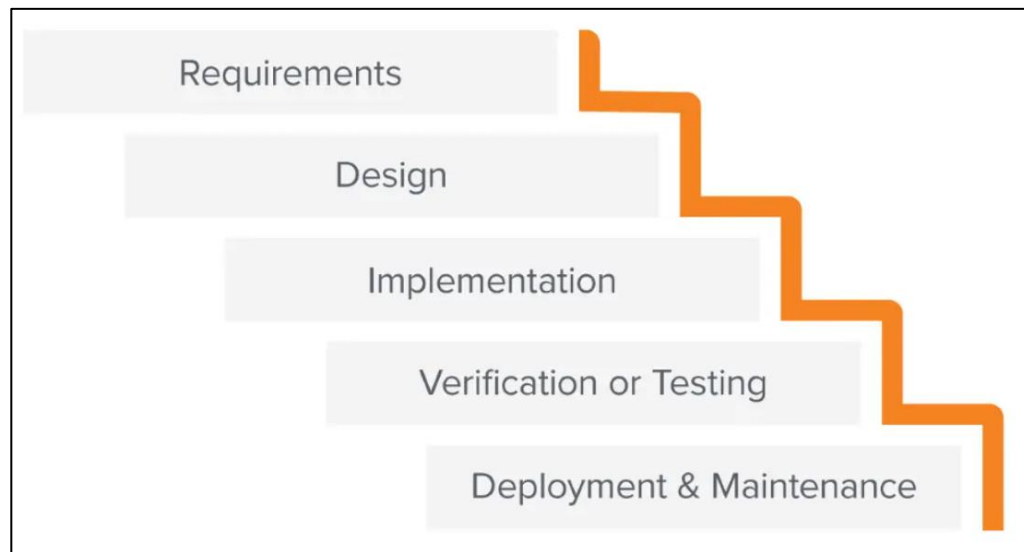


Figure 2: Waterfall Methodology (Adobe Communications Team, 2022)

Waterfall methodology was not selected because there is no room for editing. For example, if the project is in Verification or Testing stage and some design change is required then the editing to design won't be possible. The sturdiness of this methodology makes it unapplicable for this project.

##### 3.1.1.1. Advantages of Waterfall Methodology

- Waterfall module is very basic and easy to grasp, making it suitable for all types of developers.
- As one of the earliest models in software development, it ensures the development of a complete product if properly followed.

- This model's stages are explicitly separated, which aids in the definition of milestones and deadlines.

#### 3.1.1.2. Disadvantages of Waterfall Methodology

- Waterfall does not support the concept of revisions, which forces development to restart if a flaw is discovered at any point.
- This model does not support the concept of requirements changing throughout development, which is a regular occurrence in real-world development.
- Testing is not performed till the conclusion of the lifespan.
- A workable product cannot be obtained until the conclusion of the lifespan.

### **3.2. Selected Methodology**

Prototype methodology is a software development model where building, testing, and reworking is done until desired prototype is achieved. There are four types of prototype methodologies, and they are Extreme, Incremental, Throwaway and Evolutionary prototype methodology. (Martin, 2022)

This project employs the evolutionary prototyping method. It all started with a simple notion, and then more features were added over time. The initial stage was to develop an abstract specification, which gave a general concept of how the project will end out. The abstract specification was then used to create a system prototype. The prototype was tested to see if it could meet the requirements. The final system was created if it met the requirements. If it did not match the requirements, more features were added and tested until the desired system was attained.

### **3.3. Phases of Methodology**

#### **I. Develop Abstract Specification**

Creating an abstract specification for evolutionary prototyping entails outlining the project's objectives, determining which features are most important based on user requirements, specifying the scope of the system, establishing performance metrics, developing a preliminary system architecture, and refining the specifications based on feedback from stakeholders and users. This technique enables a well-defined plan for developing prototypes that meet user needs as well as system objectives.

#### **II. Build Prototype System**

Building a prototype system in evolutionary prototyping is an iterative process of designing, developing, testing, and receiving feedback. The prototype is quickly built with minimal documentation to show a certain system feature or action. The prototype is tested and developed based on feedback from end users. This procedure is done several times throughout the development phase to allow for speedy system development and testing while ensuring that the final product meets the needs of stakeholders and end-users.

### **III. Use Prototype System**

In evolutionary prototyping, testing, and updating the prototype is critical to ensuring that the system meets the needs of the final users. The development team collects user feedback, does usability testing, and makes changes to the prototype as needed. This iterative approach is performed until the system satisfies all the required criteria. The development team may construct a system that aligns with the goals and expectations of stakeholders and end-users by leveraging the prototype system and incorporating feedback from end-users.

### **IV. System Adequate?**

A system is regarded acceptable in evolutionary prototyping when it meets end-user requirements and performs as intended. To that purpose, the system is constantly tested and modified depending on feedback from end users. When the system is judged adequate, it is deployed and refined to meet changing demands.

### **V. Deliver Final System**

Delivering the final system in evolutionary prototyping entails deploying a properly tested and enhanced prototype that meets the objectives and aims defined in the abstract specification. The system should suit the needs of both stakeholders and end users, be dependable, and work as intended. Continuous monitoring and improvement guarantee that it remains adequate and meets changing requirements. Although the development phase concludes with delivery, the process of refinement should continue to assure performance and relevance.

### **3.4. Survey Results**

Surveys with commuters of various demographics were done for requirement analysis and product input throughout the development of the application and feedback on the generated application. The gist of the participant input has been reported in the following section.

Pre survey has a sample space of 16 participants, but post survey has a sample space of 9 individuals, with all post survey participants being a subset of pre survey participants.

For Visual Reference visit [Appendix 1](#) and [Appendix 2](#)

### 3.4.1. Pre-Survey Results

- a) 50% of people are aware of the IoT domain, whereas more than 43% are unaware and the remainder may be aware. More than 62% people use IoT system
- b) More than 68% of people have heard of Smart Garage, and the remainder may or may not be aware of it.
- c) The authentication and verification feature are useful, time saving, and easy to use.
- d) 50% of people desire to use the system, and more than 43% might utilize it, whereas more than 6% will not.

### 3.4.2. Post-Survey Results

- a) On a scale of 1 to 10, the notion of shutter opening employing sensors received an 8 from 55.6%, a 9 from 33.3%, and a 10 from 11.1%.
- b) On a scale of 1 to 10, the authentication function is scored 7 by 11.1%, 8 by 22.2%, 9 by 44.4%, and 10 by 22.22%.
- c) On a scale of 1 to 10, the authentication function is scored 7 by 11.1%, 8 by 22.2%, 9 by 44.4%, and 10 by 22.22%.
- d) On a scale of 1 to 10, the feature of verification is scored 7 by 11.1%, 8 by 22.2%, 9 by 33.33%, and 10 by 33.33%.
- e) The feature of opening the shutter is rated 8 by 11.1%, 9 by 33.3%, and 10 by 55.6% on a scale of 1 to 10.
- f) The feature of storing logs is rated 8 by 22.2%, 9 by 55.6%, and 10 by 22.2% on a scale of 1 to 10.
- g) The Mobile Application is rated 8 by 11.1%, 9 by 44.4%, and 10 by 44.4% on a scale of 1 to 10.
- h) Increasing the security of the mobile app while logging in is feedback received.

### **3.5. Requirement Analysis**

#### **3.5.1. Feature Requitelements**

This system's features are designed to meet the project's aim and objectives. The feature criteria are as follows:

- a)** Authenticate User
- b)** Verify User
- c)** Rest API
- d)** Login Validation
- e)** Opening Shutter by mobile application
- f)** Viewing logs in mobile app

#### **3.5.2. Software Requirements**

##### **3.5.2.1. Planning and Development**

- a) Draw.io to create Diagrams like Data flow, Communication, Sequence, and flowcharts.
- b) Excel to create Gantt Chart and plan the development timeframe
- c) Word to Create tables and document the development in report format.

##### **3.5.2.2. IDE**

- Visual Studio Code was used in frontend as well as backend. In frontend dart programming language was used where in backend python was used.

##### **3.5.2.3. Mobile Application**

- Flutter is used to develop the mobile application

##### **3.5.2.4. IoT System**

- Python is used to code the IoT System

##### **3.5.2.5. Mobile D**

- flutter:



sdk: flutter

- file\_picker: "5.2.3"
- firebase\_core: "2.3.0"
- firebase\_auth: "4.1.5"
- firebase\_analytics: "10.0.6"
- cloud\_firestore: "4.1.0"
- firebase\_storage: "11.0.6"
- get\_it: "7.2.0"
- http: ^0.13.4

#### 3.5.2.6. Python Package

- Flask to host API
- Fswebcam
- RPi.GPIO to use GPIO pins of Raspberry Pi
- HX711 to use HX711 module

#### 3.5.3. Hardware Requirements

- Raspberry Pi is the microprocessor used to operate the action and activities of this project
- USB camera to click pictures of Numberplate
- Ultrasonic Sensor to read the distance from the object
- Load cell Sensor to read the weight of the object
- HX711 to convert signals from load cell to weight
- Components to connect these modules(breadboard and jumper wires)

### 3.6. Design

To make the development process easier, the entire application development pipeline must first be planned. This provides a detailed picture of the development process and makes its characteristics easy to understand. To do this, the project was divided into specialized elements, and high-level designs for the entire project, represented by UML diagrams, were constructed. This ensures a better grasp of the project's structure before moving forward with actual development.

#### 3.6.1. Application Design



Figure 3: Login Page Design

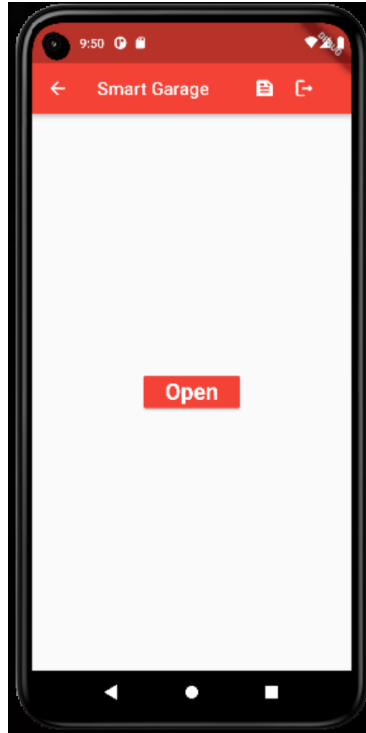


Figure 4: Home Page Design

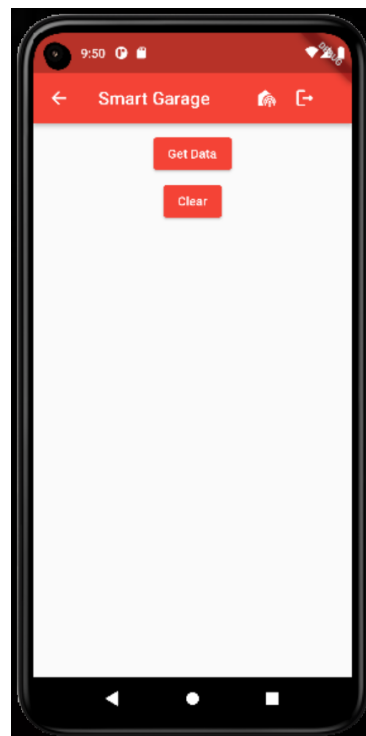


Figure 5: Log Info Page Design

### 3.6.2. System Design

#### 3.6.2.1. Use Case Diagram

The use case diagram is used to represent the overall system features, including how the characteristics connect to the individuals who use the program. Actors are people who utilize the program, whereas use cases are the features itself.

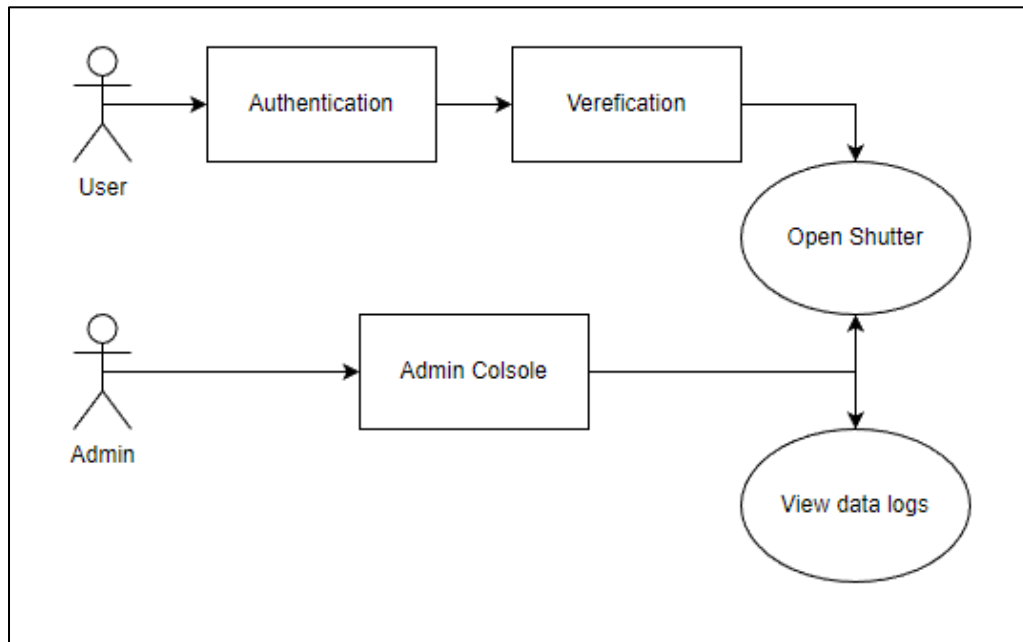


Figure 6: System Diagram

### 3.6.2.1. Activity Diagram

An activity diagram, like a flowchart or a data flow diagram, visually represents a succession of actions or the flow of control in a system. They can also use a use case graphic to describe the steps.

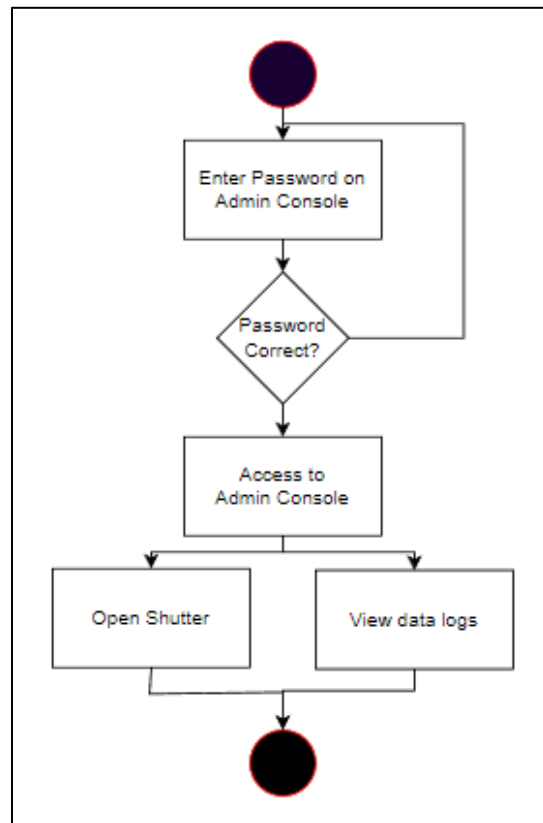


Figure 7: Activity Diagram for Admin Console

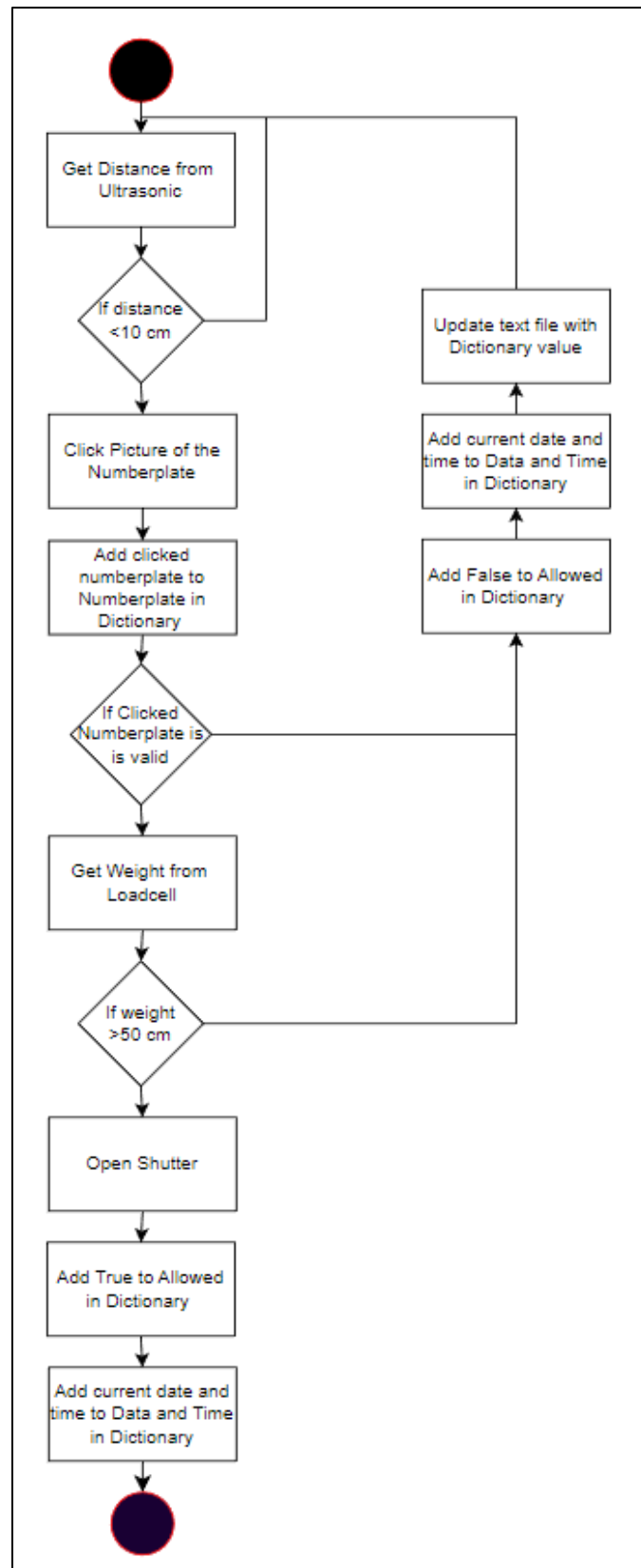


Figure 8: Activity Diagram for IoT System

### 3.6.3. Feature Design

The working designs for individual system core features have been given in the following parts.

#### 3.6.3.1. Verification

##### a) Data Flow Diagram

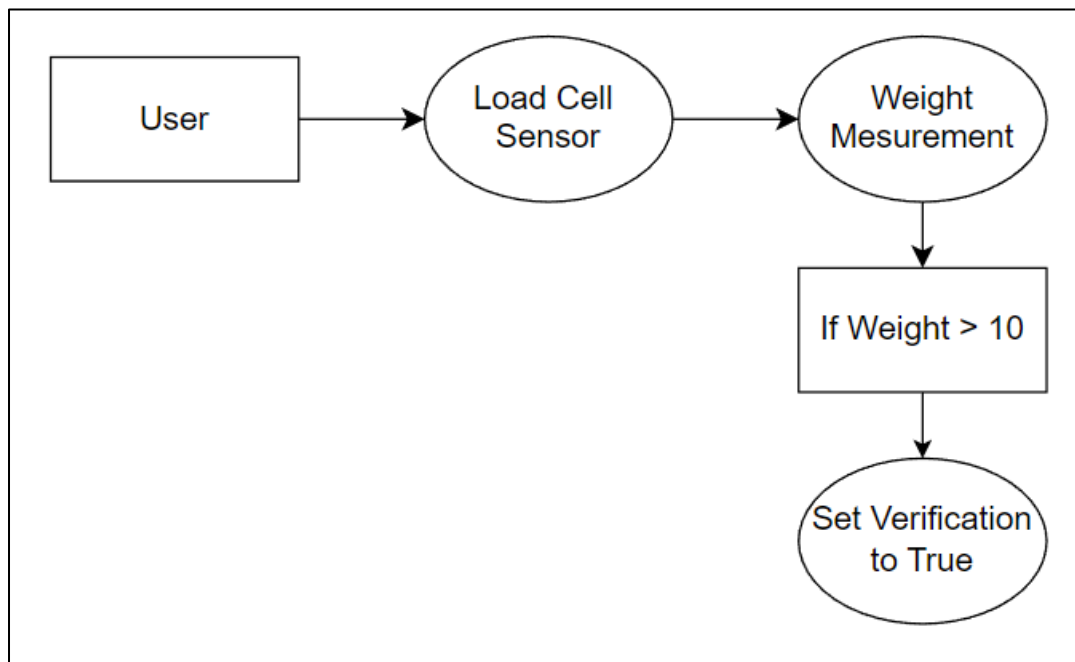


Figure 9: Data Flow Diagram for Verification

##### b) Communication Diagram

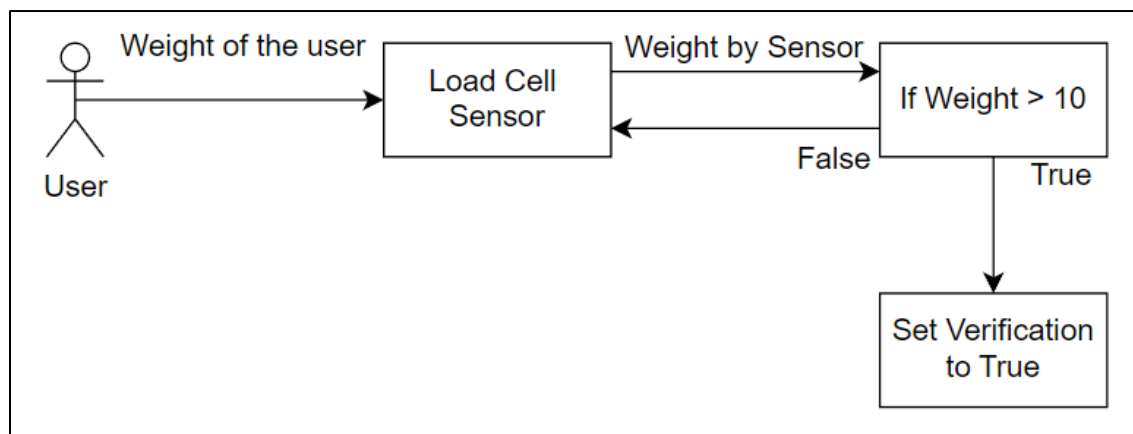
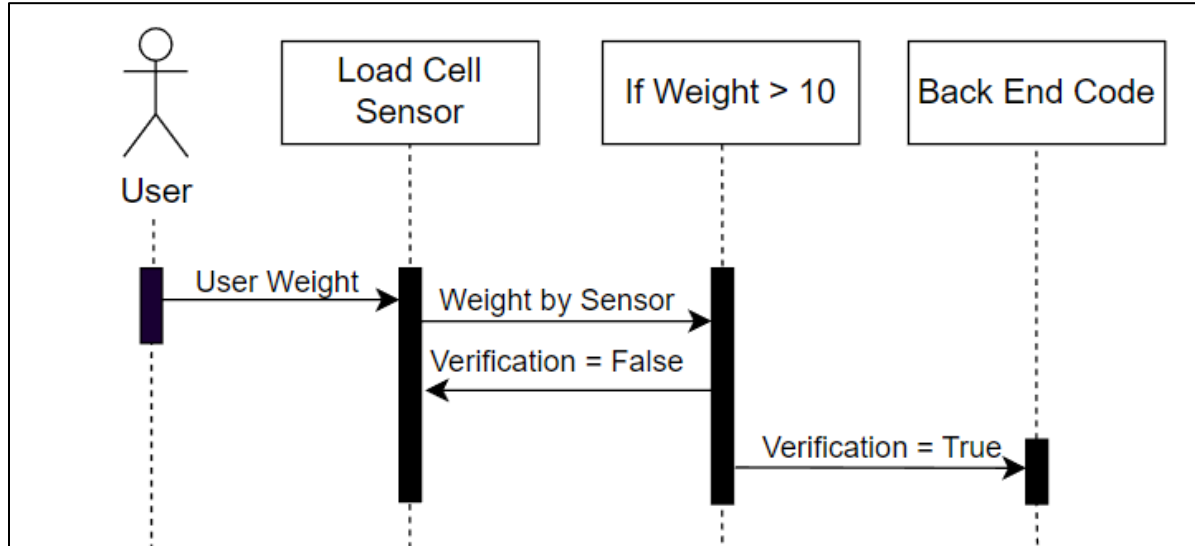


Figure 10: Communication Diagram for Verification

## c) Sequence Diagram

*Figure 11: Sequence Diagram for Verification*



## 3.6.3.2. Authentication

## a) Data Flow Diagram

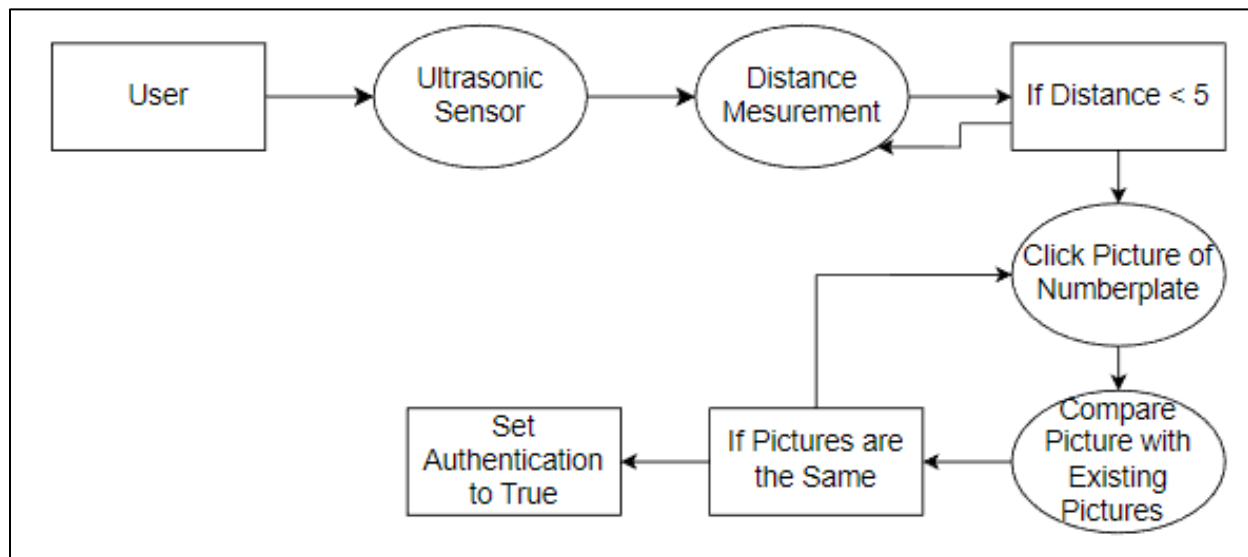


Figure 12: Data Flow Diagram for Authentication

## b) Communication Diagram

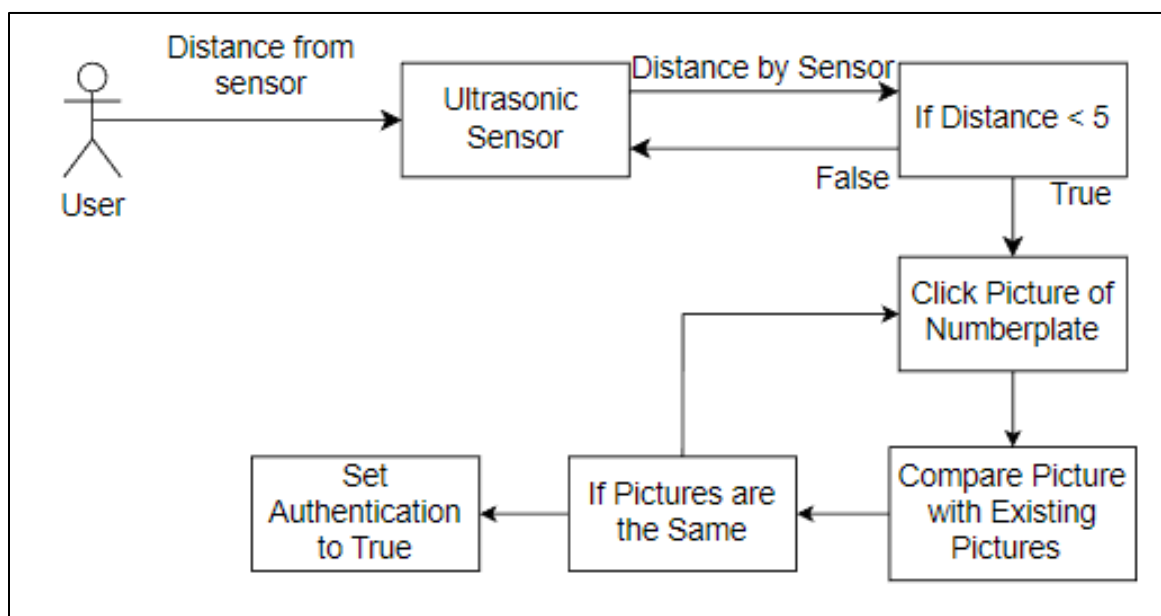
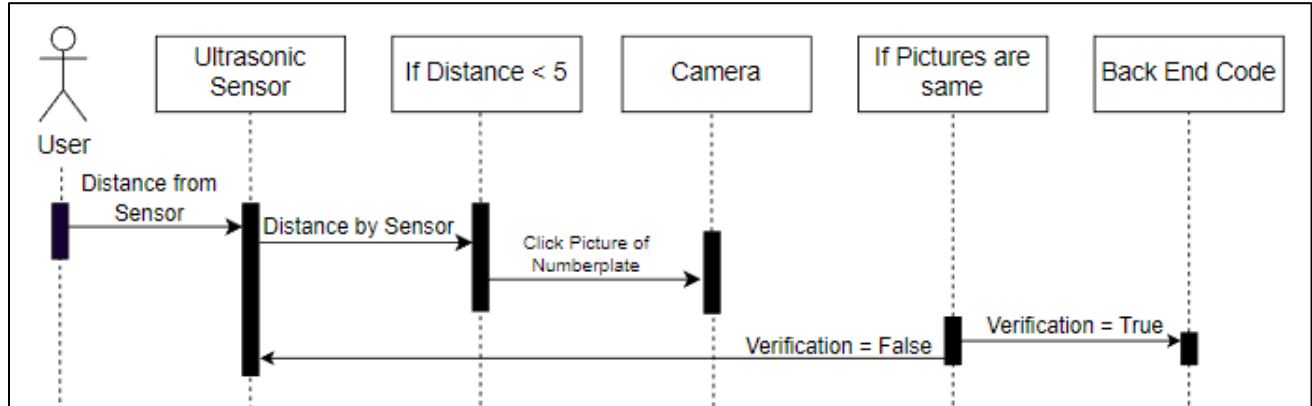


Figure 13: Communication Diagram for Authentication

## c) Sequence Diagram

*Figure 14: Sequence Diagram for Authentication*

## 3.6.3.3. Open Button

## a) Data Flow Diagram

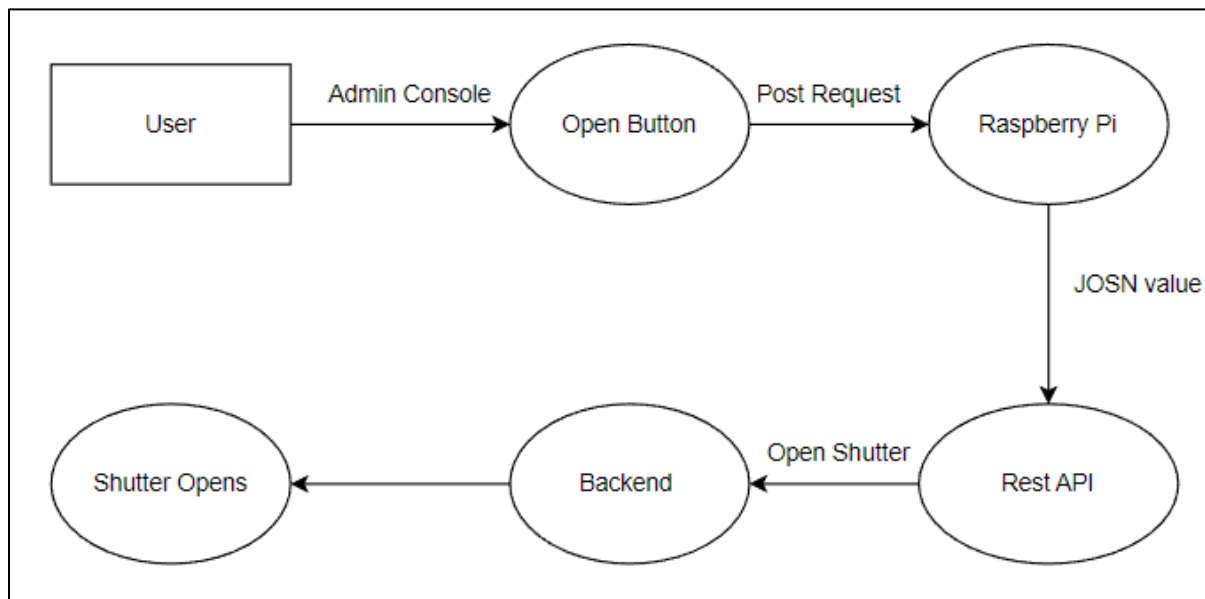


Figure 15: Data Flow Diagram for Open Button

## b) Communication Diagram

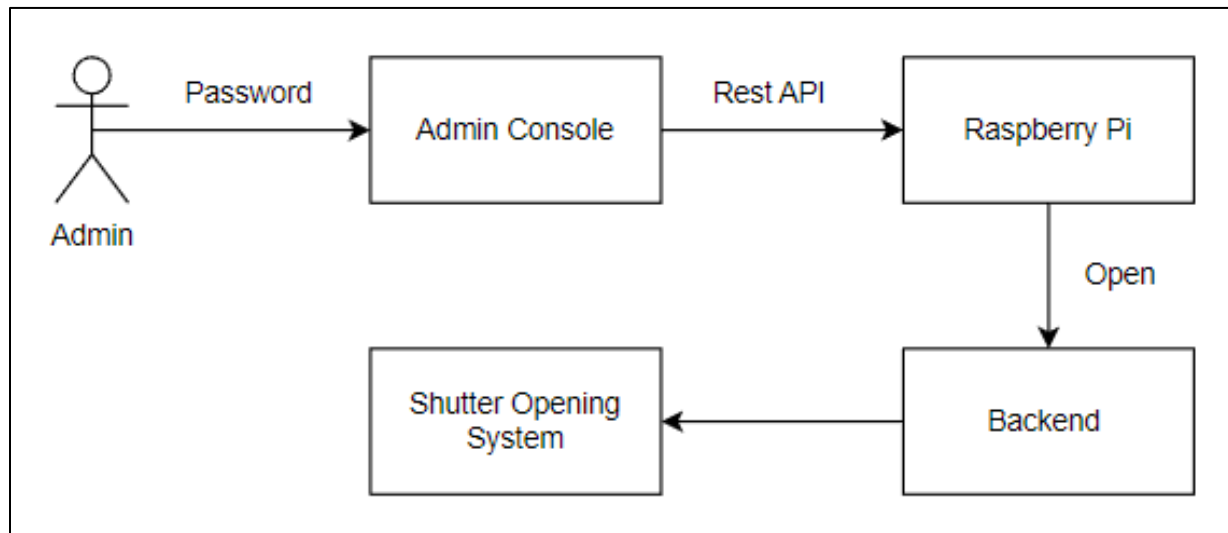
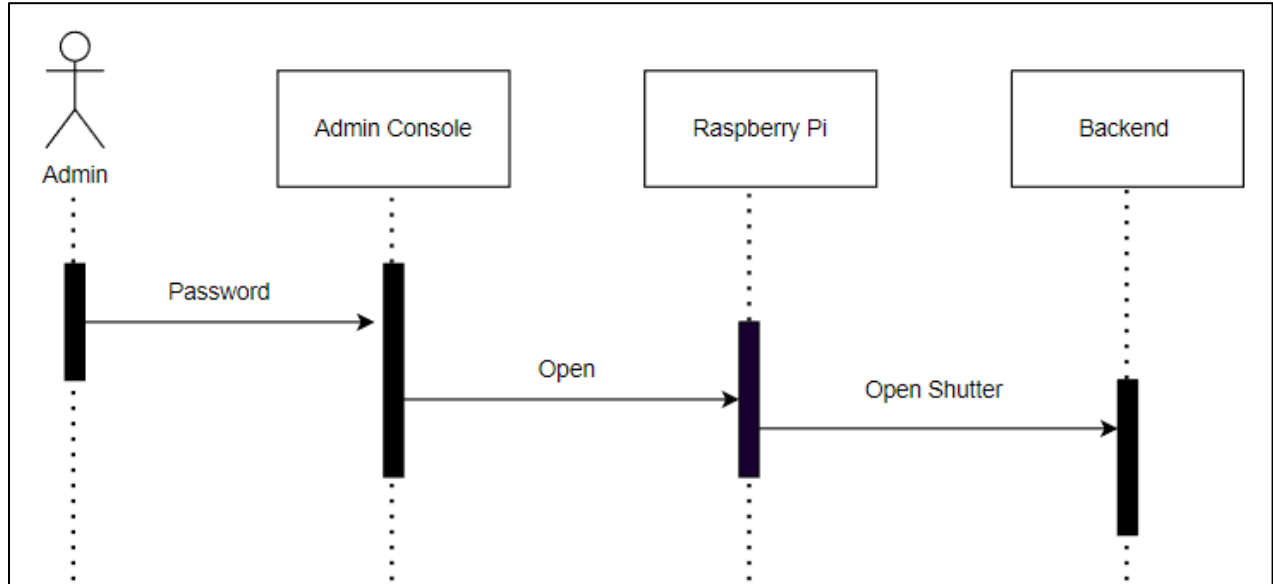


Figure 16: Communication Diagram for Open Button

## c) Sequence Diagram

*Figure 17: Sequence Diagram for Open Button*

## 3.6.3.4. Get data Button

## a) Data Flow Diagram

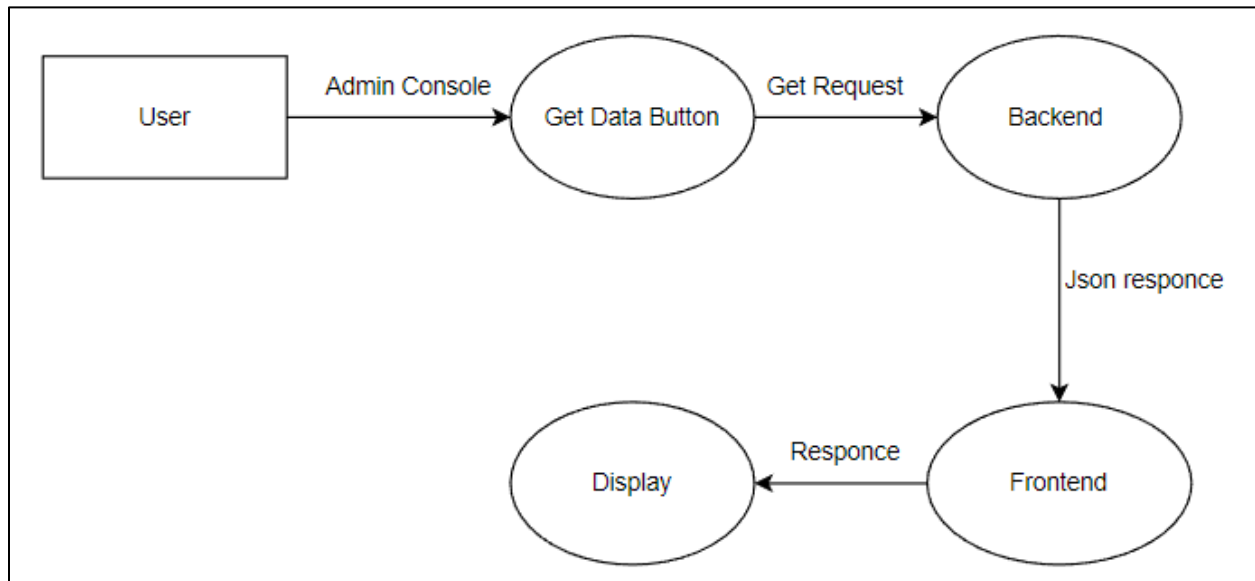


Figure 18: Data Flow Diagram for Get data Button

## b) Communication Diagram

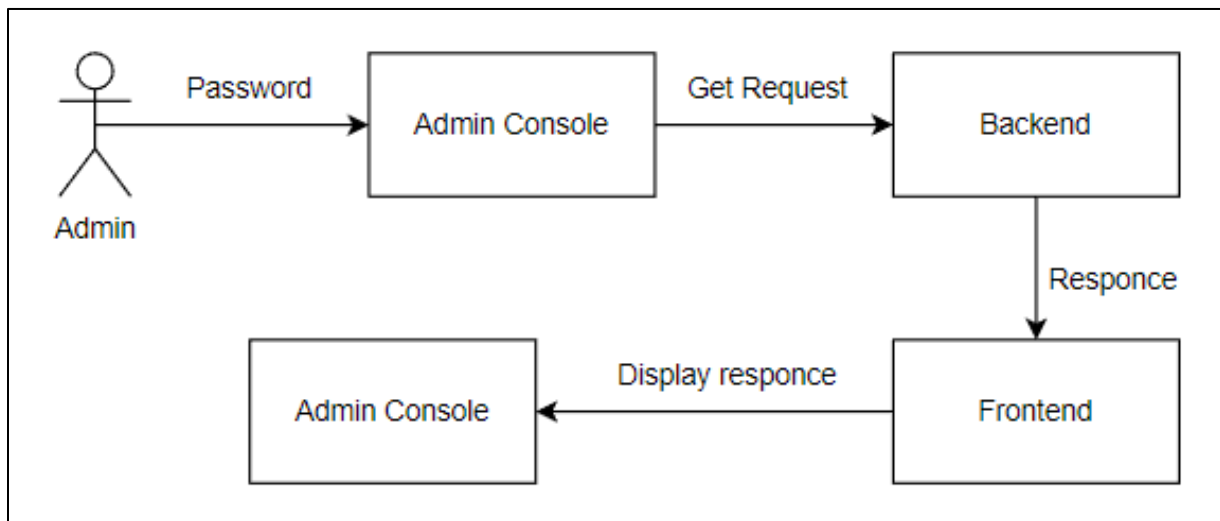


Figure 19: Communication Diagram for Get data Button

## c) Sequence Diagram

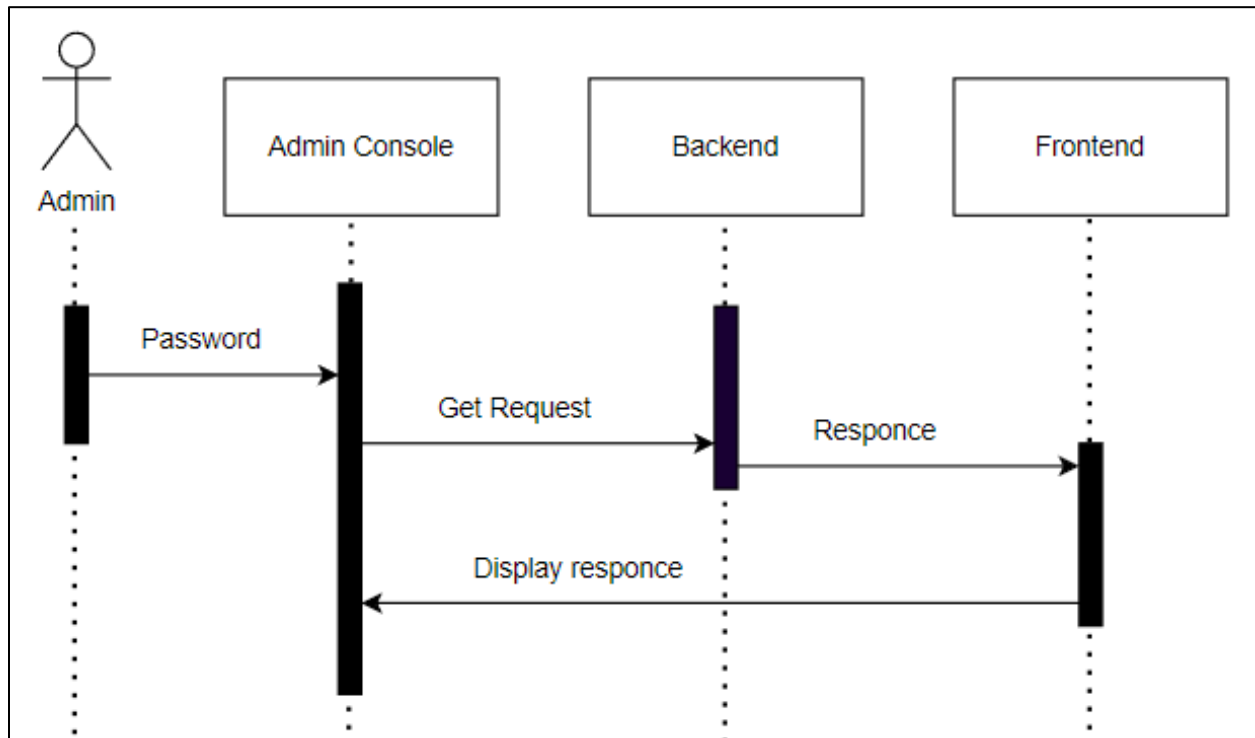


Figure 20: Sequence Diagram for Get data Button

## 3.6.3.5. Admin Login

## a) Data Flow Diagram

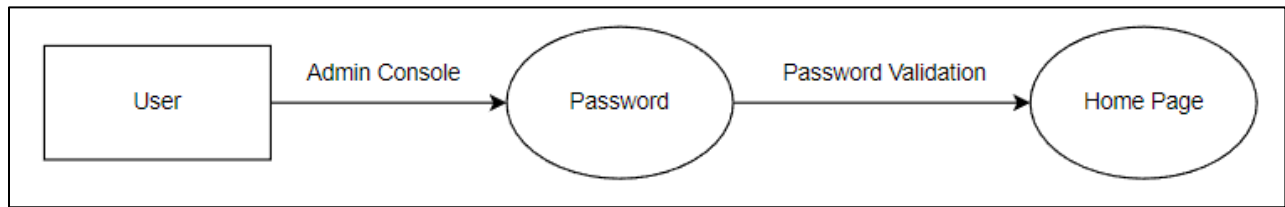


Figure 21: Data Flow Diagram for Admin Login

## b) Communication Diagram

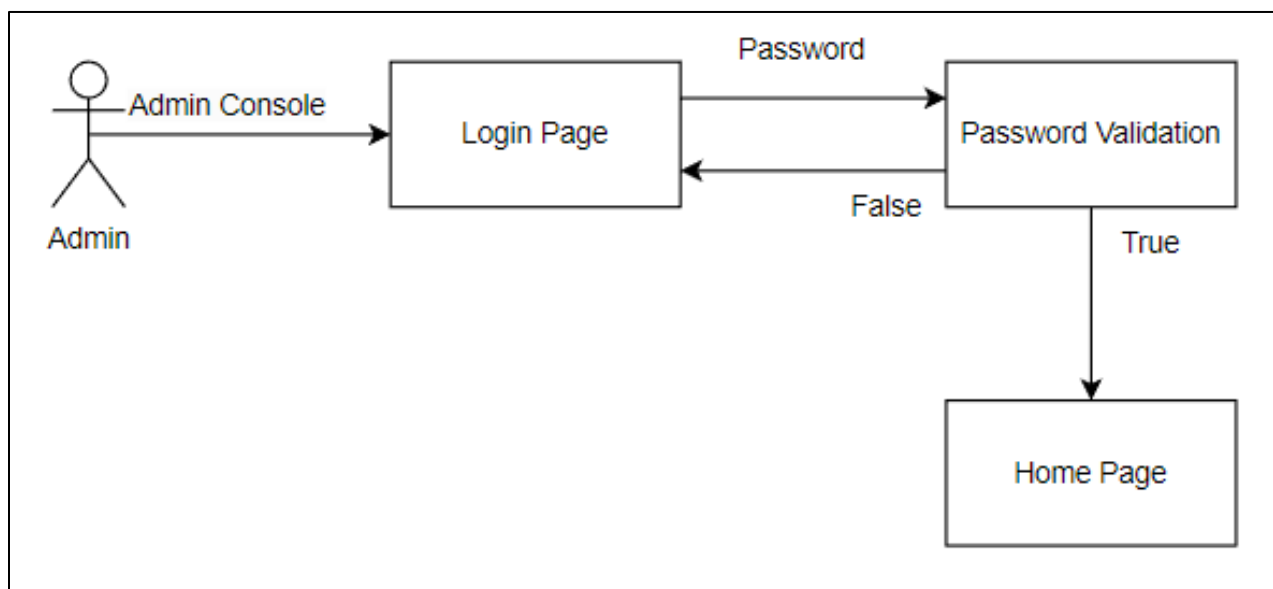
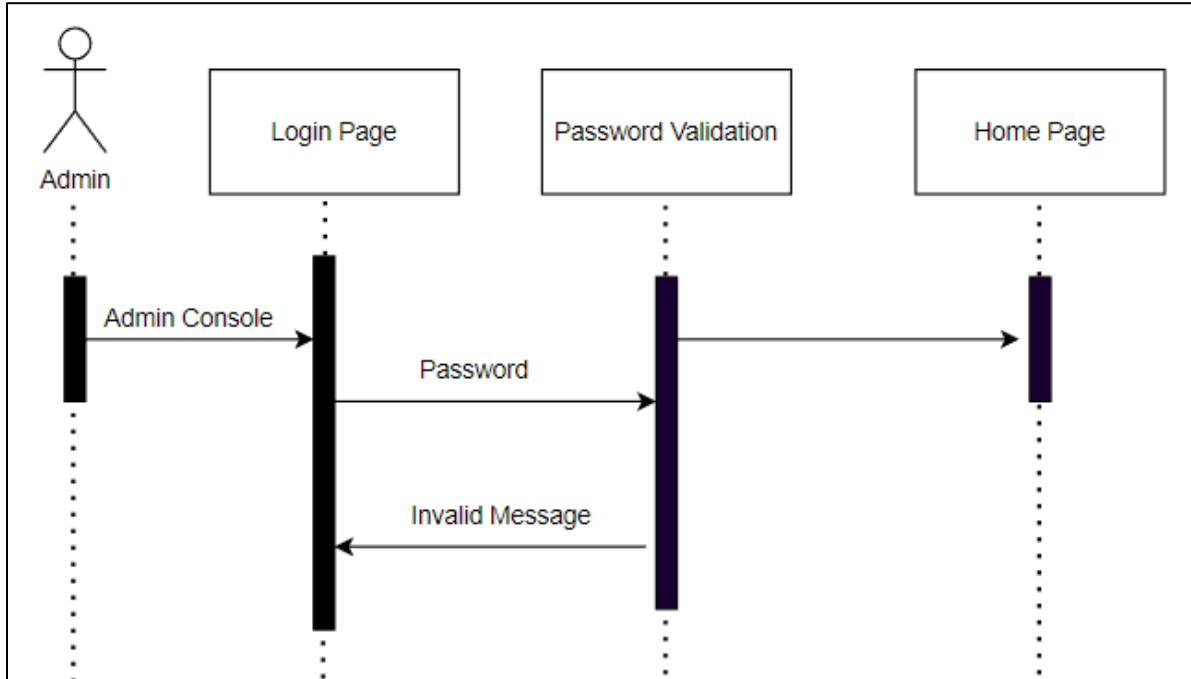


Figure 22: Communication Diagram for Admin Login

## c) Sequence Diagram

*Figure 23: Sequence Diagram for Admin Login*



## 3.6.3.6. Log/Home page Button

## a) Data Flow Diagram

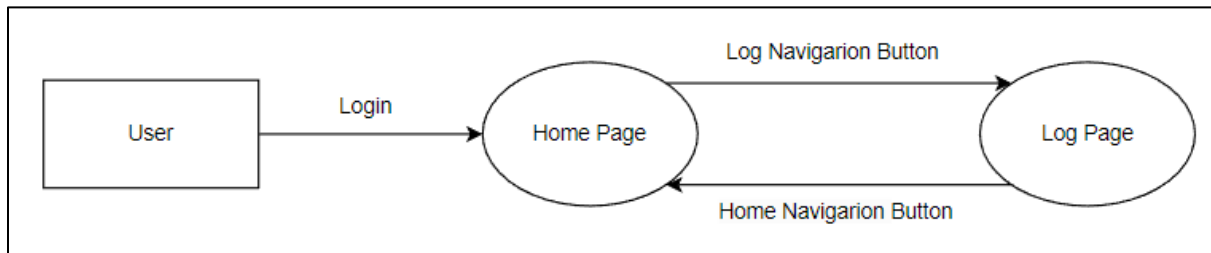


Figure 24: Data Flow Diagram for Log/Home page Button

## b) Communication Diagram

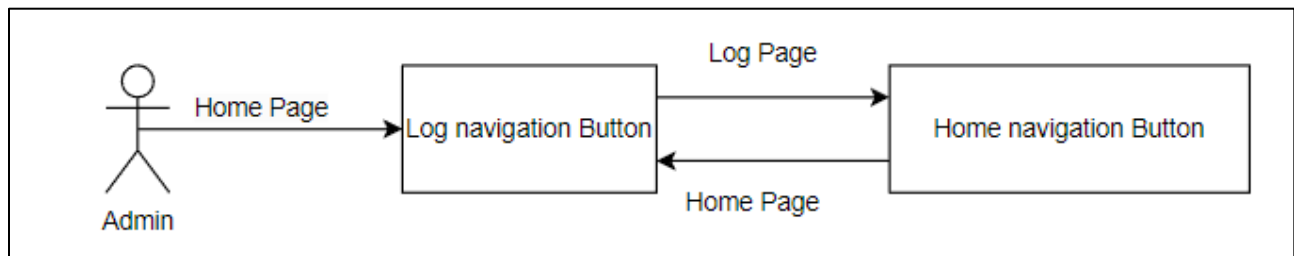


Figure 25: Communication Diagram for Log/Home page Button

## c) Sequence Diagram

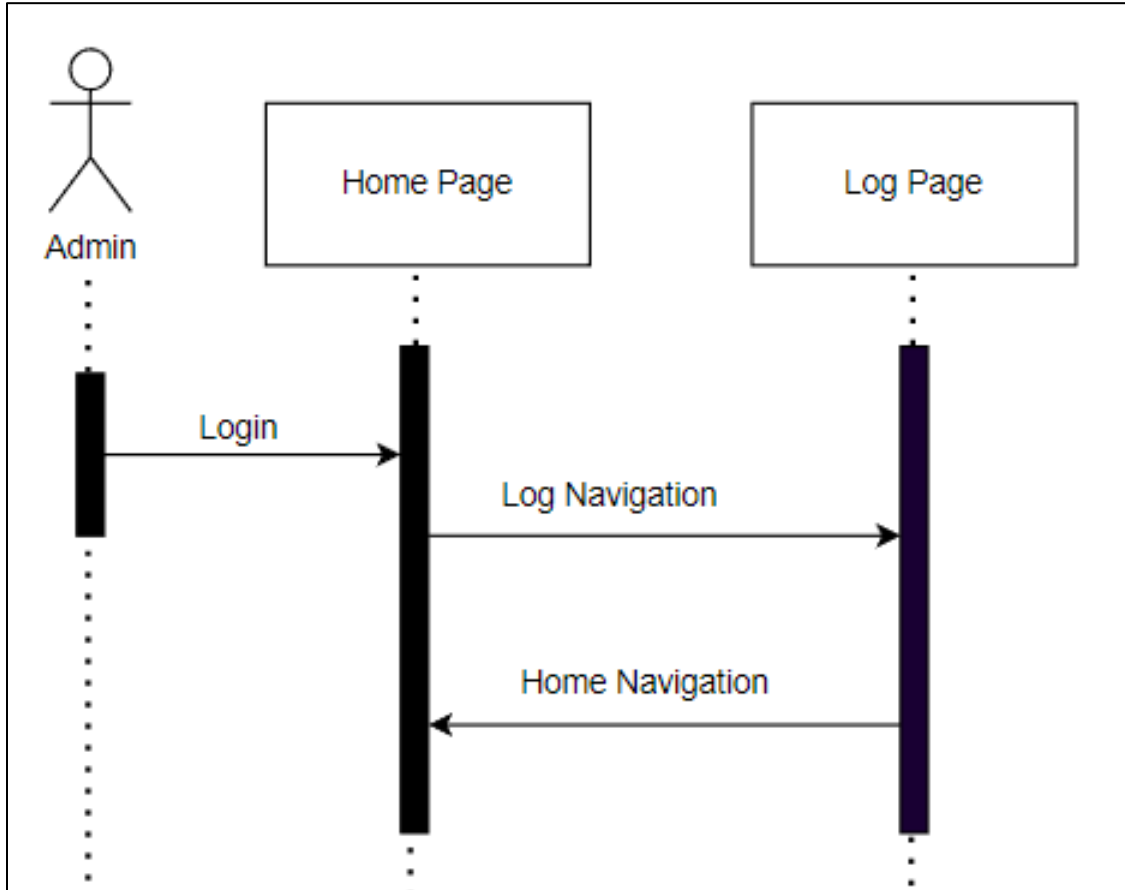
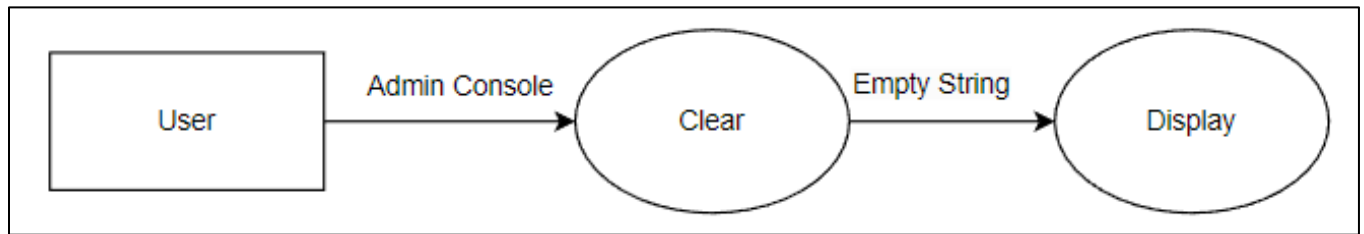


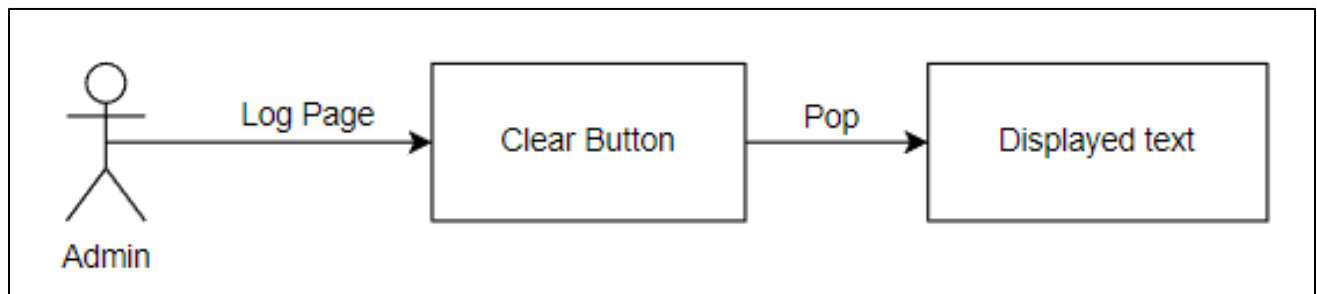
Figure 26: Sequence Diagram for Log/Home page Button

## 3.6.3.7. Clear Button

## a) Data Flow Diagram

*Figure 27: Data Flow Diagram for Clear Button*

## b) Communication Diagram

*Figure 28: Communication Diagram for Clear Button*

## c) Sequence Diagram

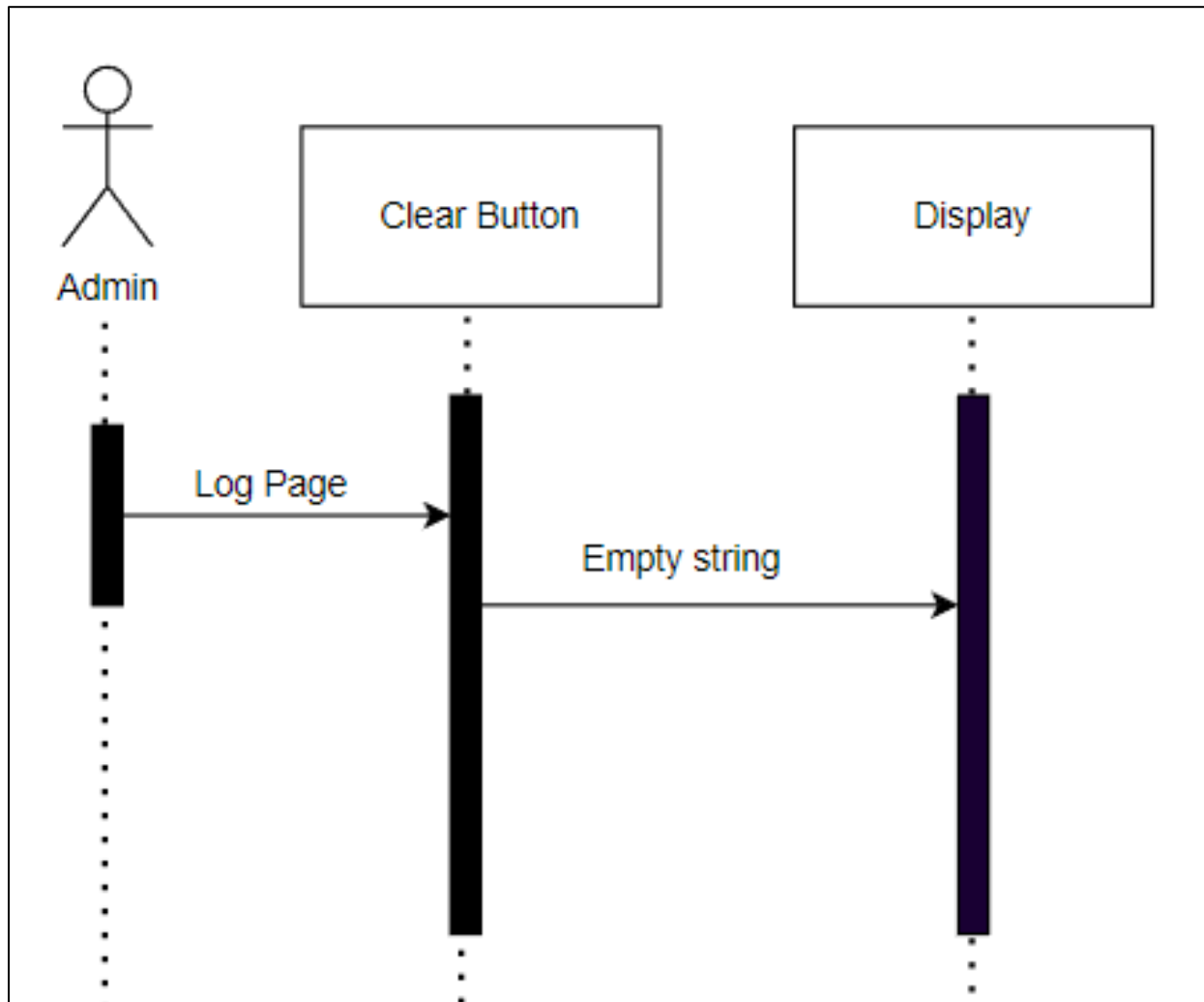


Figure 29: Sequence Diagram for Clear Button

## 3.6.3.8. Logout Button

## a) Data Flow Diagram

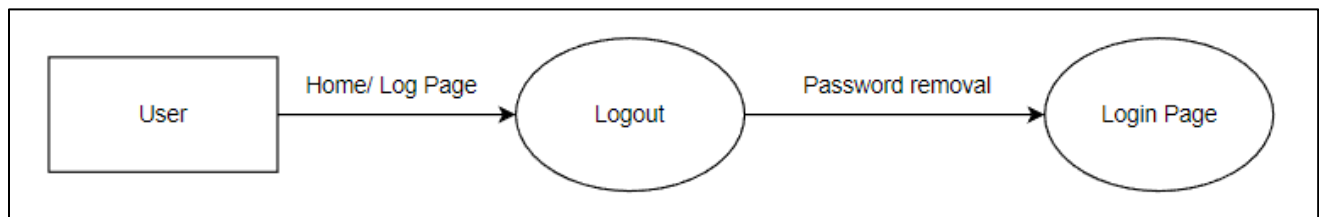


Figure 30: Data Flow Diagram for Logout Button

## b) Communication Diagram

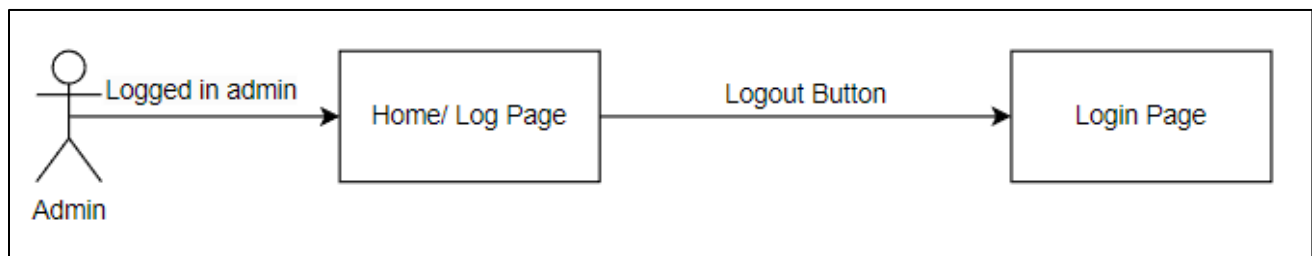


Figure 31: Communication Diagram for Logout Button

## c) Sequence Diagram

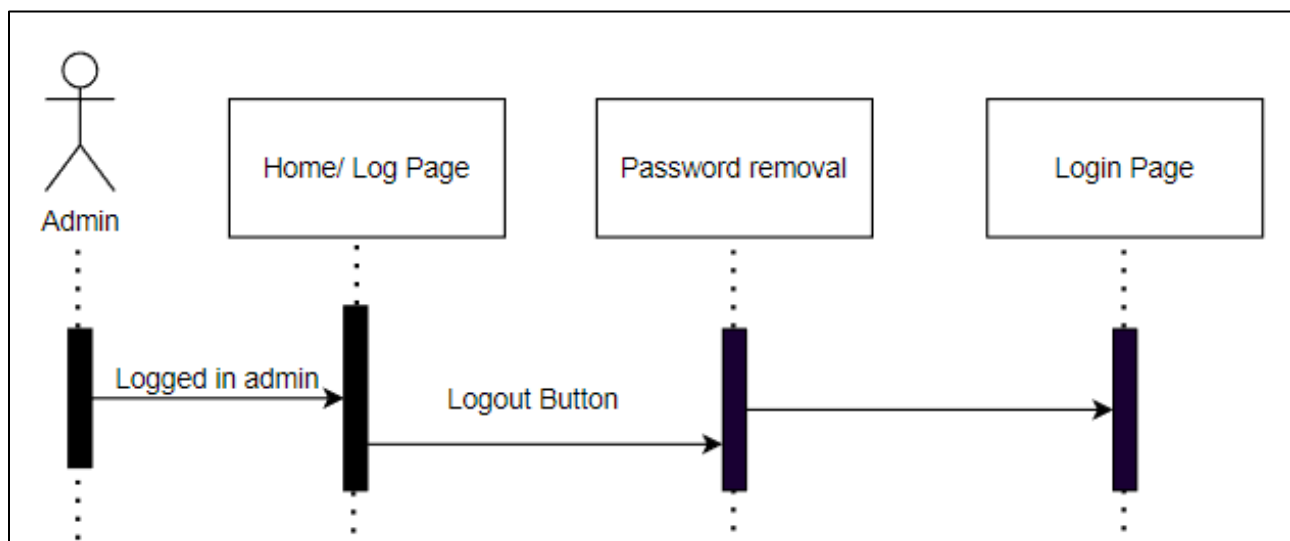


Figure 32: Sequence Diagram for Logout Button

For more Designs like Gantt chart, Work breakdown structure, block diagram and hardware arcature design go to [Appendix 4](#).

### 3.7. Implementation

The activities to accomplish this project were carried out in accordance with the Gantt chart in the appendix section.

For Sample code refer to [Appendix 3](#).

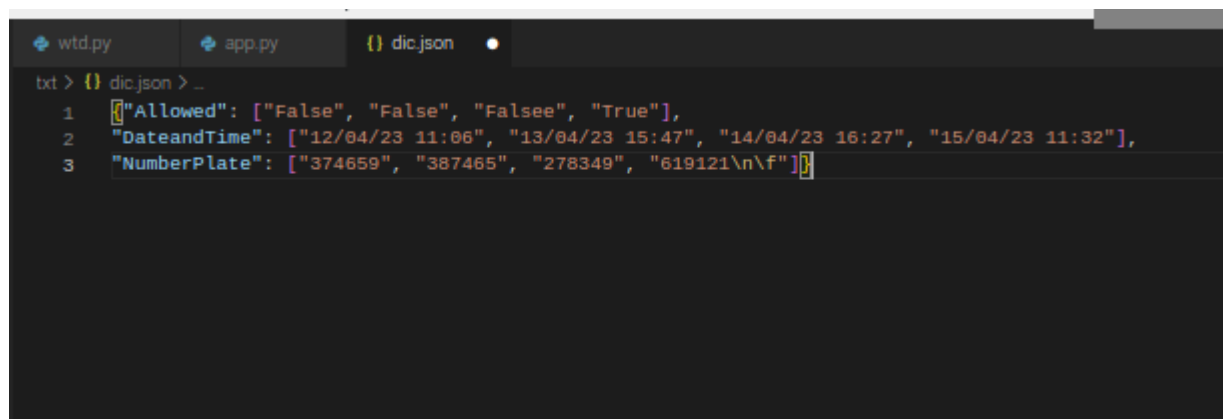
#### 3.7.1. IoT System

The backend code was written in accordance with the project's specifications in order to fulfil all of the project's requirements.

##### 3.7.1.1. Opening json file and storing it in variables

```
62 """Main"""
63 file = open("txt/dic.json")
64 dic = json.load(file)
65 NumPlate = dic["NumberPlate"]
66 DateNTime = dic["DateandTime"]
67 Allow = dic["Allowed"]
```

Figure 33: Opening json file and extracting values



```
txt > {} dic.json > ...
1 [{"Allowed": ["False", "False", "False", "True"],
2  "DateandTime": ["12/04/23 11:06", "13/04/23 15:47", "14/04/23 16:27", "15/04/23 11:32"],
3  "NumberPlate": ["374659", "387465", "278349", "619121\n\nf"]}]
```

Figure 34: JSON File

##### 3.7.1.2. Click function

```
40 def Click():
41     cmd = 'fswebcam 640x480 -pYUYV test.jpg'
42     os.system('fswebcam 640x480 -pYUYV currentNum/test.jpg')
```

Figure 35: Command for fswebcam to click Picture

## 3.7.1.3. OCR function

```
43 def OCR(NumPlate):
44     i = 1
45     os.system("tesseract currentNum/5.jpg numTxt/output --dpi 300 --oem 1 --psm 6")
46     currentFile = open("numTxt/output.txt", "r")
47     CurrentNum = currentFile.read()
48     while i <= 5:
49         os.system("tesseract validNum/" + str(i) + ".jpg numTxt/output" + str(i) + " --dpi 300 --oem 1 --psm 6")
50         existingFile = open("numTxt/output" + str(i) + ".txt", "r")
51         ExistingNum = existingFile.read()
52         if ExistingNum == CurrentNum:
53             print("Authentication Is Successful")
54             NumPlate.extend([str(CurrentNum)])
55             return [True, NumPlate]
56         i = i + 1
57     NumPlate.extend([str(CurrentNum)])
58     return [False, NumPlate]
```

*Figure 36: Use of Tesseract OCR*

## 3.7.1.4. Authentication and Verification

```

73 while Run == True:
74     Distance = Ultrasonic()
75     print("Trying Authentication ...")
76     while Distance <= 20:
77         Click()
78         tempList = OCR(NumPlate)
79         AUTH = tempList[0]
80         NumPlate = tempList[1]
81         if AUTH == True:
82             break
83         else:
84             Distance = Ultrasonic()
85             print("Trying Authentication Again ...")
86     weight = Loadcell()
87     while weight >= 1:
88         print("Trying Verification")
89         VFON = True
90         if VFON == True:
91             print("Verification is Successful")
92             break
93         else:
94             weight = Loadcell()
95             print("Trying Verification Again ...")
96     if AUTH == True & VFON == True:
97         Open()
98         Allow.extend(["True"])
99         dt = datetime.datetime.now()
100         dnt = dt.strftime("%d/%m/%y %H:%M")
101         DateTime.extend([str(dnt)])
102         Run = False
103     else:
104         Allow.extend(["False"])

```

Figure 37: Authentication and verification backend code

## 3.7.1.5. Updating Json file

```

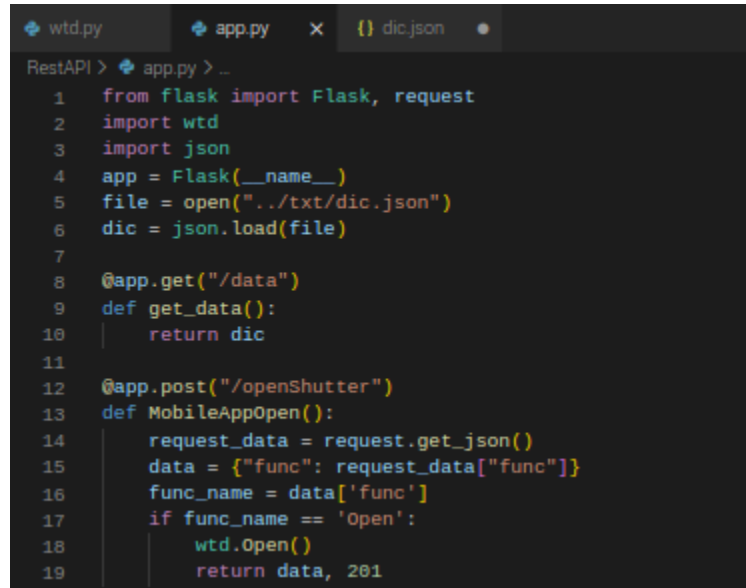
105 with open("txt/dic.json","w") as file:
106     json.dump(dic, file)

```

Figure 38: Writing to json file



## 3.7.1.6. Rest API



The screenshot shows a code editor with three tabs: 'wtd.py', 'app.py', and 'dic.json'. The 'app.py' tab is active, displaying the following Python code:

```
RestAPI > app.py > ...
1  from flask import Flask, request
2  import wtd
3  import json
4  app = Flask(__name__)
5  file = open("../txt/dic.json")
6  dic = json.load(file)
7
8  @app.get("/data")
9  def get_data():
10     return dic
11
12  @app.post("/openShutter")
13  def MobileAppOpen():
14     request_data = request.get_json()
15     data = {"func": request_data["func"]}
16     func_name = data['func']
17     if func_name == 'Open':
18         wtd.Open()
19     return data, 201
```

*Figure 39: Rest API*

### 3.7.2. Mobile Application

Mobile application is developed as a platform for admin console, where it can be used to control the shutter and view saved logs.

#### 3.7.2.1. Main

```
1  import 'package:flutter/material.dart';
2  import 'package:smartgarrage/Pages/HomePage.dart';
3  import 'package:smartgarrage/Pages/Login.dart';
4  import 'package:firebase_core/firebase_core.dart';
5  import 'package:get_it/get_it.dart';
6  import 'package:smartgarrage/Services/firebase_service.dart';
7  import 'package:smartgarrage/Pages/LogInfo.dart';
8  void main() async {
9    WidgetsFlutterBinding.ensureInitialized();
10   await Firebase.initializeApp();
11   GetIt.instance.registerSingleton<FirebaseService>(
12     FirebaseService(),
13   );
14   runApp(const SmartGarage( ));
15 }
16 class SmartGarage extends StatelessWidget {
17   const SmartGarage({Key? key}) :super(key: key);
18   @override
19   Widget build(BuildContext context) {
20     return MaterialApp(
21       title: 'Smart Garage',
22       theme: ThemeData(
23         primarySwatch: Colors.red,
24       ), // ThemeData
25       initialRoute: 'Login',
26       routes: {
27         'Login': (context) => Login(),
28         'Home': (context) => HomePage(),
29         'Log': (context) => LogInfo(),
30       },
31     ); // MaterialApp
32   }
33 }
```

Figure 40: Main Page

## 3.7.2.2. Login Page

```

1 // ignore: file_names
2 import 'package:flutter/material.dart';
3 import 'package:smartgarage/Pages/HomePage.dart';
4 class Login extends StatefulWidget{
5   const Login({super.key});
6   @override
7   State<StatefulWidget> createState(){
8     return _LoginState();
9   }
10 }
11
12 class _LoginState extends State<Login>{
13   double? _deviceHeight, _deviceWidth;
14   final GlobalKey<FormState> _loginFormkey = GlobalKey<FormState>();
15   String? _psd;
16   @override
17   Widget build(BuildContext context) {
18     _deviceHeight = MediaQuery.of(context).size.height;
19     _deviceWidth = MediaQuery.of(context).size.width;
20     return Scaffold(
21       body: SafeArea(
22         child: Container(
23           padding: EdgeInsets.symmetric(horizontal: _deviceWidth! * 0.05,
24           ), // EdgeInsets.symmetric
25           child: Center(
26             child: Column(
27               mainAxisAlignment: MainAxisAlignment.spaceAround,
28               mainAxisAlignment: MainAxisAlignment.max,
29               crossAxisAlignment: CrossAxisAlignment.center,
30               children: [
31                 _titleWidget(),
32                 _loginForm(),
33                 _loginButton(),
34               ],
35             ), // Column
36           ), // Center
37         ), // Container
38       ), // SafeArea
39     ); // Scaffold
40 }

```

Figure 41: Login Page 1

```
41 Widget _titleWidget() {
42   return const Text("Smart Garage",
43     style: TextStyle(
44       color: Color.fromARGB(255, 255, 0, 0),
45       fontSize: 40,
46       fontWeight: FontWeight.w600,
47     ) // TextStyle
48   ); // Text
49 }
50 Widget _loginButton(){
51   return MaterialButton(
52     onPressed: _loginUser,
53     minWidth: _deviceWidth! * 0.70,
54     height: _deviceHeight! * 0.06,
55     color: Colors.red,
56     child: const Text("Login",
57       style: TextStyle(
58         color: Colors.white,
59         fontSize: 25,
60         fontWeight: FontWeight.w600,
61       ), // TextStyle
62     ), // Text
63   ); // MaterialButton
64 }
65 Widget _loginForm() {
66   return SizedBox(
67     height: _deviceHeight! * 0.20,
68     child: Form(
69       key: _loginFormkey,
70       child: Column(
71         mainAxisAlignment: MainAxisAlignment.spaceBetween,
72         mainAxisSize: MainAxisSize.max,
73         crossAxisAlignment: CrossAxisAlignment.center,
74         children: [
75           _admin(),
76           _passwordTextField(),
77         ],
78       ), // Column
79     ), // Form
80   ); // SizedBox
81 }
```

Figure 42: Login Page 2

```
82   Widget _admin(){
83     return const Text("Admin",
84       style: TextStyle(
85         color: Color.fromARGB(255, 0, 0, 0),
86         fontSize: 35,
87         fontWeight: FontWeight.w600,
88       ) // TextStyle
89     ); // Text
90   }
91   Widget _passwordTextField(){
92     return TextFormField(
93       obscureText: true,
94       decoration: const InputDecoration(hintText: "Password..."),
95       onSave: (_value){
96         setState(() {
97           _psd = _value;
98         });
99       },
100      validator: (_value) => _value == "SmartGarage"
101        ? null
102        : "Please enter the correct password",
103      ); // TextFormField
104    }
105    void _loginUser(){
106      print(_loginFormkey.currentState!.validate());
107      if (_loginFormkey.currentState!.validate()) {
108        Navigator.push(context, MaterialPageRoute(builder: (BuildContext _context){
109          return HomePage();
110        })); // MaterialPageRoute
111      }
112    }
113  }
```

Figure 43: Login Page 3

## 3.7.2.3. Home Page

```
1 import 'dart:convert';
2 import 'package:http/http.dart' as http;
3 import 'package:flutter/material.dart';
4 import 'package:smartgarage/Services/firebase_service.dart';
5 class HomePage extends StatelessWidget {
6   double? _deviceHeight, _deviceWidth;
7   FirebaseService? _firebaseService;
8   final usernameController = TextEditingController();
9   final passwordController = TextEditingController();
10  @override
11  Widget build(BuildContext context) {
12    _deviceHeight = MediaQuery.of(context).size.height;
13    _deviceWidth = MediaQuery.of(context).size.width;
14    return Scaffold(
15      appBar: AppBar(
16        title: const Text("Smart Garage")
17      ), // Text
18      actions: [
19        Padding(
20          padding: const EdgeInsets.only(right: 20),
21          child: GestureDetector(
22            onTap: (){
23              Navigator.pushNamed(context, 'Log');
24            },
25            child: const Icon(Icons.feed_rounded),
26          ), // GestureDetector
27        ), // Padding
28        Padding(
29          padding: const EdgeInsets.only(right: 40),
30          child: GestureDetector(
31            onTap:() async {
32              await _firebaseService?.logout();
33              Navigator.popAndPushNamed(context, 'Login');
34            },
35            child: const Icon(Icons.logout_rounded),
36          ), // GestureDetector
37        ), // Padding
38      ],
39    ), // AppBar
```

Figure 44: Home Page 1

```

40     body: SafeArea(
41       child: Container(
42         padding: EdgeInsets.symmetric(vertical: _deviceHeight! * 0.3,
43           ), // EdgeInsets.symmetric
44       child: Center(
45         child: Column(
46           mainAxisAlignment: MainAxisAlignment.spaceAround,
47           mainAxisAlignment: MainAxisAlignment.max,
48           crossAxisAlignment: CrossAxisAlignment.center,
49           children: [
50             _openButton(),
51           ],
52         ), // Column
53       ), // Center
54     ), // Container
55   ), // SafeArea
56 ); // Scaffold
57 }
58 Widget _HomePage(){
59   return Container(
60     height: _deviceHeight!*0.3,
61     child: Column(
62       mainAxisAlignment: MainAxisAlignment.spaceBetween,
63       mainAxisAlignment: MainAxisAlignment.max,
64       crossAxisAlignment: CrossAxisAlignment.center,
65     ), // Column
66   ); // Container
67 }

```

Figure 45: Home Page 2

```

68 Widget _openButton(){
69   return MaterialButton(
70     onPressed: () {
71       sendPostRequest();
72     },
73     minWidth: _deviceWidth! * 0.30,
74     height: _deviceHeight! * 0.05,
75     color: Colors.red,
76     child: const Text("Open",
77       style: TextStyle(
78         color: Colors.white,
79         fontSize: 25,
80         fontWeight: FontWeight.w600,
81       ), // TextStyle
82     ), // Text
83   ); // MaterialButton
84 }
85 Future<void> sendPostRequest() async {
86   final url = Uri.parse('http://192.168.1.139:5000/openShutter');
87   final headers = {'Content-Type': 'application/json'};
88   final body = jsonEncode({
89     "func": "Open"
90   });
91   final response = await http.post(url, headers: headers, body: body);
92   if (response.statusCode == 201) {
93     print('POST request successful');
94   } else {
95     print('POST request failed with status: ${response.statusCode}');
96   }
97 }
98 }

```

Figure 46: Home Page 3

## 3.7.2.4. Log Information Page

```

1  import 'dart:convert';
2  import 'dart:io';
3  import 'package:flutter/material.dart';
4  import 'package:path_provider/path_provider.dart';
5  import 'package:smartgarage/Services/firebase_service.dart';
6  import 'package:http/http.dart' as http;
7  class LogInfo extends StatefulWidget {
8    const LogInfo({Key? key}) : super(key: key);
9    @override
10   State<LogInfo> createState() => _LogInfo();
11 }
12 class _LogInfo extends State<LogInfo>{
13   String textFromFile = '';
14   FirebaseService? _firebaseService;
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: const Text("Smart Garage"
20         ), // Text
21         actions: [
22           ? Padding(
23             padding: const EdgeInsets.only(right: 20),
24             child: GestureDetector(
25               onTap: (){
26                 Navigator.pushNamed(context, 'Home');
27               },
28               child: const Icon(Icons.broadcast_on_personal_rounded),
29             ), // GestureDetector
30           ), // Padding
31           Padding(
32             padding: const EdgeInsets.only(right: 40),
33             child: GestureDetector(
34               onTap:() async {
35                 await _firebaseService?.logout();
36                 Navigator.popAndPushNamed(context, 'Login');
37               },
38               child: const Icon(Icons.logout_rounded),
39             ), // GestureDetector
40           ), // Padding
41         ],

```

Figure 47: Log Info Page 1



```
42     ), // AppBar
43     body: Center(
44       child: Column(
45         mainAxisAlignment: MainAxisAlignment.start,
46         children: [
47           const SizedBox(
48             //Use of SizedBox
49             height: 10,
50           ), // SizedBox
51           ElevatedButton(onPressed: () {
52             _fetchData();
53             getData();
54           }),
55           child: const Text(
56             'Get Data'
57           ), // Text
58         ), // ElevatedButton
59         const SizedBox(
60           //Use of SizedBox
61           height: 5,
62         ), // SizedBox
63         ElevatedButton(onPressed: () {
64           clear();
65         }),
66         child: const Text(
67           'Clear'
68         ), // Text
69       ), // ElevatedButton
70       const SizedBox(
71         //Use of SizedBox
72         height: 30,
73       ), // SizedBox
74       Text(
75         textFromFile,
76       ), // Text
77     ],
78   ) // Column
79 ), // Center
80 ); // Scaffold
81 }
```

Figure 48: Log Info Page 2

```

82  getData() async{
83      final directory = await getExternalStorageDirectory();
84      final file = File('${directory!.path}/test.txt');
85      final response = await file.readAsString();
86      setState(() {
87          textFromFile = response;
88      });
89  }
90  clear(){
91      setState(() {
92          textFromFile = '';
93      });
94  }
95  void _fetchData() async {
96      var url = Uri.parse('http://192.168.1.139:5000/data');
97      var response = await http.get(url);
98      final Map<String, dynamic> data = json.decode(response.body);
99      String display = "        Date & Time        Numberplate        Allowed" ;
100     List<dynamic> DateandTime = data['DateandTime'];
101     int num = DateandTime.length;
102     int i = 0;
103     while (i < num) {
104         String Allowed = data['Allowed'][i];
105         String DateandTime = data['DateandTime'][i];
106         String NumberPlate = data['NumberPlate'][i];
107         display += "\n        $DateandTime        $NumberPlate        $Allowed";
108         i++;
109     }
110     final directory = await getExternalStorageDirectory();
111     File file = File('${directory!.path}/test.txt');
112     file.writeAsStringSync(display);
113 }
114 }

```

Figure 49: Log Info Page 3

## **Chapter 4: Testing and Analysis**

### **4.1.Test Plan**

Unit testing and system testing are used to test the entire application. Each test case and its results are documented and analysed in the next section.

#### **4.1.1. Unit Testing, Test Plan**

Unit testing in software testing is evaluating each individual module or component of an application in isolation to confirm that it works as intended. Because the tester has access to the application's internal workings, this sort of testing is also known as white box testing. By testing one module at a time, the tester can find and isolate any problems or errors in the code, allowing for faster debugging and overall performance improvement.

The documentation for testing will be grouped into system functions. Because the system consists of two parts: the IoT system and the mobile application.

#### **4.1.2 System Testing, Test Plan**

System testing is a type of software testing in which the performance of a freshly generated software product is verified. Because the program is tested without knowledge of its internal workings, it is commonly referred to as black box testing. The testing guarantees that the software operates in accordance with the test criteria, evaluating both its behaviour and functionality.

The test cases will display builds of all applications. This demonstrates the Project's general real-time operation.

## 4.2. Unit Testing

### 4.2.1. Test case1: Authentication

*Table 2: Test 1 Authentication*

Test Objective	To test whether Authentication will be successful for Valid Numberplates
Action Performed	Object Placed within 20 cm from the Ultrasonic sensor and Comparing clicked picture with valid images
Expected Output	Authentication is Successful should be displayed
Actual Output	Authentication is Successful is displayed
Conclusion	Authentication Successful

```

pi@raspberrypi:~/Documents/SmartGarage $ cd /home/pi
/ Documents/SmartGarage ; /usr/bin/env /bin/python /ho
me/pi/.vscode/extensions/ms-python.python-2023.4.1/py
thonFiles/lib/python/debugpy/adapter/../../debugpy/la
uncher 43653 -- /home/pi/Documents/SmartGarage/wtd.py

Trying Authentication ...
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to '640x480'.
Writing JPEG image to 'currentNum/test.jpg'.
Tesseract Open Source OCR Engine v4.1.1 with Leptonic
a
Tesseract Open Source OCR Engine v4.1.1 with Leptonic
a
Authentication Is Successful

```

Figure 50: Authentication Successful

```

wtd.py  app.py  {} dictionary  {} dic.json
wtd.py > ...
73  while Run == True:
74      Distance = Ultrasonic()
75      print("Trying Authentication ...")
76      while Distance <= 20:
77          Click()
78          tempList = OCR(NumPlate)
79          AUTH = tempList[0]
80          NumPlate = tempList[1]
81          if AUTH == True:
82              break
83          else:
84              Distance = Ultrasonic()
85              print("Trying Authentication Again ...")

```

Figure 51: Authentication backend code

```
def OCR(NumPlate):  
    i = 1  
    os.system("tesseract currentNum/test.jpg numTxt/output --dpi 300 --oem 1 --psm 6")  
    currentFile = open("numTxt/output.txt", "r")  
    CurrentNum = currentFile.read()  
    while i <= 5:  
        os.system("tesseract validNum/"+ str(i) + ".jpg numTxt/output"+str(i)+" --dpi 300 --oem 1 --psm 6")  
        existingFile = open("numTxt/output"+str(i)+".txt", "r")  
        ExistingNum = existingFile.read()  
        if ExistingNum == CurrentNum:  
            print("Authentication Is Successful")  
            NumPlate.extend([str(CurrentNum)])  
            return [True, NumPlate]  
        i = i + 1  
    NumPlate.extend([str(CurrentNum)])  
    return [False, NumPlate]
```

Figure 52: OCR backend code

#### 4.2.2. Test case 2: Verification

Table 3: Test 2 Verification

Test Objective	To test whether verification will be successful when heavy weight is detected
Action Performed	Some weight is places in the Loadcell sensor
Expected Output	Verification is Successful should be displayed
Actual Output	Verification is Successful is displayed
Conclusion	Verification Successful

```
Trying Verification
Verification is Successful
Open Shutter
```

Figure 53: Verification Successful

```
weight = Loadcell()
print("Trying Verification")
while weight >= 1:
    VFON = True
    if VFON == True:
        print("Verification is Successful")
        break
    else:
        weight = Loadcell()
        print("Trying Verification Again ...")
```

Figure 54: Verification backend code

### 4.2.3. Test case 3: Clicking Picture of The Numberplate

Table 4: Test 3 Clicking Pictures

Test Objective	To test whether the picture is being clicked.
Action Performed	Click function is called
Expected Output	The picture should be clicked and saved in the currentNum folder with name test.jpg
Actual Output	Picture is saved as test.jpg inside currentNum folder
Conclusion	Clicking Picture Successful

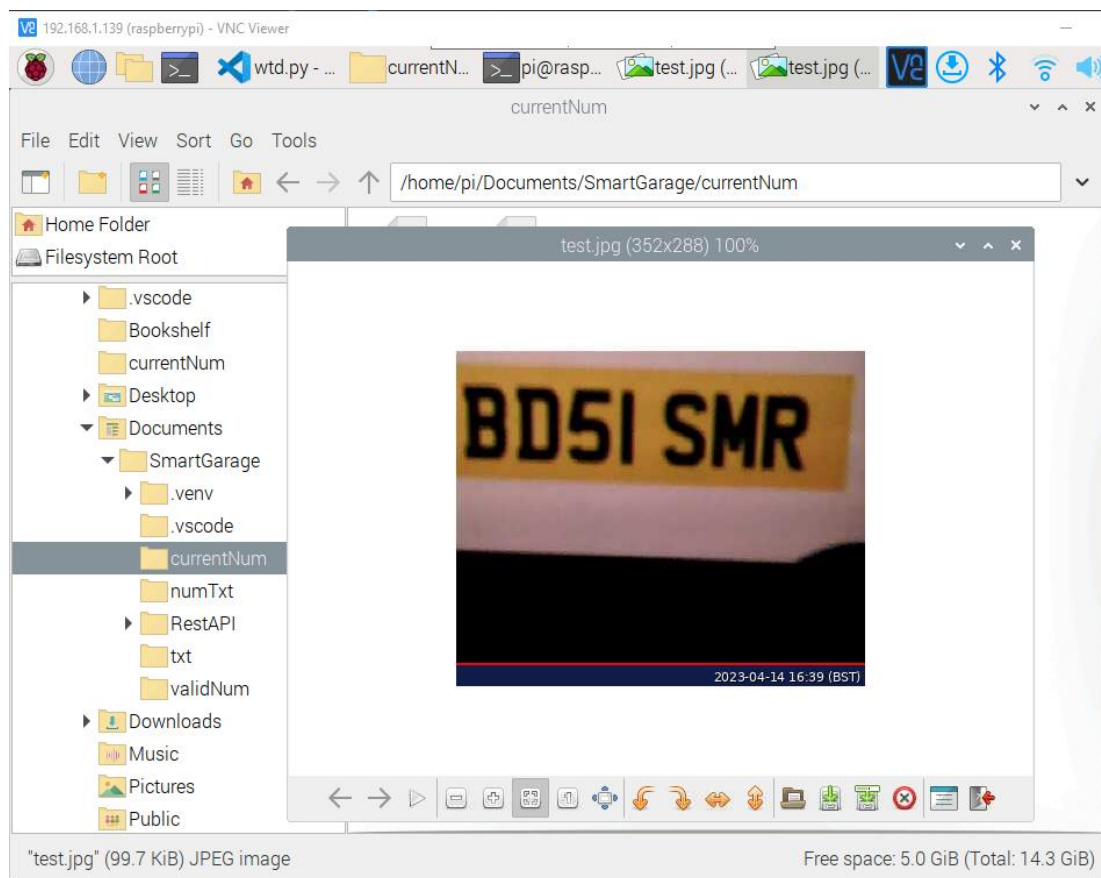


Figure 55: Picture clicked by USB cam



#### 4.2.4. Test case 4: OCR in a Loop

Table 5: Test 4 OCR in Loop

Test Objective	To test whether Tesseract OCR is reading the Numberplates correctly
Action Performed	Invalid Password is entered while logging in.
Expected Output	Error message should be displayed.
Actual Output	Error message saying "Please enter the Correct Password" is displayed.
Conclusion	Admin console Validation Successful.

```

48 while i <= 5:
49     os.system("tesseract validNum/" + str(i) + ".jpg numTxt/output"+str(i)+" --dpi 300 --oem 1 --psm 6"
50     existingFile = open("numTxt/output"+str(i)+".txt", "r")

```

Figure 56: Command for OCR in loop

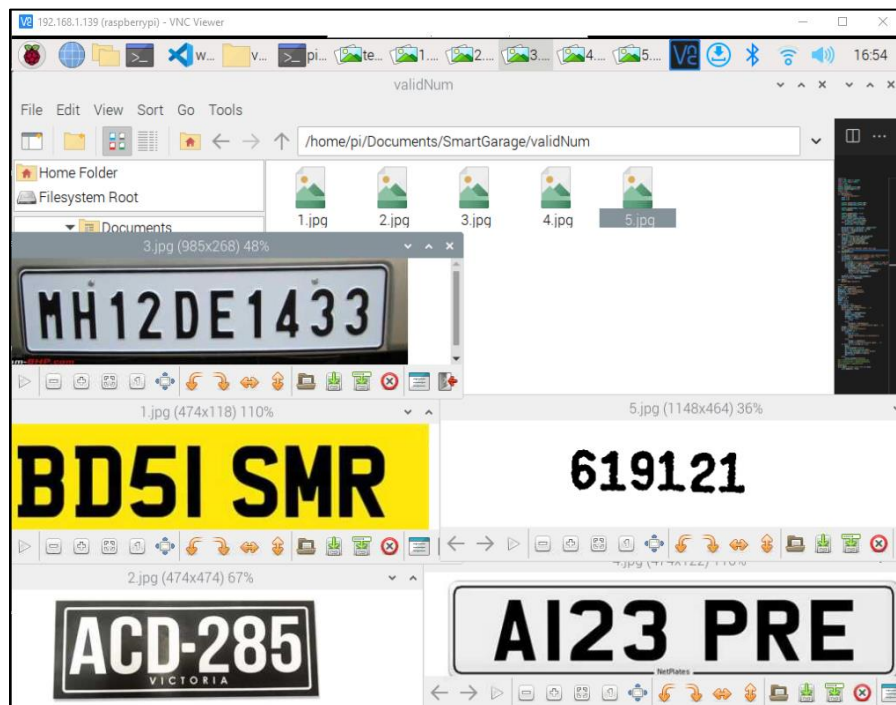


Figure 57: Valid images for OCR

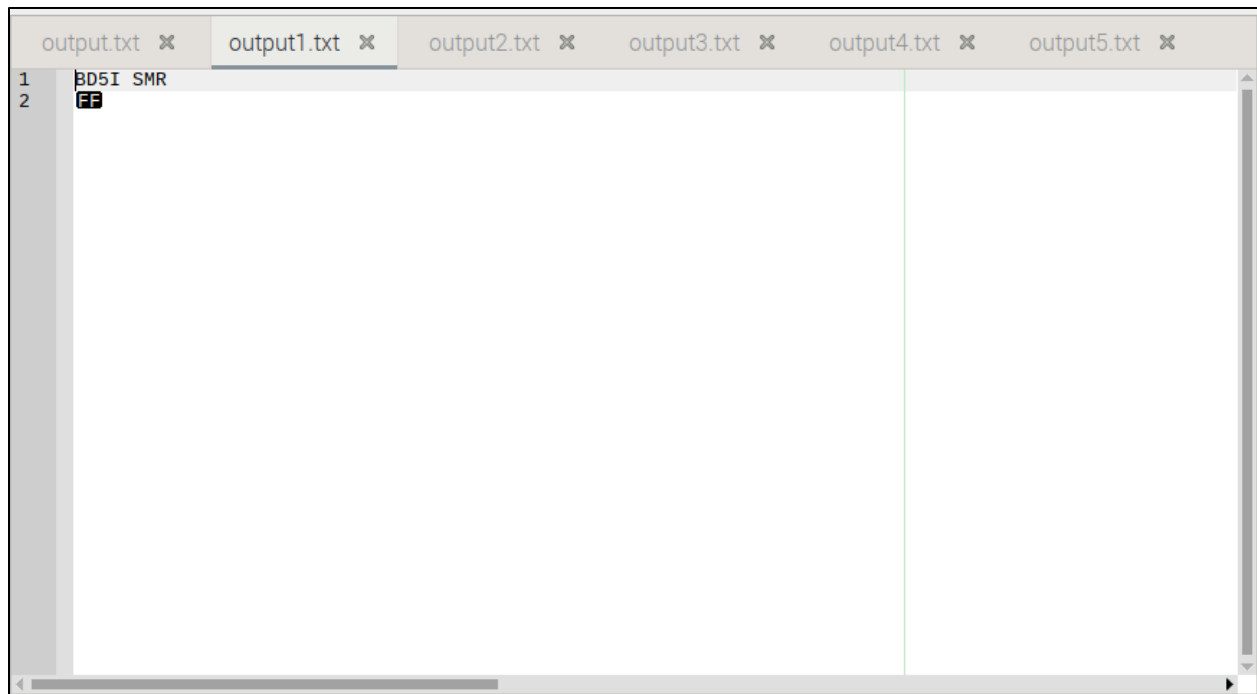


Figure 58: OCR value of 1.jpg

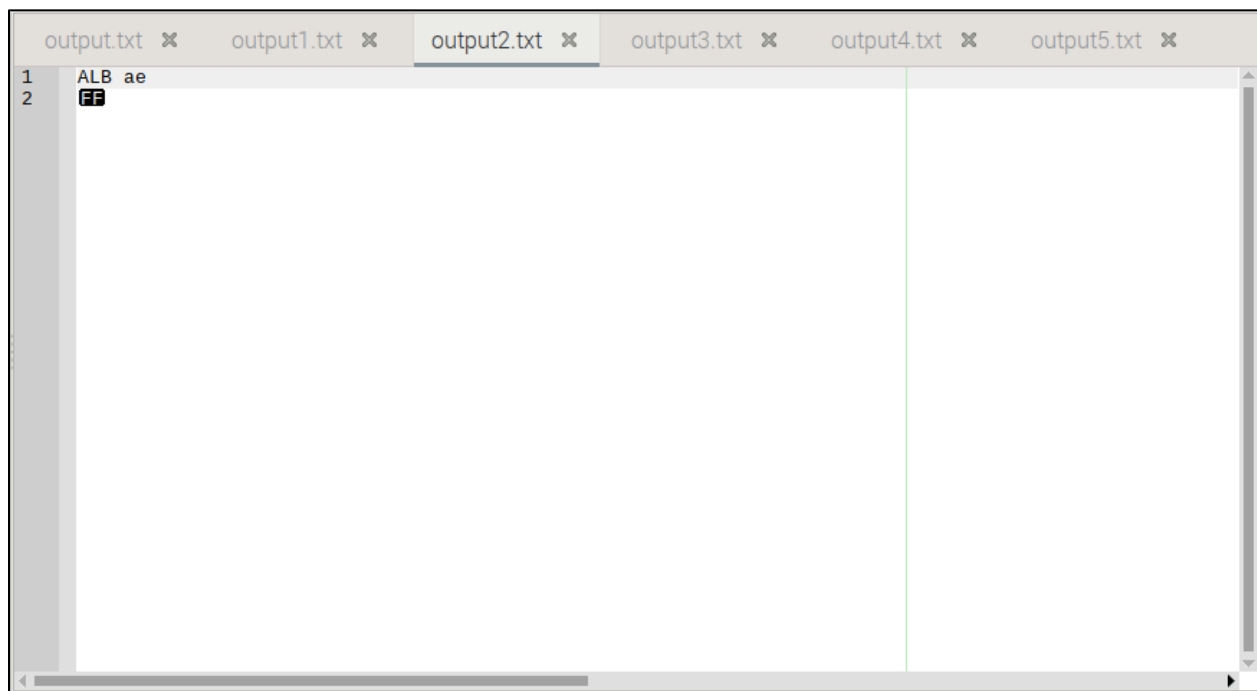


Figure 59: OCR value of 2.jpg

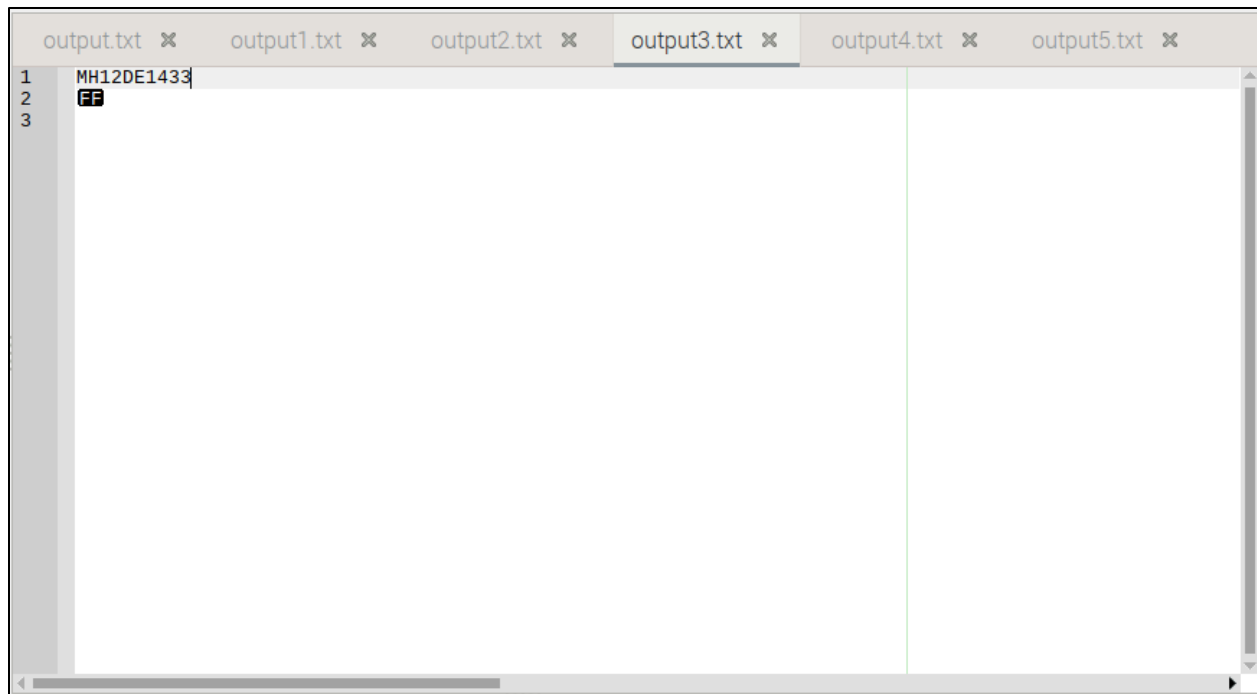


Figure 60: OCR value of 3.jpg

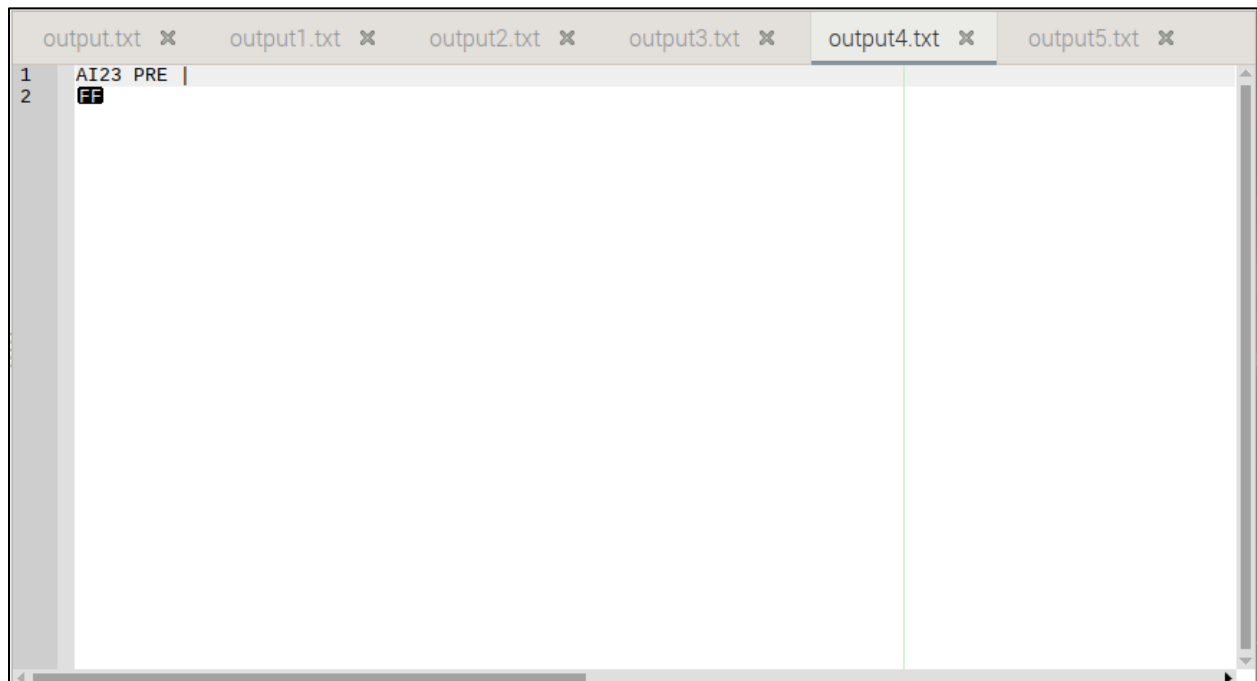


Figure 61: OCR value of 4.jpg

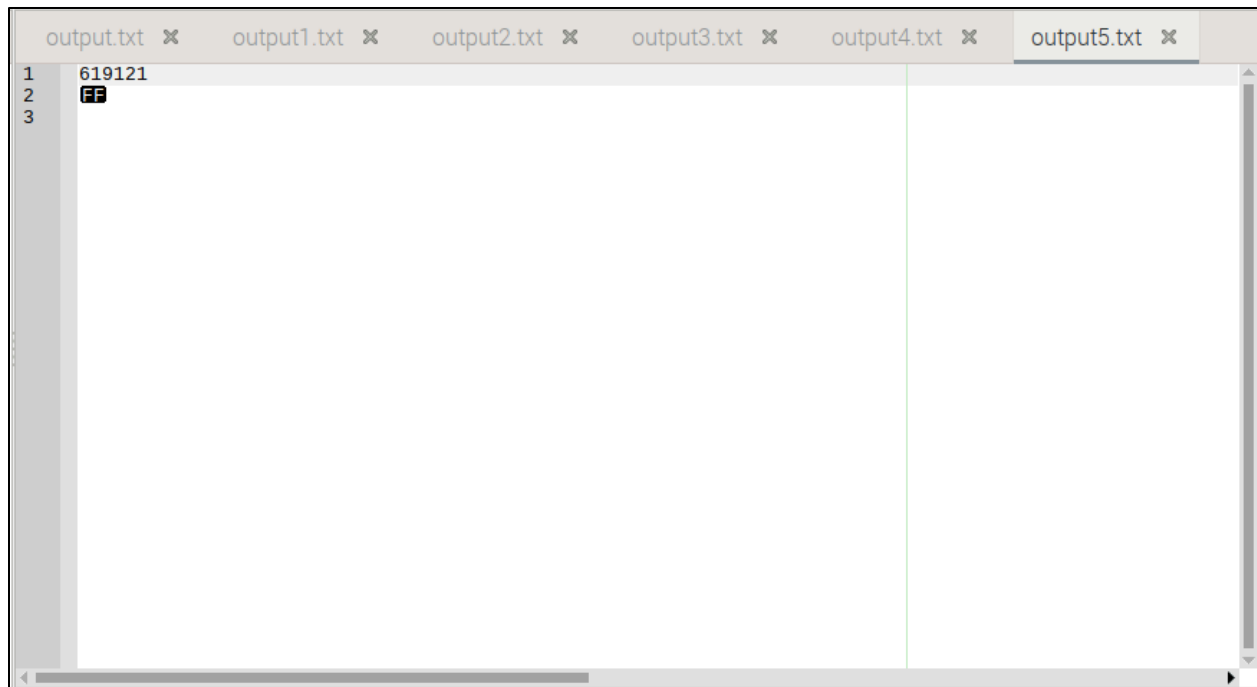


Figure 62: OCR value of 5.jpg

#### 4.2.5. Test case 5: Admin Password Validation

Table 6: Test 5 Admin password Validation

Test Objective	To test whether login will be successful with incorrect password
Action Performed	Invalid Password is entered while logging in.
Expected Output	Error message should be displayed.
Actual Output	Error message saying “Please enter the Correct Password” is displayed.
Conclusion	Admin console Validation Successful.

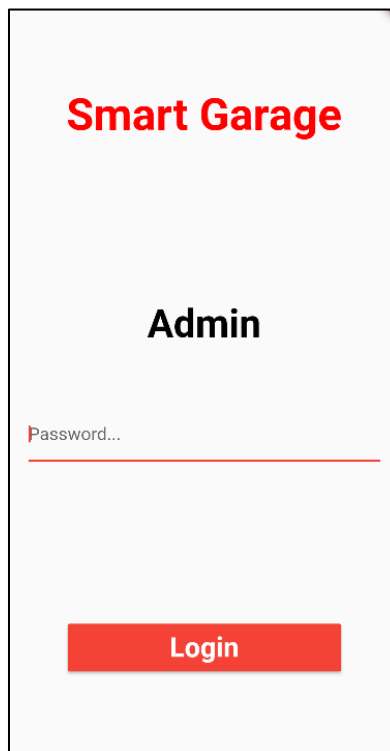
A screenshot of the Admin Login Page. The page has a light gray background. At the top, the text "Smart Garage" is displayed in bold red font. Below it, the word "Admin" is centered in bold black font. Underneath "Admin", there is a label "Password..." in a small gray font, followed by a red horizontal line representing the password input field. At the bottom of the page, there is a red rectangular button with the word "Login" in white text.

Figure 63: Admin Login Page

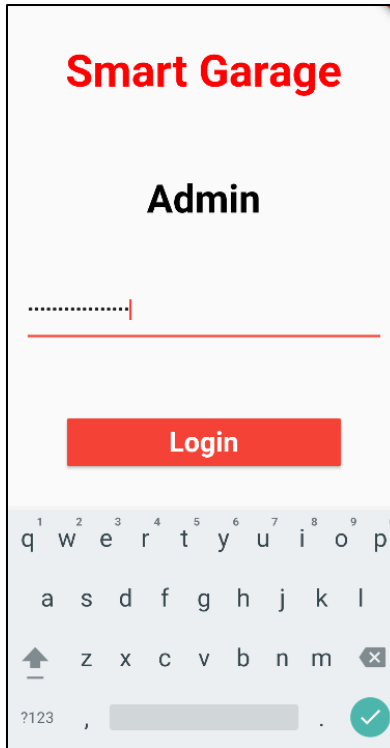


Figure 64: Entering incorrect password

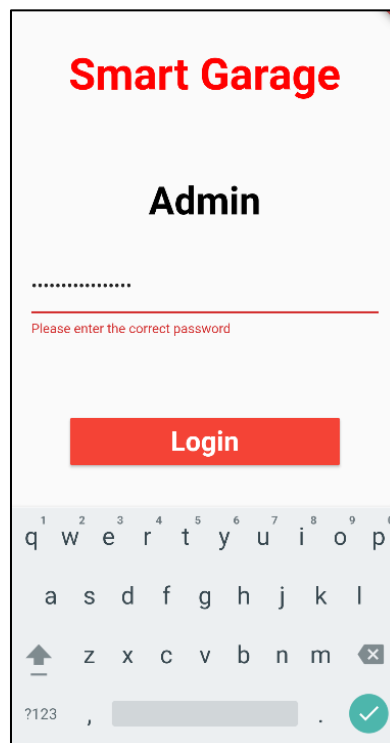


Figure 65: Password validation

#### 4.2.6. Test case 6: Admin Console Login

Table 7: Test 6 Admin Console Login

Test Objective	To test whether login will be successful with correct password.
Action Performed	Valid Password is entered while logging in
Expected Output	Login to home page
Actual Output	Login to Home page is successful
Conclusion	Login successful

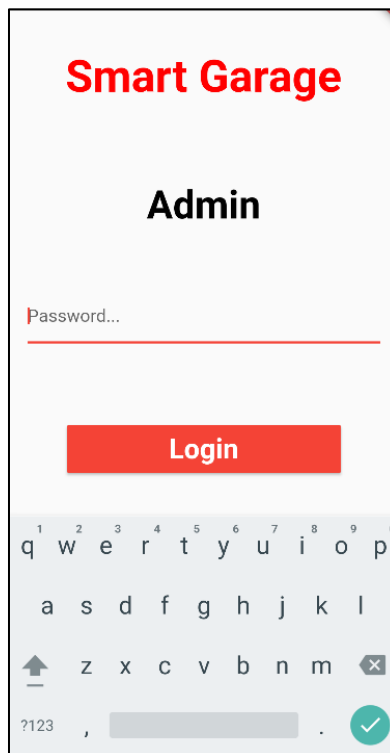


Figure 66: Admin Login page

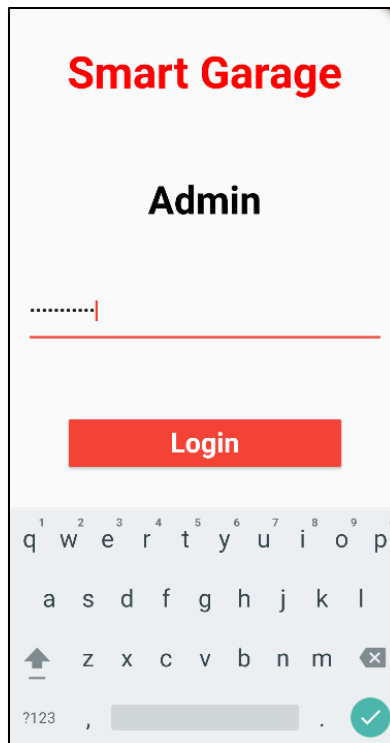


Figure 67: Entering Correct password

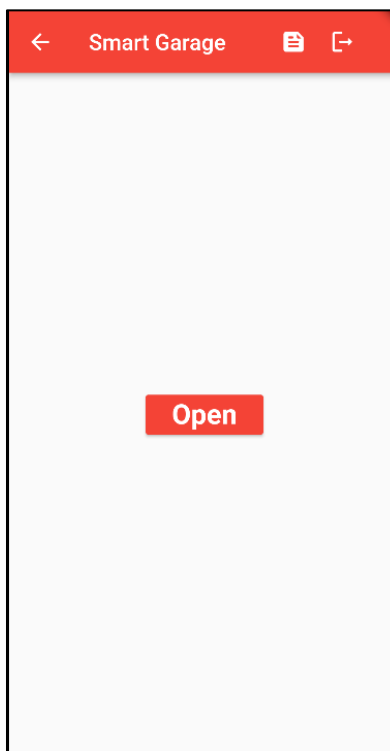


Figure 68: Home page is displayed



#### 4.2.7. Test case 7: Open Button

Table 8: Test 7 Open Button

Test Objective	To test whether post request is sent to open the shutter
Action Performed	Open Button in home page is clicked
Expected Output	Shutter Open message should be displayed in the backend and successful post request message should be displayed in frontend debug console.
Actual Output	Shutter Open message and successful post request message was displayed
Conclusion	Post request from Frontend to backend Successful.

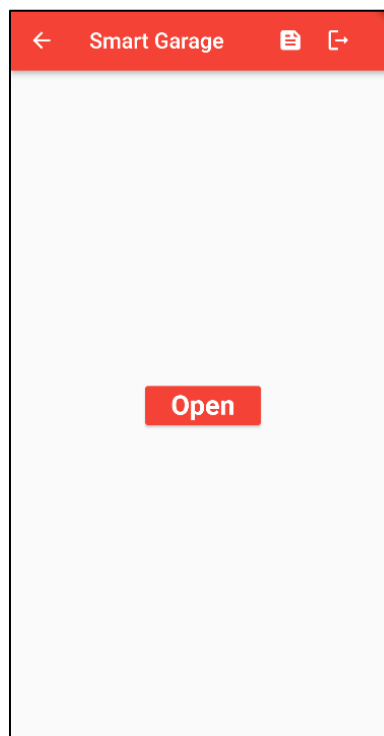
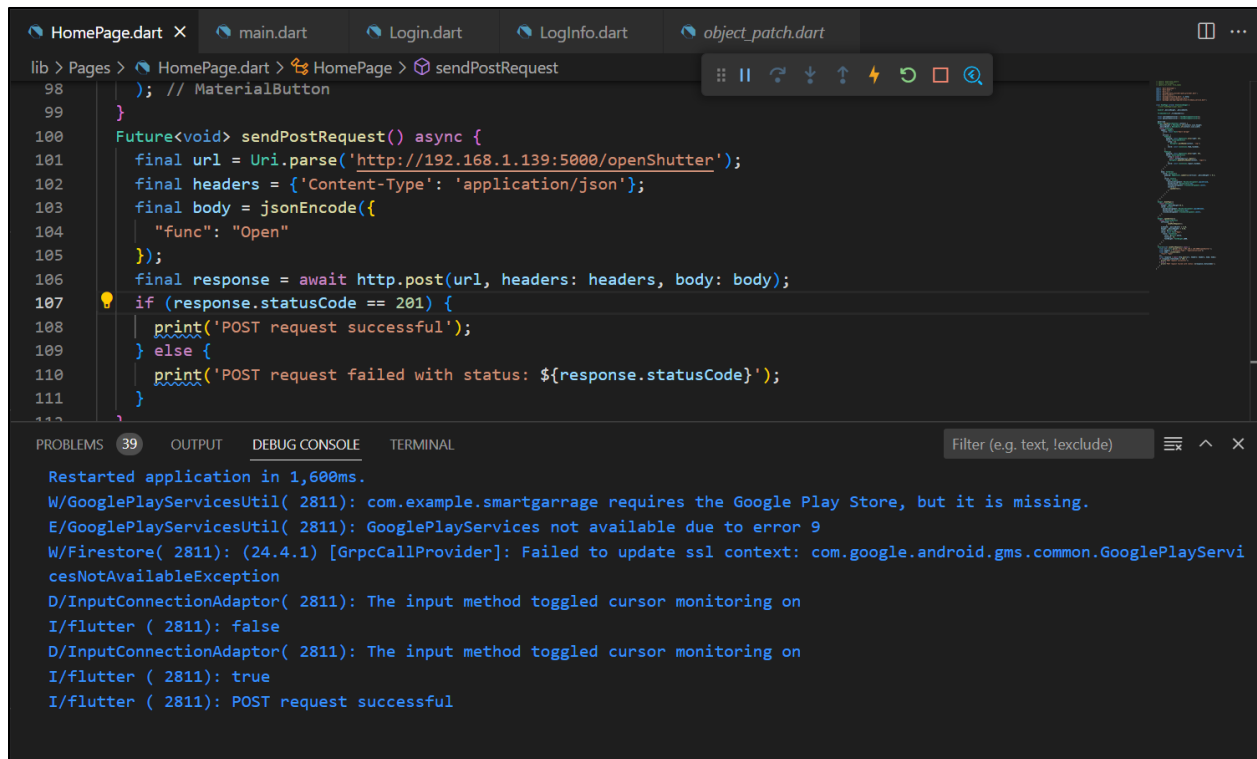


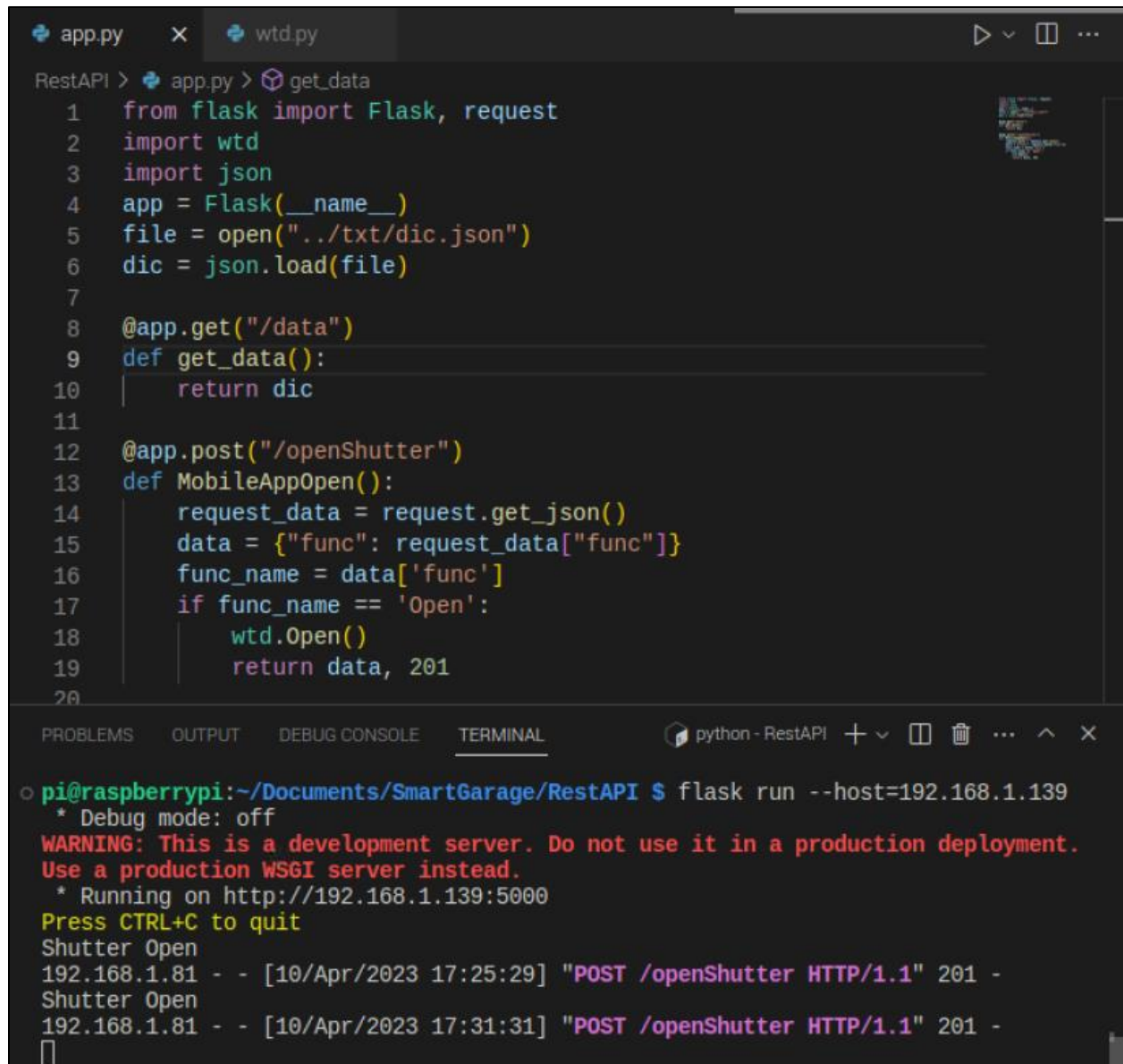
Figure 69: Open Button in Home page



The screenshot shows an IDE with a Dart file named `HomePage.dart` open. The code defines a `sendPostRequest` function that sends a POST request to `http://192.168.1.139:5000/openShutter` with a JSON body `{ "func": "Open" }`. The response status code is checked, and a success message is printed. The bottom panel shows the `DEBUG CONSOLE` with the following output:

```
Restarted application in 1,600ms.
W/GooglePlayServicesUtil( 2811): com.example.smartgarrage requires the Google Play Store, but it is missing.
E/GooglePlayServicesUtil( 2811): GooglePlayServices not available due to error 9
W/Firestore( 2811): (24.4.1) [GrpcCallProvider]: Failed to update ssl context: com.google.android.gms.common.GooglePlayServicesNotAvailableException
D/InputConnectionAdaptor( 2811): The input method toggled cursor monitoring on
I/flutter ( 2811): false
D/InputConnectionAdaptor( 2811): The input method toggled cursor monitoring on
I/flutter ( 2811): true
I/flutter ( 2811): POST request successful
```

Figure 70: Frontend post request successful



The image shows a code editor with two files: `app.py` and `wtd.py`. The `app.py` file contains the following Python code:

```
RestAPI > app.py > get_data
1 from flask import Flask, request
2 import wtd
3 import json
4 app = Flask(__name__)
5 file = open("../txt/dic.json")
6 dic = json.load(file)
7
8 @app.get("/data")
9 def get_data():
10     return dic
11
12 @app.post("/openShutter")
13 def MobileAppOpen():
14     request_data = request.get_json()
15     data = {"func": request_data["func"]}
16     func_name = data['func']
17     if func_name == 'Open':
18         wtd.Open()
19     return data, 201
20
```

Below the code editor is a terminal window showing the output of the `flask run` command. The terminal output is as follows:

```
pi@raspberrypi:~/Documents/SmartGarage/RestAPI $ flask run --host=192.168.1.139
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://192.168.1.139:5000
Press CTRL+C to quit
Shutter Open
192.168.1.81 - - [10/Apr/2023 17:25:29] "POST /openShutter HTTP/1.1" 201 -
Shutter Open
192.168.1.81 - - [10/Apr/2023 17:31:31] "POST /openShutter HTTP/1.1" 201 -
```

Figure 71: Backend Post request successful

#### 4.2.8. Test case 8: Navigation Button

Table 9: Test 8 Navigation Button

Test Objective	To test whether navigation to another page will be successful
Action Performed	Clicking on Log button in the navigation bar
Expected Output	Navigating to Log Page
Actual Output	Home page to Log page Navigation
Conclusion	Navigation was successful

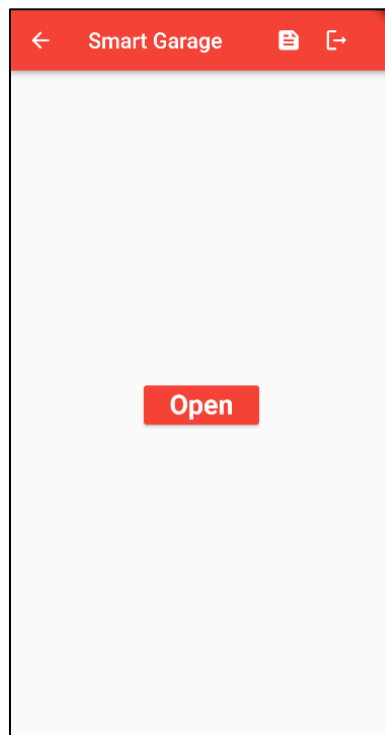
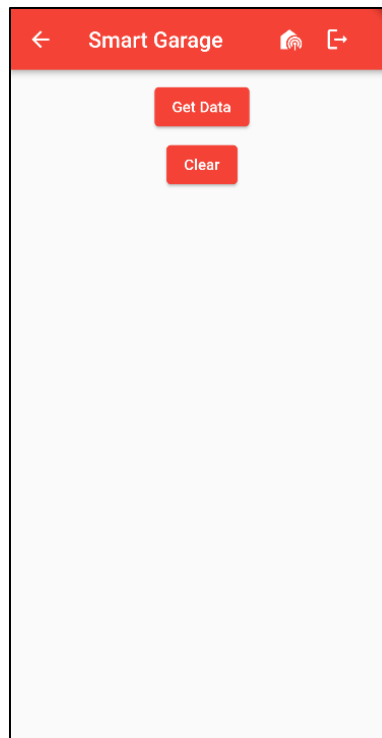


Figure 72: Navigation button in App bar of the Home Page



*Figure 73: Log Info Page displayed*

#### 4.2.9. Test case 9: Get Data Button

Table 10: Test 9 Get Data Button

Test Objective	To test whether the data is coming from backend and is displayed in the mobile application
Action Performed	Get Data Button is Clicked
Expected Output	Contents of json file in Raspberry Pi should be displayed and GET request should be successful.
Actual Output	Contents of json file is displayed in the mobile app and successful GET request message is displayed in Backend
Conclusion	Get request Successful and Contents are displayed

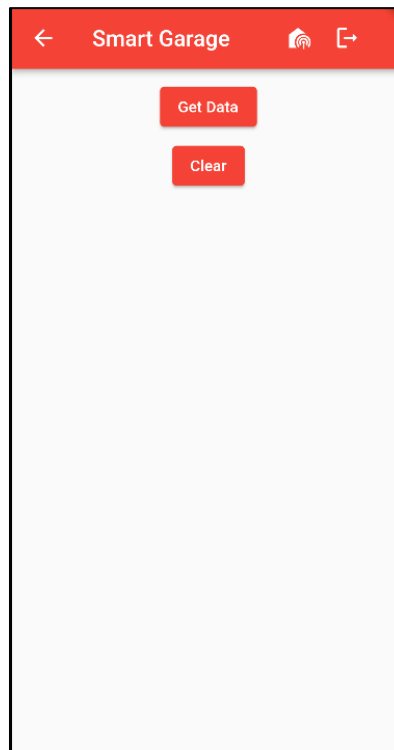


Figure 74: Get button In Log Info Page

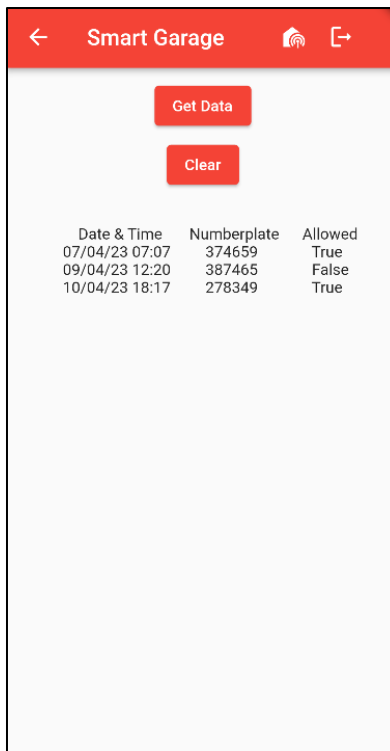


Figure 75: Data from Json file stored in Backend in displayed

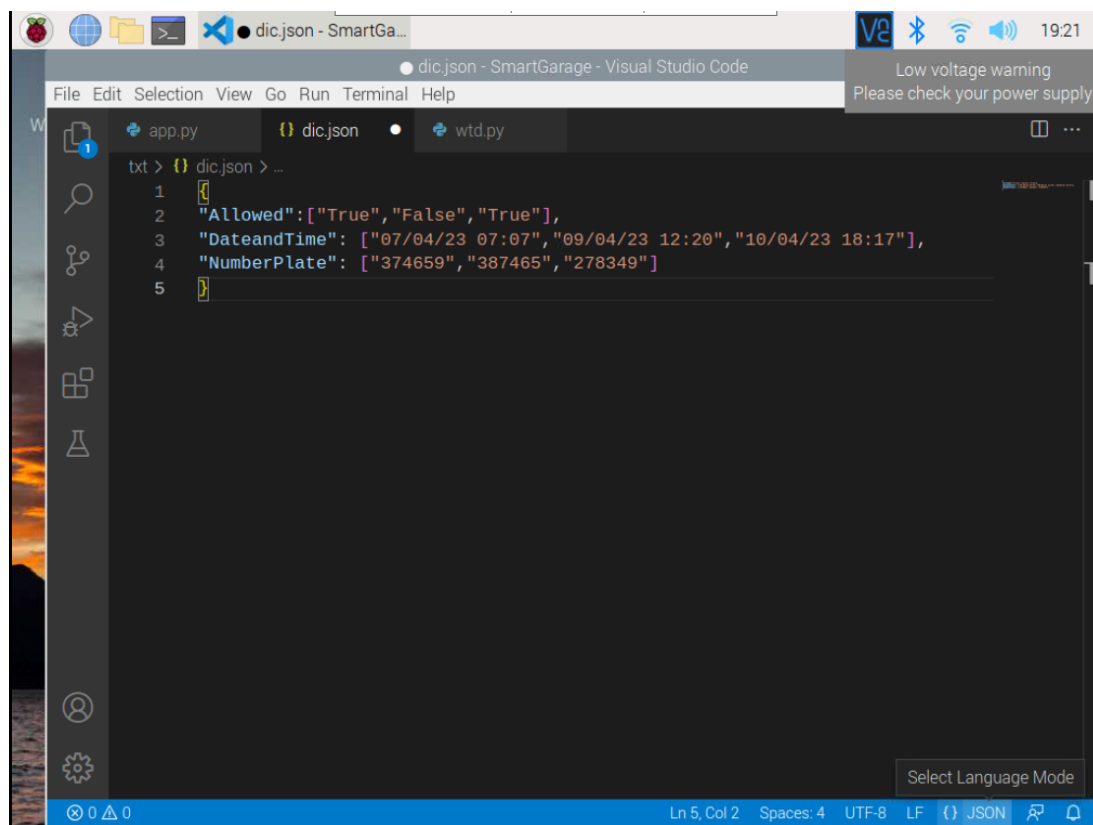
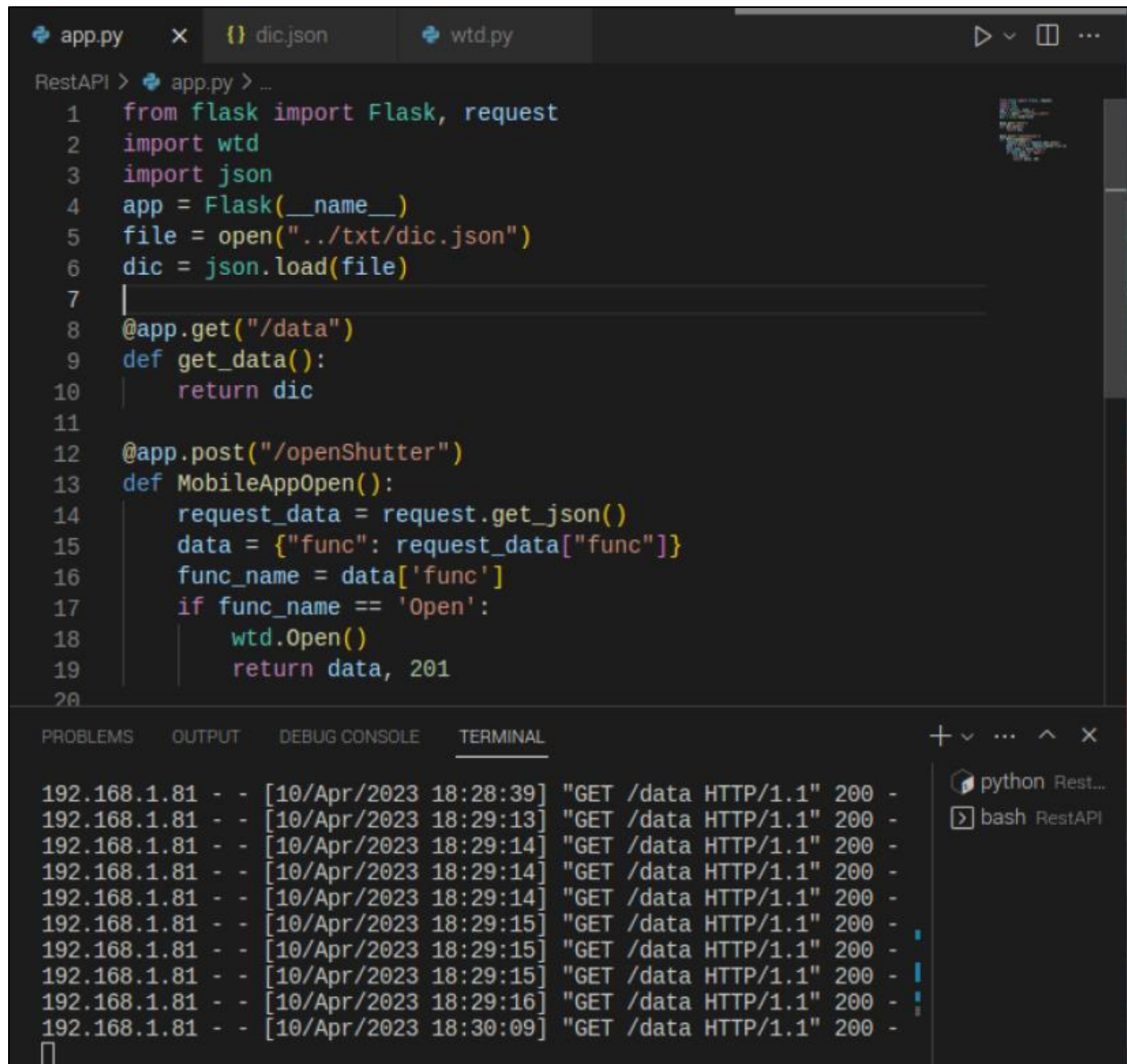


Figure 76: Json file in the backend



The image shows a VS Code editor with three tabs: `app.py`, `dic.json`, and `wtd.py`. The `app.py` tab is active, displaying the following Python code:

```
RestAPI > app.py > ...
1  from flask import Flask, request
2  import wtd
3  import json
4  app = Flask(__name__)
5  file = open("../txt/dic.json")
6  dic = json.load(file)
7
8  @app.get("/data")
9  def get_data():
10     return dic
11
12  @app.post("/openShutter")
13  def MobileAppOpen():
14     request_data = request.get_json()
15     data = {"func": request_data["func"]}
16     func_name = data['func']
17     if func_name == 'Open':
18         wtd.Open()
19     return data, 201
20
```

Below the code editor, the `TERMINAL` tab is active, showing the output of the REST client:

```
192.168.1.81 - - [10/Apr/2023 18:28:39] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:13] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:14] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:14] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:14] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:15] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:15] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:15] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:29:16] "GET /data HTTP/1.1" 200 -
192.168.1.81 - - [10/Apr/2023 18:30:09] "GET /data HTTP/1.1" 200 -
```

On the right side of the terminal, there are two tabs: `python Rest...` and `bash RestAPI`. The `python RestAPI` tab is selected, showing the REST client interface.

Figure 77: Get Request successful



#### 4.2.10. Test case 10: Clear Button

Table 11: Test 10 Clear Button

Test Objective	To test whether Clear button will remove the displayed data
Action Performed	Clear Button is clicked
Expected Output	The Displayed text should be cleared
Actual Output	The Displayed text from Get Data is cleared
Conclusion	Clearing Data displaying Successful

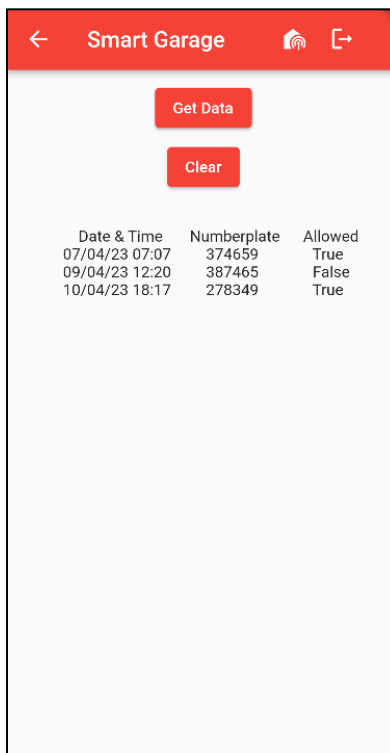
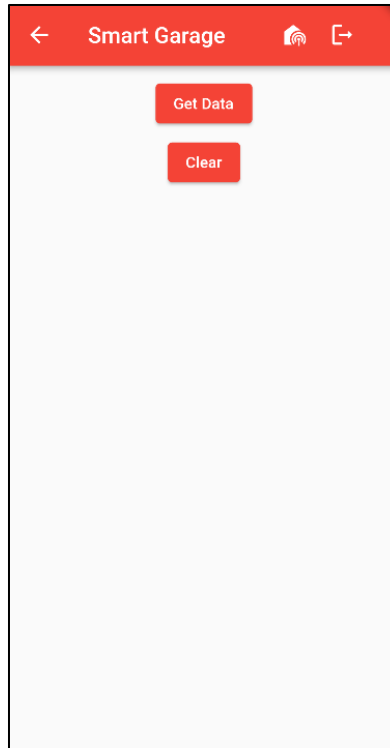


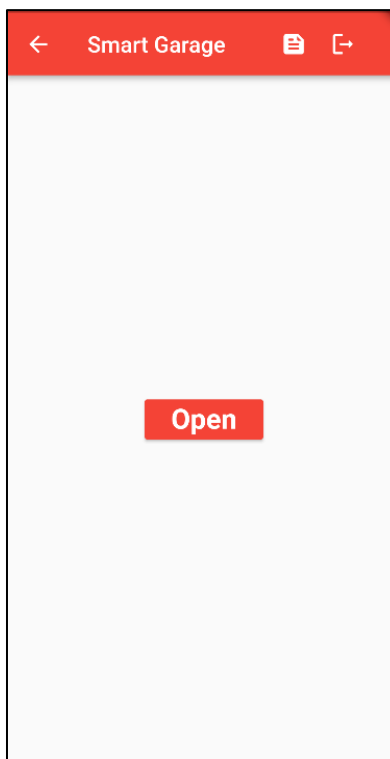
Figure 78: Data displayed in Log info page

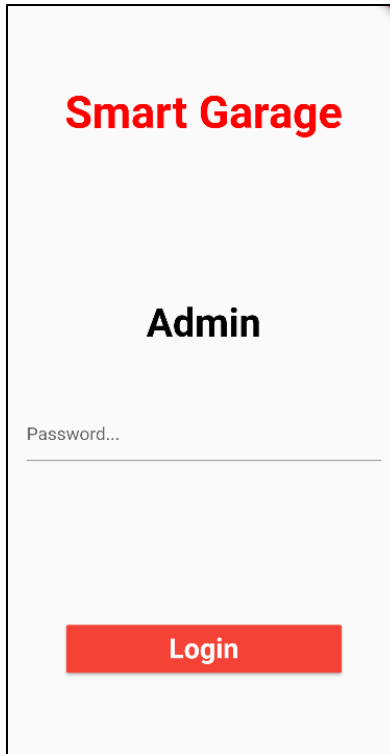


*Figure 79: Clear button clearing data*

**4.2.11. Test case 11: Logout Button***Table 12: Test 11 Logout Button*

Test Objective	To test whether the login page is displayed with password field being empty after clicking the logout button
Action Performed	Logout Button is Clicked
Expected Output	Should get back to the Login page with nothing in the text field of password
Actual Output	Logout from home or log page back to Login Page with the password removed
Conclusion	Logout successful

*Figure 80: Logout Button in the app bar*



The image shows a mobile application interface for 'Smart Garage'. At the top, the text 'Smart Garage' is displayed in a bold red font. Below this, the word 'Admin' is centered in a bold black font. Underneath 'Admin', there is a text input field with the placeholder text 'Password...'. At the bottom of the screen, there is a red rectangular button with the word 'Login' in white text.

*Figure 81: Logged out of the admin console*

### 4.3. System Testing

#### 4.3.1. Test case 12: Verification and Authentication Shutter Opening

Table 13: Test 12 Verification and Authentication Shutter Opening

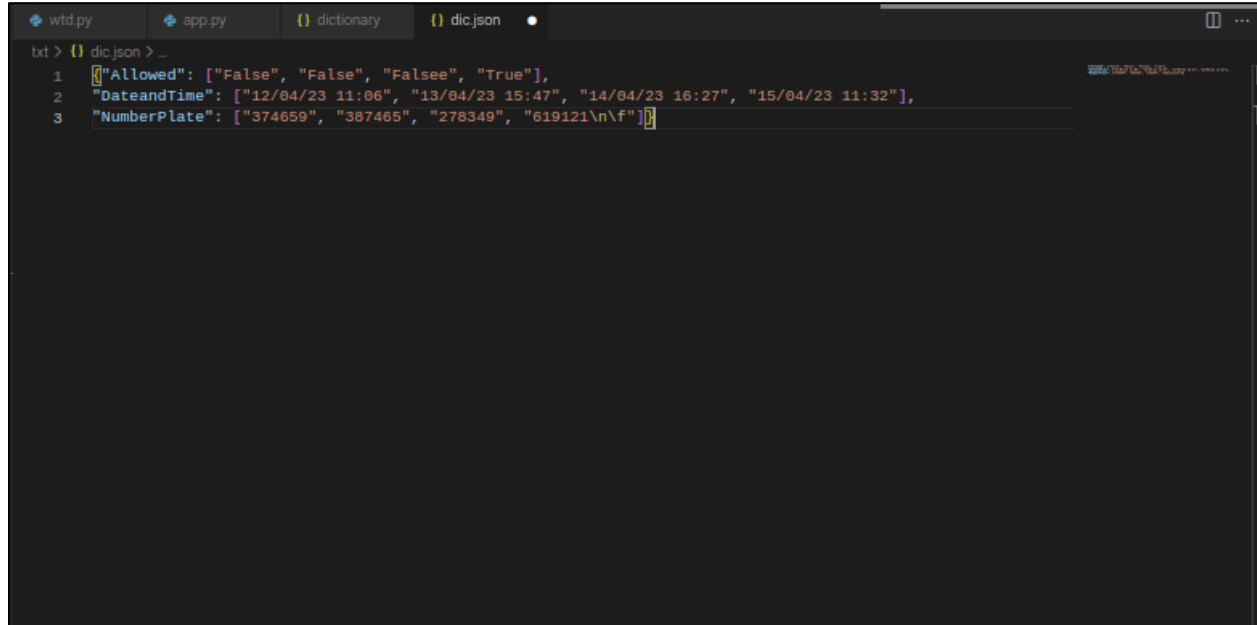
Test Objective	To test whether the shutter will open automatically after authentication and verification
Action Performed	Placing an object within 20 cm of the ultrasonic sensor, Clicking Picture of Numberplate and pass positive weight in loadcell sensor.
Expected Output	Open Shutter should be displayed and Json file should be updated.
Actual Output	Open Shutter is displayed and Json file is updated.
Conclusion	Verification and Authentication shutter opening with log updating Successful

```

pi@raspberrypi:~/Documents/SmartGarage $ /usr/bin/env /bin/python /home/pi/.vscode/extensions/ms-python.python-2023.4.1/pythonFiles/lib/python/debugpy/adapter/../../../../debugpy/launcher 36037 -- /home/pi/Documents/SmartGarage/wtd.py
Trying Authentication ...
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to '640x480'.
Writing JPEG image to 'currentNum/test.jpg'.
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Authentication Is Successful
Trying Verification
Verification is Successful
Open Shutter
pi@raspberrypi:~/Documents/SmartGarage $

```

Figure 82: Running the backend code



```
txt > {} dic.json > _
1  {"Allowed": ["False", "False", "False", "True"],
2  "DateandTime": ["12/04/23 11:06", "13/04/23 15:47", "14/04/23 16:27", "15/04/23 11:32"],
3  "NumberPlate": ["374659", "387465", "278349", "619121\n\f"]}

```

Figure 83: Dictionary file updated

**4.3.2. Test case 13: Mobile App***Table 14: Test 13 Mobile App*

Test Objective	To test whether the Mobile app is working as needed
Action Performed	Login into Mobile app, clicking on Open, navigating to a different page, Clicking on Get Data and Clear the Displayed text
Expected Output	<ul style="list-style-type: none"> <li>• Login success</li> <li>• Pose request success</li> <li>• Shutter Open should Display</li> <li>• Get Request Successful</li> <li>• Log should Display</li> <li>• Displayed log should be cleared</li> </ul>
Actual Output	<ul style="list-style-type: none"> <li>• Login successful</li> <li>• Pose request Successful</li> <li>• Shutter Open displayed</li> <li>• Get Request Successful</li> <li>• Log Displaying</li> <li>• Clearing displayed log</li> </ul>
Conclusion	Mobile testing successful

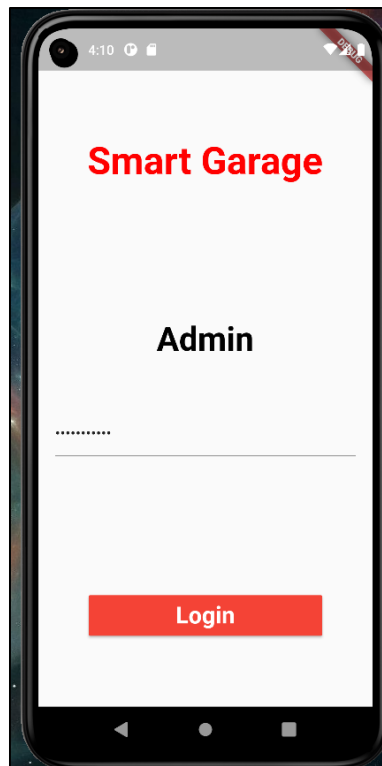


Figure 84: Entering password in the admin login

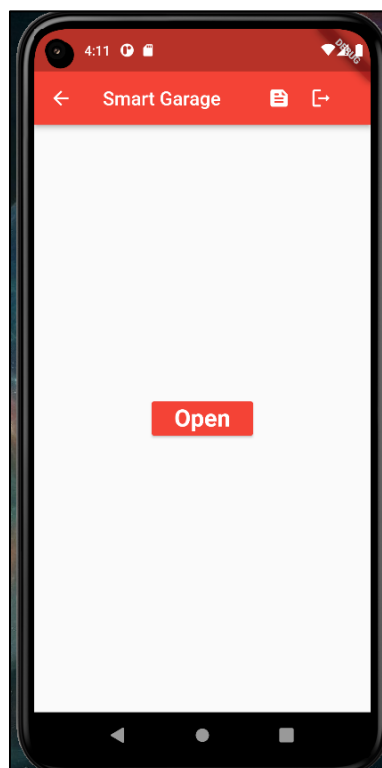


Figure 85: Home displayed



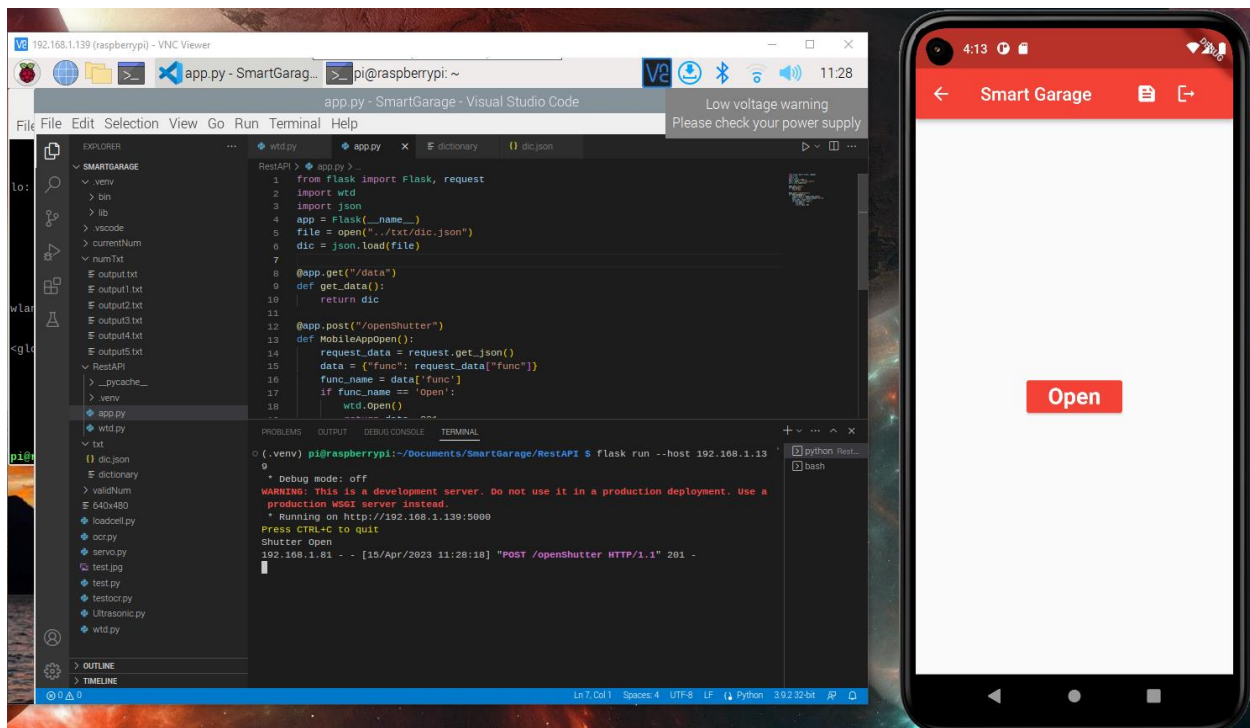


Figure 86: Post request to open Shutter

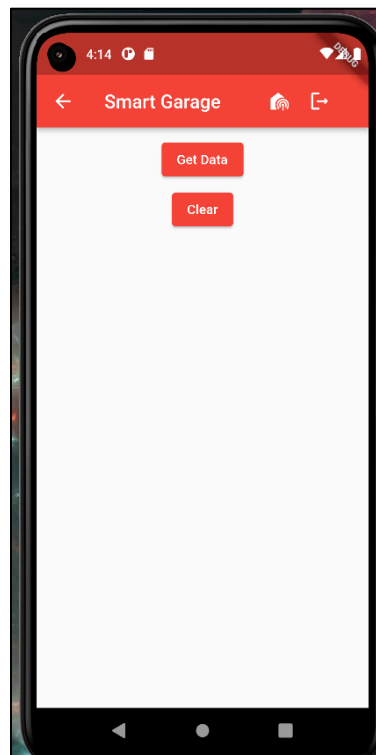


Figure 87: Navigating to Log info page

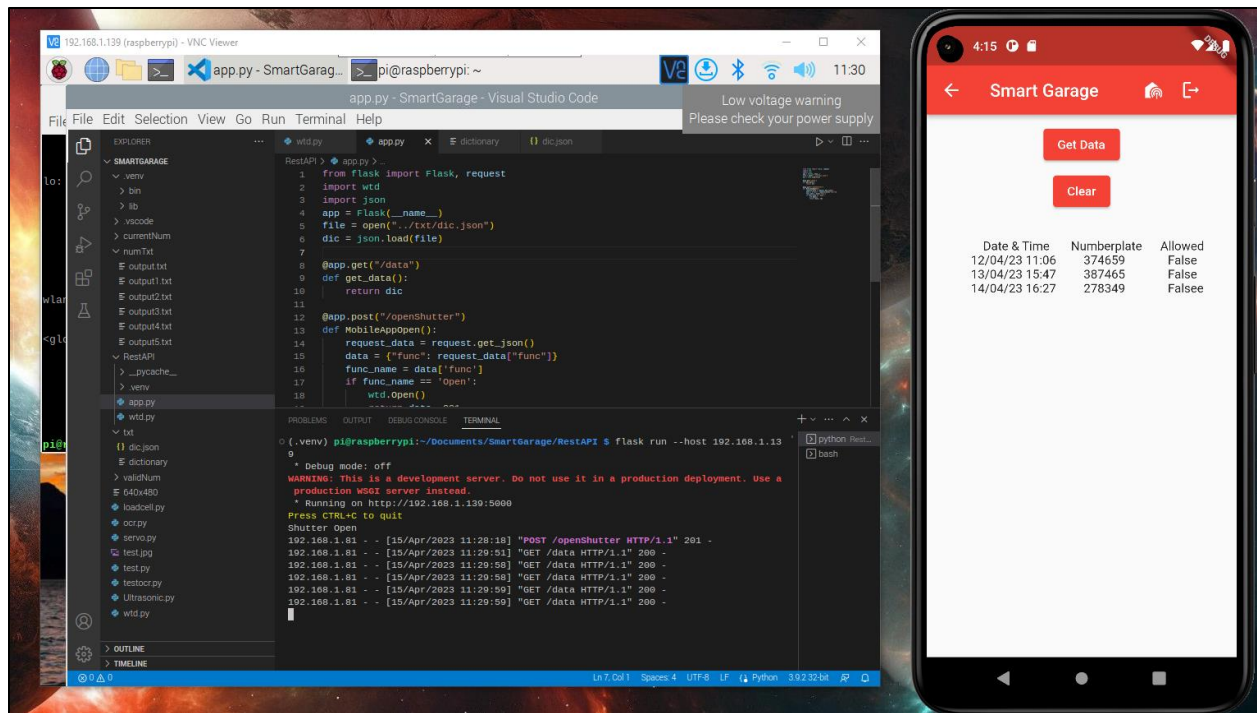


Figure 88: Get request to display contents of backend

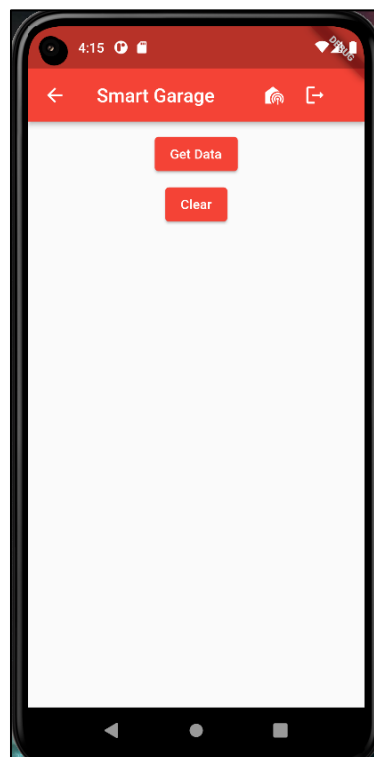
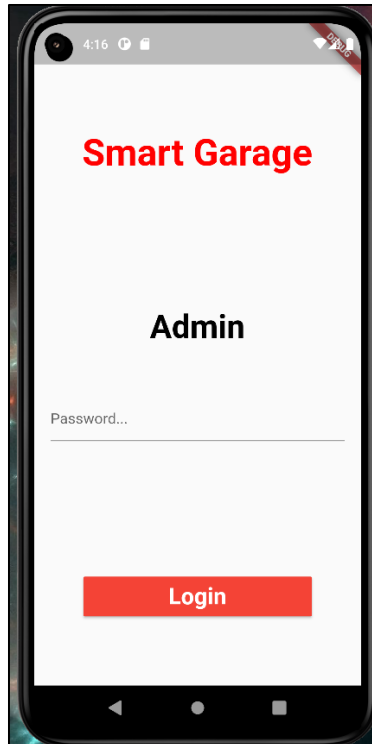


Figure 89: Clear button



*Figure 90: Logged out*

#### 4.4. Critical Analysis

One significant benefit of this technology is that it improves security by preventing unwanted access to the shutter. This is accomplished by demanding authentication and verification using sensors, making it difficult for anyone who do not have permission to open the shutter. This is especially handy when standard keys are insufficiently secure or when numerous people need access to the shutter.

One of the benefits of this is convenience. Instead of traveling to the shutter and manually unlocking it, users can open it remotely using a smartphone application, saving time and effort.

This IoT project has a number of serious concerns that must be addressed. The potential of hacking and data breaches is the most serious threat. Because the mobile application is merely secured with a password, hackers who obtain access to it can swiftly infiltrate the system. Once they obtain access to the admin panel, they may engage in illicit activities such as data theft by exploiting the shutter.

In addition, the initiative raises certain concerns about potential technological challenges and malfunctions. If the IoT system or mobile application has technical difficulties, users may be unable to open the shutter when required. This could be especially problematic in emergency scenarios if quick access to the shutter is required. Furthermore, worries about data security and privacy may develop. The mobile application may capture and retain users' personal information, such as their license plate, posing a risk of misuse or theft.

In general, adopting an IoT project that allows users to open shutters via a mobile application after proper authentication and verification delivers major security and convenience benefits. However, it also raises concerns about privacy, security, and technical issues that must be addressed.

#### 4.4.1. Analysing Failed Test Cases

This part documents the failed test cases and ways to mitigate this type of error.

##### 4.4.1.1. Camera not clicking the Numberplate accurately

For this test, the error of camera not clicking the numberplate is shown. When the Vehicle is not aligned with the camera perfectly, the picture clicked won't have the full numberplate. This leads to authenticated Vehicles fail in the authentication phase.

```
40 def Click():
41     os.system('fswebcam 640x480 -pYUYV currentNum/test.jpg')
42 def OCR(NumPlate):
43     i = 1
44     os.system("tesseract currentNum/test.jpg numTxt/output --dpi 300 --oem 1 --psm 6")
45     currentFile = open("numTxt/output.txt", "r")
46     CurrentNum = currentFile.read()
47     while i <= 5:
48         os.system("tesseract validNum/" + str(i) + ".jpg numTxt/output" + str(i) + " --dpi 300 --oem 1 --psm 6")
49         existingFile = open("numTxt/output" + str(i) + ".txt", "r")
50         ExistingNum = existingFile.read()
51         if ExistingNum == CurrentNum:
52             print("Authentication Is Successful")
53             NumPlate.extend([str(CurrentNum)])
54             return [True, NumPlate]
55         i = i + 1
56     NumPlate.extend([str(CurrentNum)])
57     return [False, NumPlate]
```

Figure 91: Code for OCR

The code shown above clicks picture in line 41 and OCR for the clicked image in done in line 44. Similarly, OCR of valid Numberplates in done in line 48.



Figure 92: Inaccurate Numberplate picture clicking

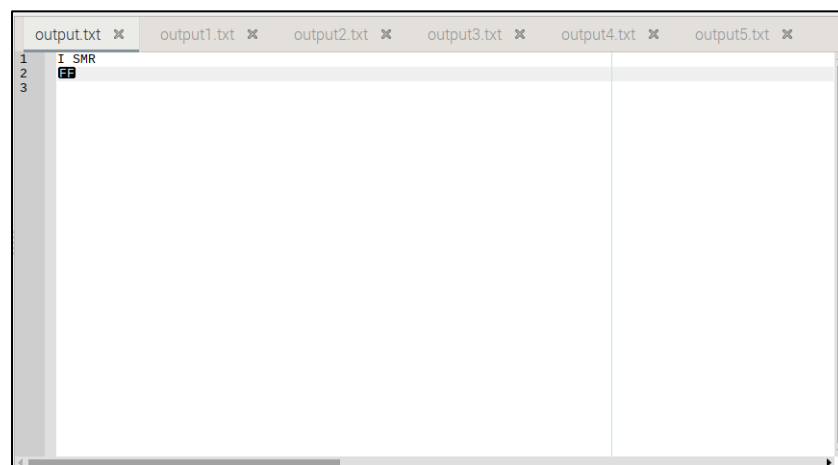


Figure 93: Output according to the Image

According to the Image clicked, The OCR has detected the text of the image provided.

```

pi@raspberrypi:~/Documents/SmartGarage $ /usr/bin/env /bin/python /home/pi/.vscode/extensions/ms-python.python-2023.4.1/pythonFiles/
lib/python/debugpy/adapter/../../debugpy/launcher 38867 -- /home/pi/Documents/SmartGarage/wtd.py
Trying Authentication ...
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to '640x480'.
Writing JPEG image to 'currentNum/test1.jpg'.
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Trying Authentication Again ...
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to '640x480'.
Writing JPEG image to 'currentNum/test1.jpg'.
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Trying Authentication Again ...

```

Figure 94: Authentication not being successful

In the above image, the process of authentication is carried out in a loop because the image clicked doesn't return the valid Numberplate although the Vehicle's full number plate is Valid.

To avoid this from happening, marking on the ground where the vehicle's number plate can fully be seen should be implemented for accuracy of this project.

#### 4.4.2. Test Summary

Several critical components are typically included in the test report of an IoT project that allows customers to open a shutter following authentication and verification via a mobile application. These components are examined to guarantee that the IoT system and mobile application's primary functionalities function properly and generate the intended outcomes.

##### 4.4.2.1 Strong Aspects of the System

- Shutter will open only after Authentication and Verification
- Admin has access over Admin Console
- Log of every attempt is stored and can be viewed in Admin console
- Visitors and Guests might not be valid users so, Admin can open the shutter for special cases
- Part of a Smart Home

##### 4.4.2.2 Weak Aspects of the System

- Threat to Sensors in rainy weather
- Dust and dirt in camera can lead to valid numberplate being rejected
- Security while logging in Admin Console

As seen by all of the test cases, the overall system was a success, and the system is now ready for usage. Future development, on the other hand, will improve the system's performance.



## Chapter 5: Conclusion

In conclusion, incorporating a verification and authentication procedure into a smart garage is critical for maintaining security and ensuring only approved vehicles are permitted within. This process employs cutting-edge technology such as sensors and cameras to detect the presence of a vehicle and identify it based on its license plate. The garage door opens to let the vehicle in once it has been authenticated.

The verification and authentication procedure in a smart garage not only improves security but also gives useful information that may be shown in an administrative interface application. This information comprises the vehicle's license plate number, the date and time of entry, and the frequency of use. This data can be used by the garage owner to track usage patterns, identify potential security issues, and improve overall garage operations.

Furthermore, the admin console program enables authorized users to operate the garage door without going through the regular verification and authentication steps. When the garage owner needs to manually operate the door, such as during maintenance or repair work, this capability comes in handy.

To summarize, the inclusion of a verification and authentication procedure in a smart garage, as well as the availability of an admin console application, provides an efficient and secure approach to manage garage access. This system improves security and delivers crucial data to garage owners, while also providing convenience features like privileged access to the garage door.

## **5.1 Legal, Social and Ethical Issues**

### **5.1.1. Legal Issues**

- a) **Liability:** Liability arises when smart garages fail or are not used properly, potentially causing property damage or personal injury. This raises the issue of who is responsible for any resulting harm.
- b) **Cybersecurity:** Smart garages are vulnerable to hacks since they are connected to the internet. Such attacks may result in the theft of personal information, generating damage and legal ramifications for the manufacturer or homeowner. Intellectual property is another concern that emerges with smart garages. They may employ proprietary or copyrighted technology, generating issues about unlawful usage or replication by others.
- c) **Privacy:** Because smart garages have the ability to collect personal information about homeowners and their automobiles, they create privacy and data protection concerns.
- d) **Compliance:** To operate safely and legally, smart garages must adhere to a variety of rules and standards, including construction requirements and safety norms.

### 5.1.2. Social Issues

- a) Economic inequality: Smart garages are frequently expensive to build and maintain, which may create in economic disparities between those who can and cannot afford them. This may result in a schism between those who benefit from the benefits of smart garages, such as greater convenience and safety, and those who do not.
- b) Privacy problems: Smart garages frequently collect data on how vehicles are utilized, which might raise privacy concerns. Third parties, such as insurance companies or law enforcement organizations, may utilize this data for purposes that the vehicle owner would not anticipate.
- c) Cybersecurity risks: Smart garages are vulnerable to hacks because to their internet connection, which could jeopardize residents' security and privacy. Malicious actors may take advantage of vulnerabilities to obtain access to the garage and home network.
- d) Dependence on technology: The dependency of a smart garage on technology for operation may cause issues for homeowners if the technology fails or the homeowner is unable to use it.
- e) Job displacement: As smart garages grow more common, demand for traditional garage door technicians may fall, resulting in financial challenges for these individuals and their families.

### 5.1.3. Ethical Issues

- a) Privacy and data collection: Smart garages can collect personal information, such as how frequently a vehicle is used, which could reveal individuals' identities. This raises concerns about the possibility of data theft or misuse, putting the homeowner's and their family's privacy at danger.
- b) Discrimination: When smart garages use data to determine access, some groups of people, such as those who cannot afford specific types of vehicles, may be disadvantaged. This may result in unfair treatment and prejudice.
- c) Security: Smart garages are vulnerable to hacking and cyberattacks, putting the homeowner's and their property's safety and security at risk. As a result, it is critical to deploy proper security measures to protect against such hazards.
- d) Dependence on technology: Smart garages rely on technology to operate the garage, which can cause problems for homeowners if the equipment breaks, or they are unable to use it for whatever reason.
- e) Environmental impact: Smart garages may use more energy and generate more garbage than traditional garages, which may have a detrimental impact on the environment.

## **5.2. Advantages**

### **5.2.1. Increased Security**

Smart garages include improved security features such as authentication and verification methods, motion sensors, and surveillance cameras. These elements give an additional layer of protection for the vehicle and other precious things kept in the garage, which can provide the homeowner with a sense of peace and confidence.

### **5.2.2. Remote Access**

A smart garage allows you to remotely access your garage from a smartphone, tablet, or computer. You can quickly open and close the garage door from anywhere, which is useful if you need to give someone access to your garage while you are away.

### **5.2.3. Automated operation**

A smart garage enables the user to automate actions based on factors such as time of day, movement detection, or the user's location. Because of this automation function, the garage door can open and close automatically, eliminating the need for manual operation each time.

### **5.2.4. Activity Monitoring**

A smart garage allows homeowners to watch and monitor the opening and closing of their garage door, which can help them keep track of who has access to their garage.

### **5.2.5. Improved convenience**

Smart garages offer the added advantage of being able to communicate with other smart home products, such as voice assistants and smart locks, to provide an even greater level of convenience.

### **5.3. Limitations**

#### **5.3.1. Cost**

Because of the additional hardware and technology required, smart garages may be more expensive than standard versions. This expense may be difficult for some households and may be a significant barrier to adoption.

#### **5.3.2. Reliability**

A reliable internet connection is essential for the efficient operation of a smart garage. If your internet connection is frequently disconnected or inconsistent, the operation of your smart garage may be hampered and become unreliable.

#### **5.3.3. Complexity**

Installing and configuring a smart garage might be more difficult and time-consuming than configuring a regular garage. This can be especially difficult for homeowners who are unfamiliar with IoT technology or have no experience setting up smart devices.

#### **5.3.4. Privacy concerns**

Smart garages collect and store data regarding garage activity, which may cause some homeowners to be concerned about their privacy. Furthermore, if the verification and authentication method is untrustworthy, unauthorized individuals may gain access to the garage.

#### **5.3.5. Compatibility**

Some homeowners may need to modify or purchase additional hardware to make their present garage door opener system compatible with a smart garage. Some garage doors and opener systems may not work with a smart garage, which may be a limitation for some homeowners.

## **5.4. Future Work**

### **5.4.1. Two factor authentication**

Two-factor authentication is a security approach in which users must submit two distinct authentication factors in order to validate their identity and gain access to a system or application. 2FA can be an effective precaution for a smart garage mobile app to protect the user's account and prevent illegal access. 2FA can be implemented in a variety of ways, including SMS-based verification, app-based verification, and biometric verification.

To implement 2FA in a smart garage mobile application, the developer can include one or more authentication methods. This can involve requesting the user to submit their login credentials as well as a one-time password obtained through an authentication app or SMS. The app might also employ the device's biometric authentication features, such as fingerprint or facial recognition, to verify the user's identity. By enabling 2FA, the app can add an extra layer of security to the user's account and prevent illegal access.

### **5.4.2. Cloud Storage**

Cloud storage is a type of online data storage facility that allows customers to store and access their files and data from anywhere in the world via the internet. Cloud storage can be utilized in the case of a smart garage application to securely store and access data relevant to the user's garage, such as sensor data, photos, and videos.

Some benefits of Cloud storage are-

- **Remote access:** Through cloud storage, users may view their garage data from any device with an internet connection at any time. It's useful, especially when checking on the condition of the garage while no one is at home.
- **Backup and recovery:** Storing garage data on the cloud adds an extra layer of security in the event of data loss due to device failure, theft, or other problems. This is due to the fact that the data is not stored on a single device and may be readily accessed from the cloud if necessary.
- **Scalability:** Cloud storage is adaptable and can be readily altered to meet storage needs without the need to invest in extra hardware or pay any upfront fees. This is

especially useful for smart garage customers who have fluctuating storage requirements over time.

- Security: Cloud storage companies often incorporate rigorous security measures, such as access limits and data encryption, to keep customer data private and prevent illegal access or theft.

#### **5.4.3. Door locking system**

Including a door locking system in a smart garage has various advantages. Here are some of the benefits of a door locking system in a smart garage:

- Security: A door locking system in a smart garage can play an important part in securing your home by preventing illegal entry to your garage. It becomes much more important if your garage is attached to your home or includes valuable possessions. The system adds an extra degree of security, guaranteeing that only authorized people have access to your garage.
- Convenience: A smart door locking system allows you to manage garage access remotely via a smartphone app. With this feature, you can offer someone access to your garage even if you are not physically present at home.
- Peace of mind: Having a safe garage can be reassuring, especially if you are away from home for an extended period of time.

#### **5.4.4. Notification System**

Notifications in a smart garage can be quite valuable for keeping the homeowner up to date on the status of the garage door and any potential security issues. Notifications in a smart garage can be used to warn the homeowner when the garage door is left open, send a notice when the door is closed, or notify the homeowner of any attempted unlawful entrance to the garage:

- Door status notifications: A smart garage system adds an extra degree of security by notifying you anytime your garage door is opened or closed. This manner, you can feel more secure knowing that you can monitor who comes and goes from your



garage. You'll receive notifications directly to your device, allowing you to stay up to date no matter where you are.

- **Security alerts:** Some smart garage systems allow you to set up warnings for prospective break-ins or when the garage door is left open for an extended period of time. This additional layer of security keeps you informed and can provide you with peace of mind in the event of any questionable activity. With these notifications, you can remain on top of things whether you're at home or away.
- **Maintenance reminders:** Smart garage systems can also be configured to send notifications when it's time for routine maintenance, such as lubricating the garage door opener or replacing the batteries in garage door sensor.

#### **5.4.5. Power Management**

Power consumption management is an important component of a smart garage that can assist reduce energy usage and electricity expenses. Here are some ideas for incorporating power management into your smart garage:

- **Energy-efficient garage door openers:** Certain garage door openers are engineered to utilize less energy than other versions. Choosing an Energy Star certified garage door opener will help you conserve energy and reduce your monthly power bill.
- **Smart power strips:** Smart power strips are a convenient way to manage electricity to different devices in the garage, such as charging stations and power tools. With the ability to turn off devices when they are not in use.

## Chapter 6: References

Admin, 2022. *Weighing Machine using Arduino Load Cell & HX711 Module*. [Online]

Available at: [https://how2electronics.com/weighing-machine-arduino-load-cell-](https://how2electronics.com/weighing-machine-arduino-load-cell-hx711/#HX711_Load_Cell_Amplifier)

[hx711/#HX711\\_Load\\_Cell\\_Amplifier](https://how2electronics.com/weighing-machine-arduino-load-cell-hx711/#HX711_Load_Cell_Amplifier)

[Accessed 15 January 2023].

Adobe Communications Team, 2022. *Waterfall Methodology: Project Management | Adobe Workfront*. [Online]

Available at:

<https://business.adobe.com/blog/basics/waterfall#:~:text=The%20Waterfall%20methodology%20%E2%80%94%20also%20known,before%20the%20next%20phase%20begins>

:

[Accessed 20 November 2022].

Aqib, M., 2019. *Optical Character Recognition Using Raspberry Pi With OpenCV and Tesseract | Raspberry Pi | Maker Pro*. [Online]

Available at: <https://maker.pro/raspberry-pi/tutorial/optical-character-recognizer-using-raspberry-pi-with-opencv-and-tesseract>

[Accessed 3 March 2023].

Gomathy, C. K., 2022. *(PDF) IOT BASED ON GARAGE DOOR OPENERS*. [Online]

Available at:

[https://www.researchgate.net/publication/357748542\\_IOT\\_BASED\\_ON\\_GARAGE\\_DOOR\\_OPENERS](https://www.researchgate.net/publication/357748542_IOT_BASED_ON_GARAGE_DOOR_OPENERS)

[Accessed 20 November 2022].

Hasan, M., 2022. *Number of connected IoT devices growing 18% to 14.4 billion globally*. [Online]

Available at: <https://iot-analytics.com/number-connected-iot-devices/>

[Accessed 19 November 2022].

Jain, R., 2019. *IoT Smart Garage Door Opener using Raspberry Pi*. [Online]

Available at: <https://circuitdigest.com/microcontroller-projects/iot-smart-garage-door->

opener-using-raspberry-pi

[Accessed 20 November 2022].

James, 2018. *File Operations in Flutter – Read and Write Files – Easiest Example – MOBILE PROGRAMMING*. [Online]

Available at: <https://www.coderzheaven.com/2018/12/26/file-operations-in-flutter-read-and-write-files-easiest-example/>

[Accessed 20 March 2023].

Martin, M., 2022. *Prototype Model in Software Engineering*. [Online]

Available at: <https://www.guru99.com/software-engineering-prototyping-model.html>

[Accessed 22 November 2022].

Panigrahi, K. K., 2023. *Difference between Unit Testing and System Testing*. [Online]

Available at: <https://www.tutorialspoint.com/difference-between-unit-testing-and-system-testing#:~:text=Unit%20testing%20and%20system%20testing%20are%20two%20different%20types%20of,software%20product%20as%20a%20whole.>

[Accessed 3 April 2023].

Projects Factory, 2021. *IoT Smart Garage Door Opener | Electrical & Electronics Projects | Academic Projects*. [Online]

Available at: <https://projectsfactory.in/product/iot-smart-garage-door-opener/>

[Accessed 20 November 2022].

smartdraw, 2023. *Activity Diagram - Activity Diagram Symbols, Examples, and More*.

[Online]

Available at: <https://www.smartdraw.com/activity-diagram/#:~:text=Learn%20More-,What%20is%20an%20Activity%20Diagram%3F,in%20a%20use%20case%20diagram.>

[Accessed 2 April 2023].

TechTarget, 2022. *What is block diagram? | Definition from TechTarget*. [Online]

Available at: <https://www.techtarget.com/whatis/definition/block-diagram#:~:text=A%20block%20diagram%20is%20a,to%20show%20relationships%20between%20them.>

[Accessed 23 December 2022].

Zahid, H., 2022. *How to control the Raspberry Pi device using a smartphone*. [Online]

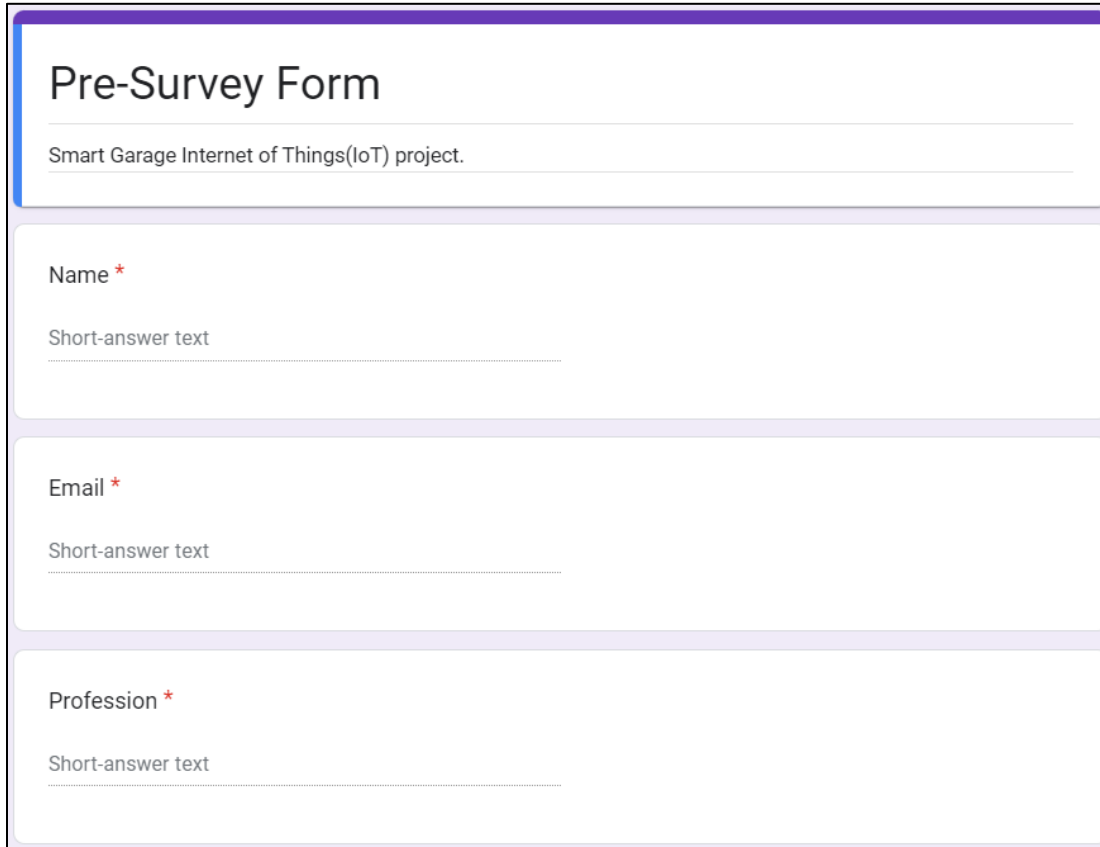
Available at: <https://linuxhint.com/control-raspberry-pi-using-smartphone/#:~:text=The%20RaspController%20is%20an%20Android,pins%20and%20access%20command%20line.>

[Accessed 10 January 2023].

## Chapter 7: Appendix

### 7.1. Appendix 1: Pre-Survey

#### 7.1.1. Pre-Survey form



The image shows a digital form titled "Pre-Survey Form" for a "Smart Garage Internet of Things(IoT) project". The form is divided into three sections, each with a light purple header bar. The first section is for "Name", the second for "Email", and the third for "Profession". Each section contains a label with a red asterisk indicating it is required, followed by a "Short-answer text" input field. The form has a blue border on the left and top, and a purple border on the right and bottom.

**Pre-Survey Form**

Smart Garage Internet of Things(IoT) project.

**Name \***

Short-answer text

**Email \***

Short-answer text

**Profession \***

Short-answer text

Do you know about IoT domain? \*

☐ Yes

☐ No

☐ Maybe

Do you use any IoT system? \*

☐ Yes

☐ No

☐ Maybe

Have you ever heard about a smart Garage? \*

☐ Yes

☐ No

What do you think about a Garage which authenticate and verifies vehicles? \*

☐ Useful

☐ Time Saving

☐ Effortless

☐ All of the above

---

Will you use this system? \*

☐ Yes

☐ No

☐ Maybe

---

Any feedback?

Long-answer text

---

### 7.1.2. Pre-Survey Result

Name

16 responses

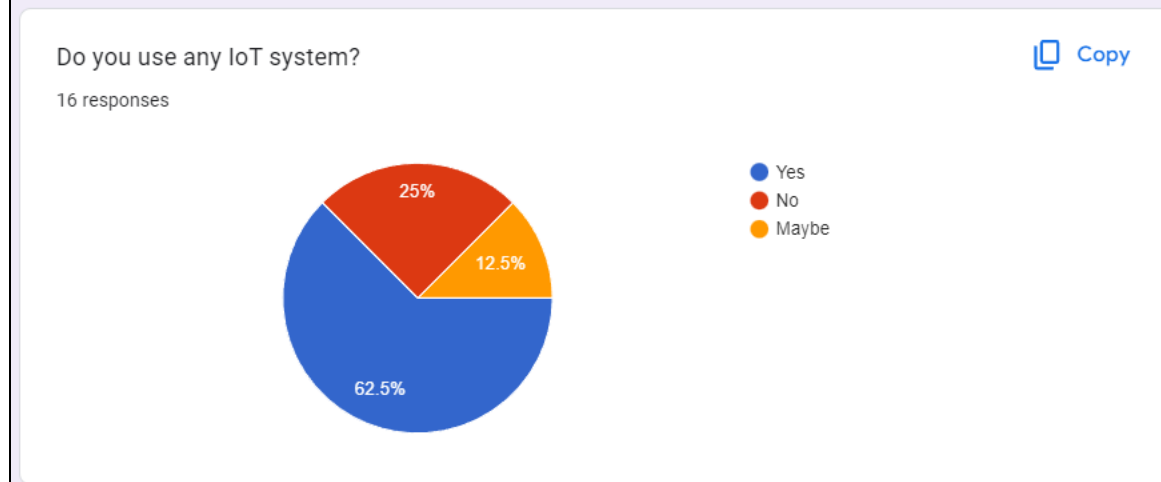
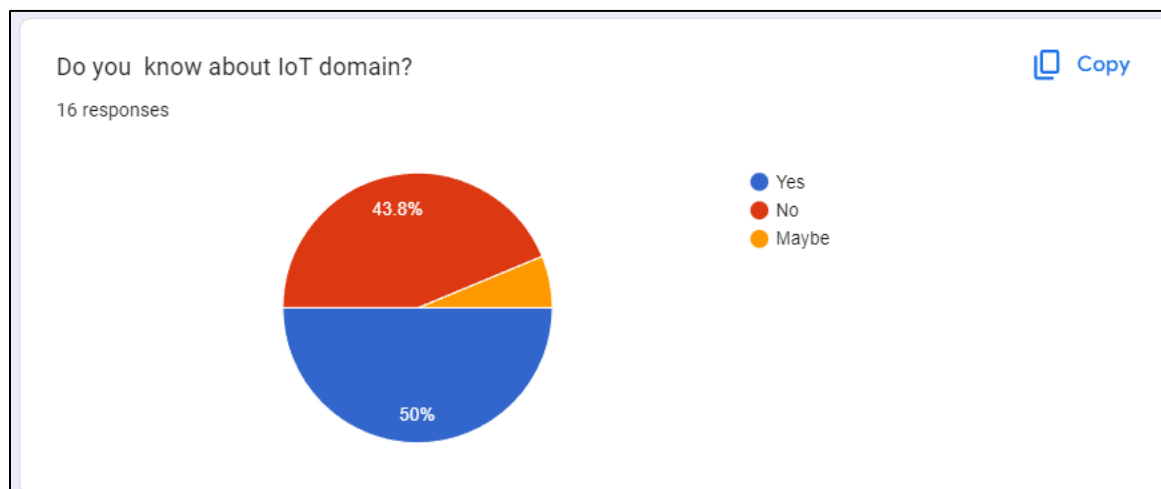
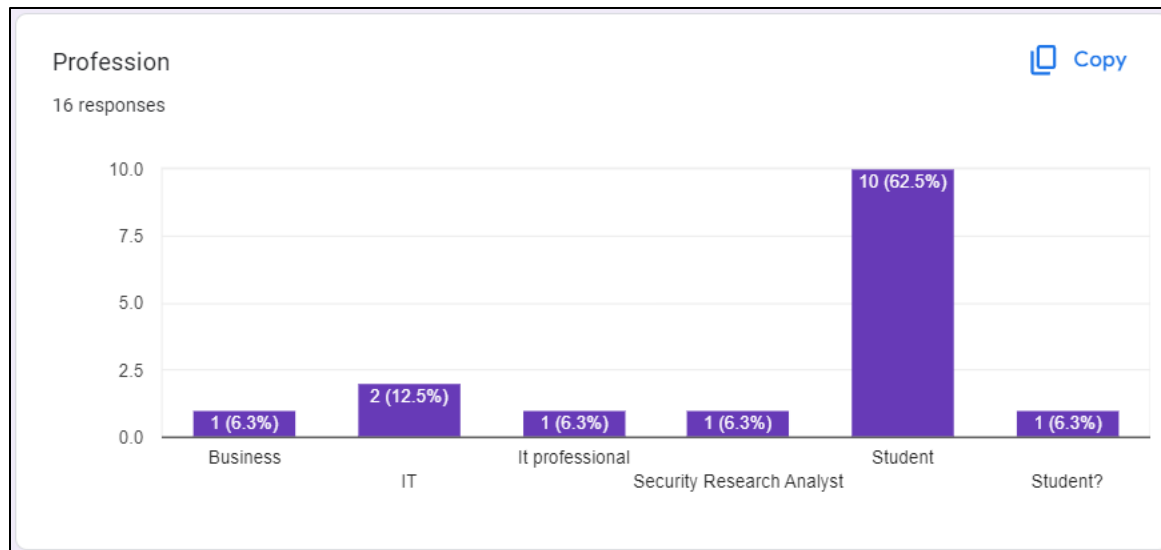
Sahil maharjan
Jenisha
Sarthak Rana
Sushi
Niraj maskey
Sabrina Tuladhar
Dorjee Tshering Sherpa
Soniya
Aarya Maharjan

Email

16 responses

Maharjansahil91@gmail.com
jenishamj27@gmail.com
sarthakbrana@gmail.com
Gyawalisush02@gmail.com
nirzmaskey@gmail.cim
sabrina.tldhr@gmail.com
np01cp4s210031@islingtoncollege.edu.np
Soniyas.shrestha@gmail.com
aaryamaharjan689@gmail.com

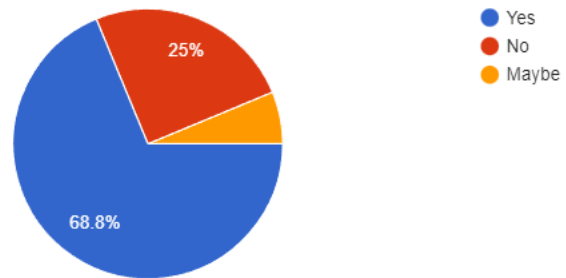




Have you ever heard about a smart Garage?

 Copy

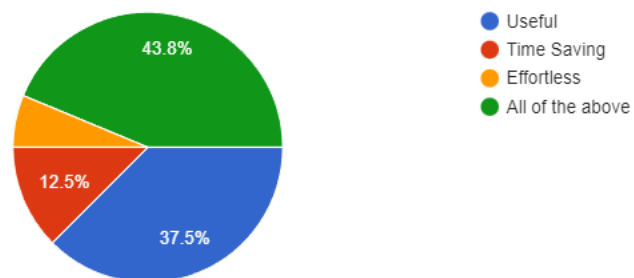
16 responses

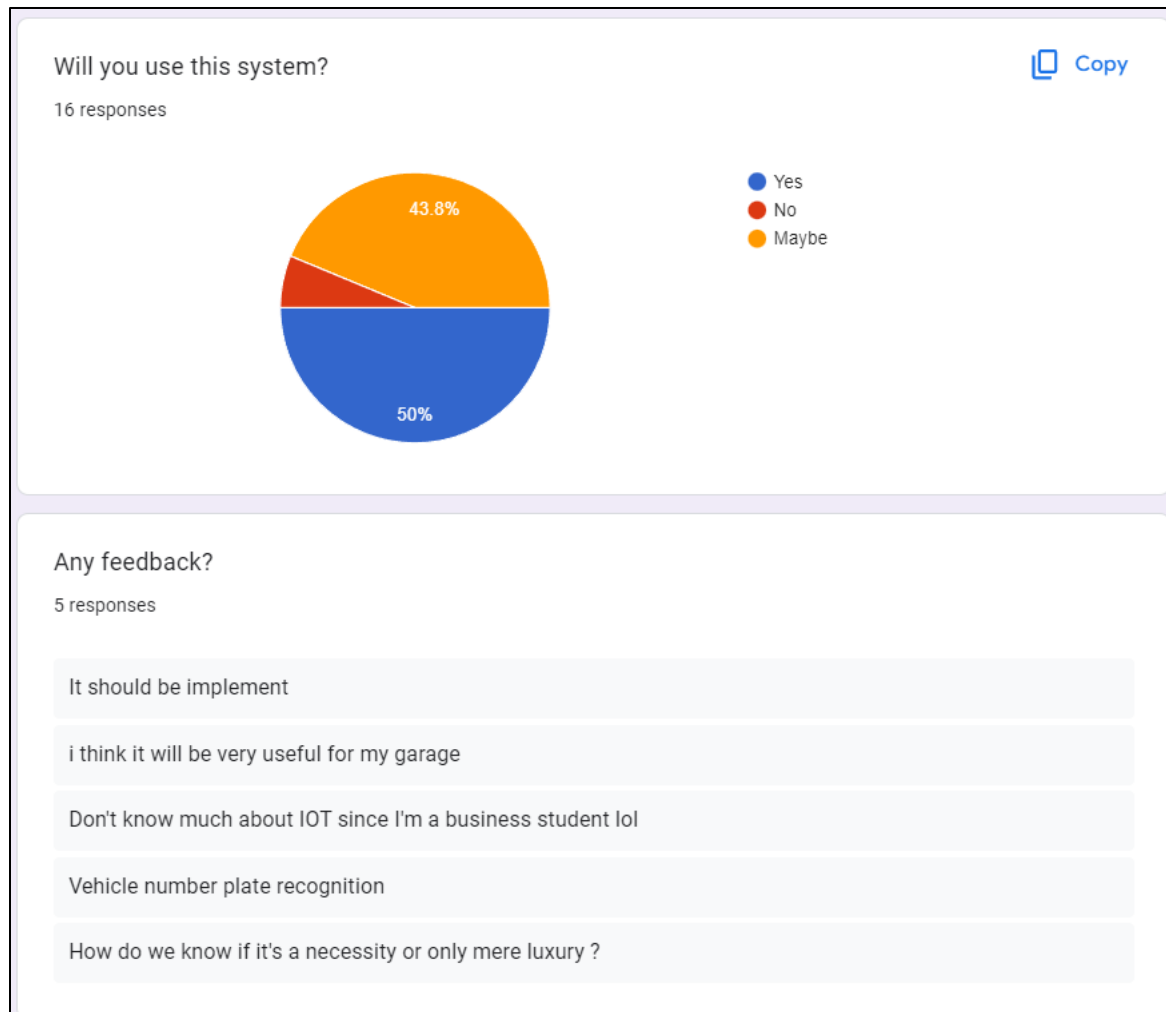


What do you think about a Garage which authenticate and verifies vehicles?

 Copy

16 responses





## 7.2. Appendix 2: Post-Survey

### 7.2.1. Post-Survey form

<h2>Post-Survey Results</h2> <hr/> <p>Form description</p> <hr/>
<p>Name *</p> <hr/> <p>Short-answer text</p> <hr/>
<p>Email *</p> <hr/> <p>Short-answer text</p> <hr/>
<p>Profession *</p> <hr/> <p>Short-answer text</p> <hr/>

Please rate your level of liking for the feature of the shutter opening using sensors on a scale <sup>\*</sup> of one to ten.

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

Using a rating system from one to ten, how satisfied were you with the Authentication feature? <sup>\*</sup>

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

Using a scale from one to ten, how satisfied were you with the Verification feature? <sup>\*</sup>

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

How would you score your preference for the mobile app-controlled shutter opening feature on <sup>\*</sup>a scale of one to ten?

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

Please rate your level of liking for the feature of storing logs on a scale of one to ten. <sup>\*</sup>

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

What score would you give the developed mobile application on a scale of one to ten? <sup>\*</sup>

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	excellent

Any Feedback?

Long-answer text

---

### 7.2.2. Post-Survey Result

Name

9 responses

Sahil maharjan

Sabrina Tuladhar

Arya Amatya

Jenisha

Abhilasha Shrestha

Niraj Maskey

Sujen Shrestha

Sarthak Rana

Soniya

Email

9 responses

Maharjansahil91@gmail.com

sabrina.tldhr@gmail.com

Np01nt4s210101@islingtoncollege.edu.np

Jenishamj27@gmail.com

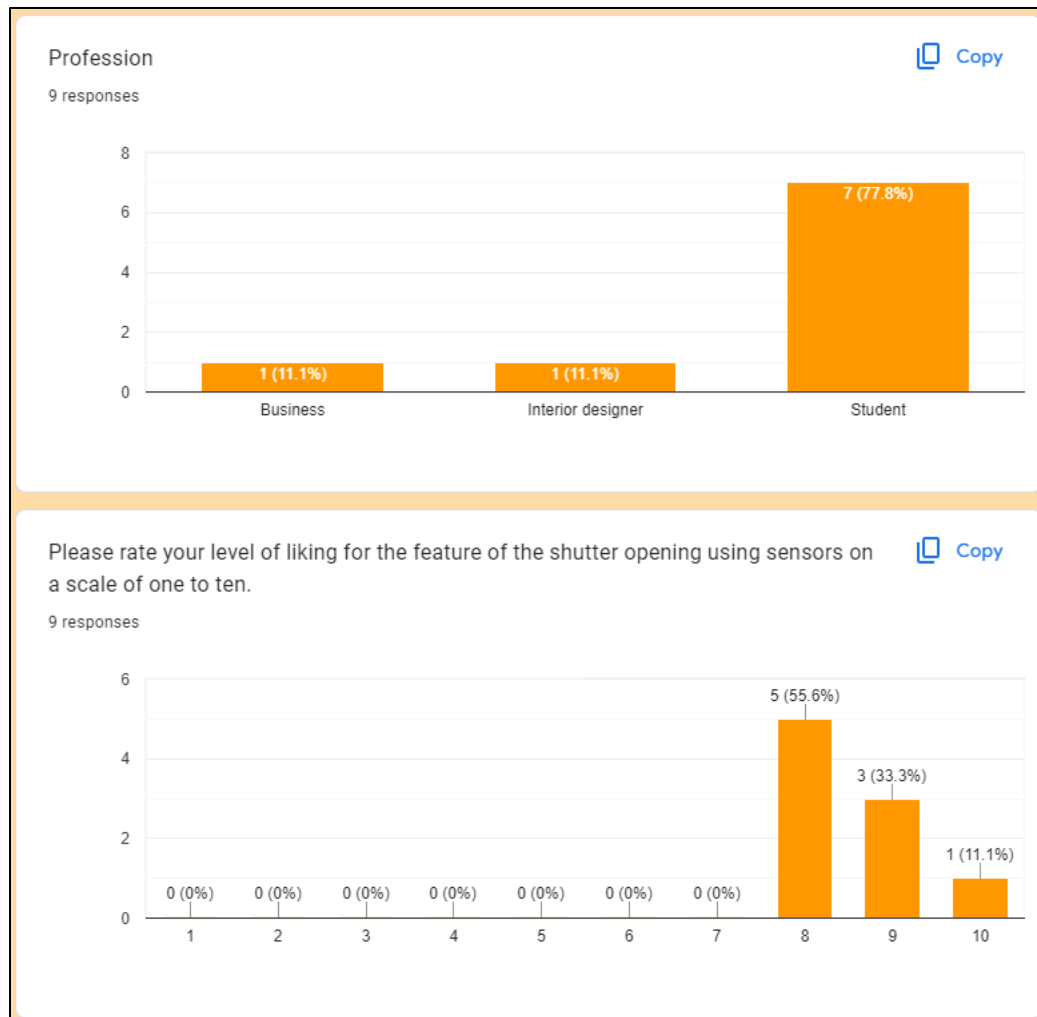
abhilashashrestha07@gmail.com

Nirzmaskey@gmail.com

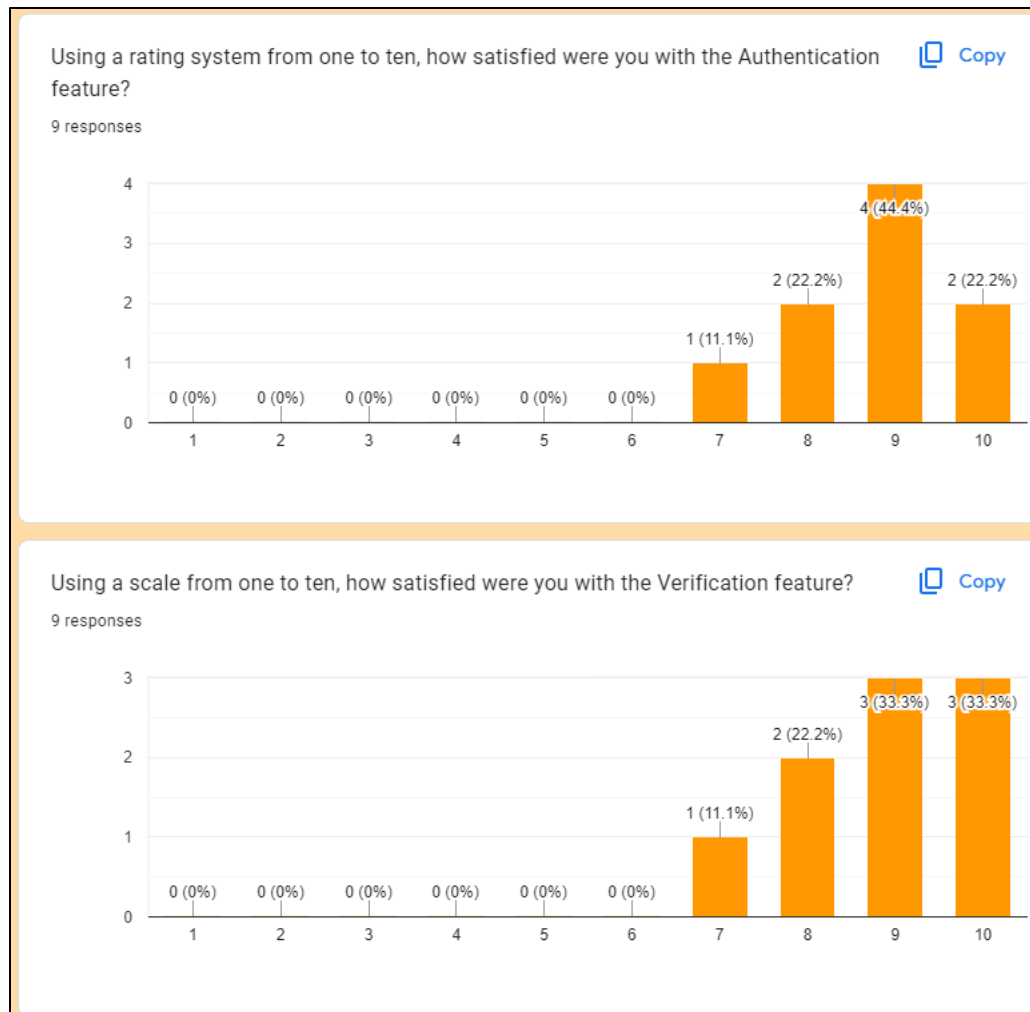
sujeopro@gmail.com

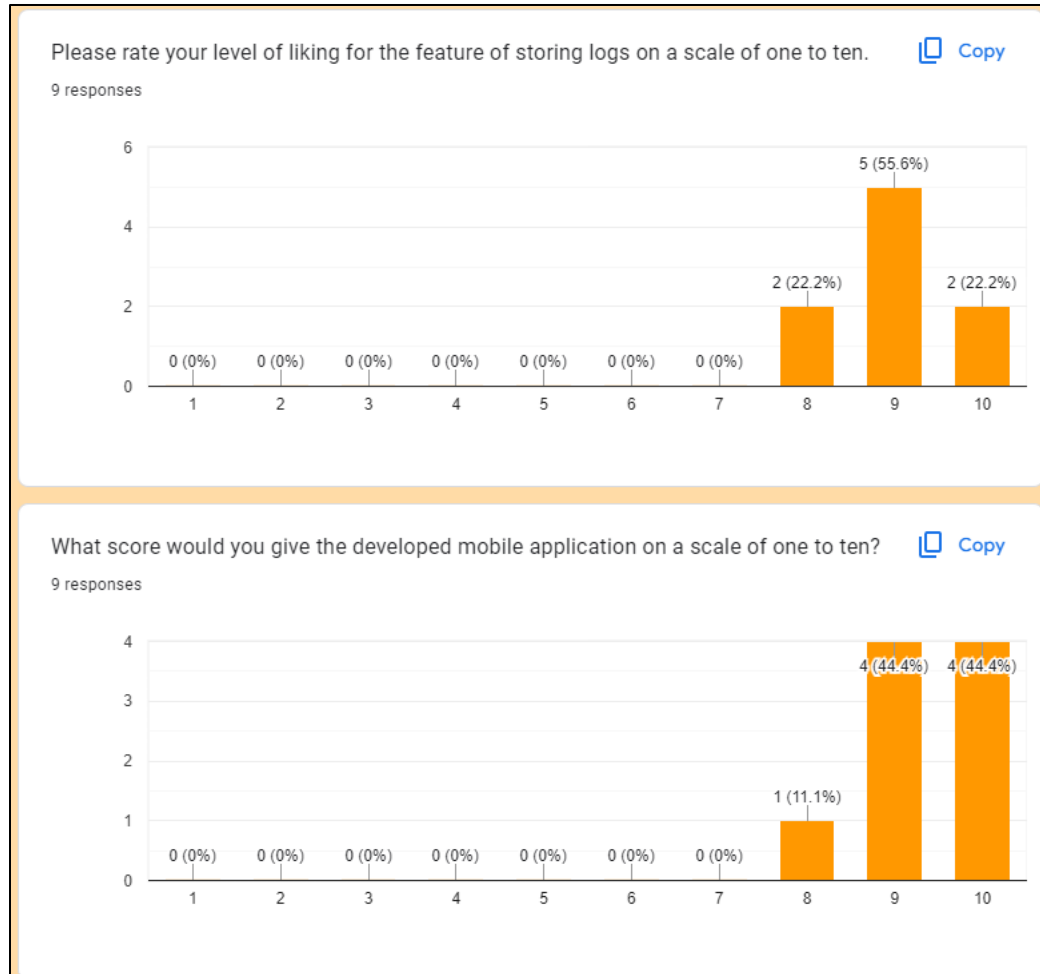
sarthakbrana@gmail.com

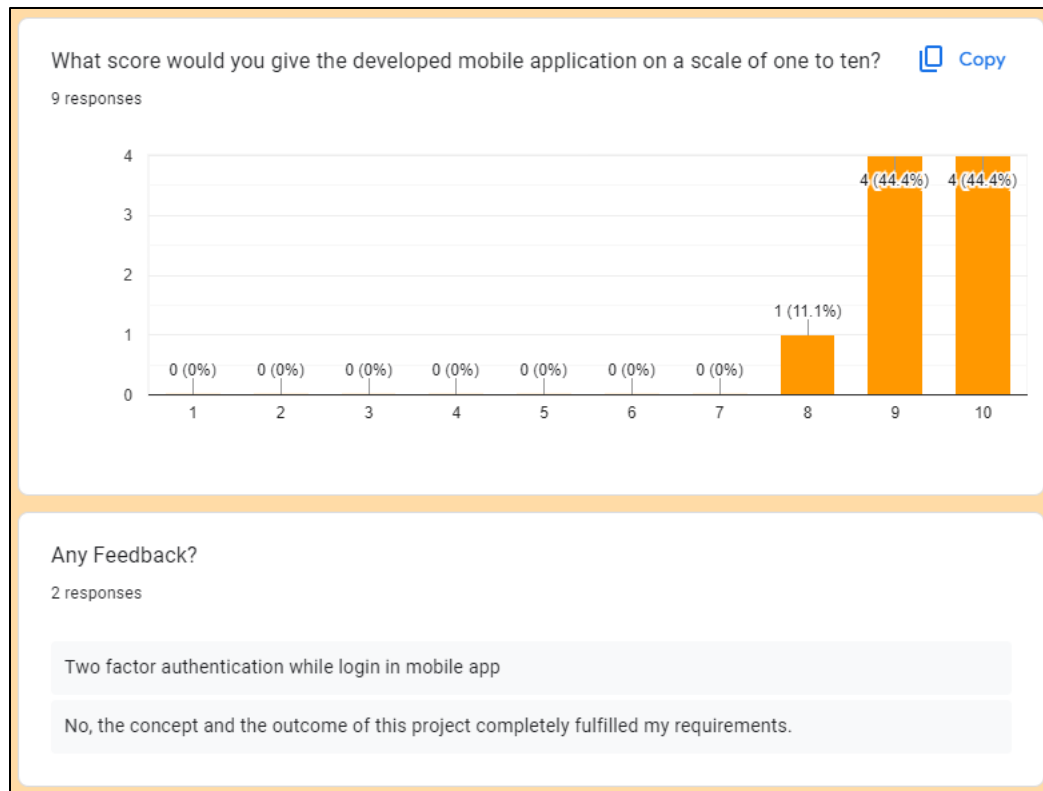
Sonyaashrestha@gmail.com











## 7.3. Appendix 3: Sample Code

### 7.3.1. Backend

#### 7.3.1.1. main.py

```
import os
import RPi.GPIO as myGPIO
from hx711 import HX711
import time
import datetime
myGPIO.setmode(myGPIO.BCM)
myGPIO.setwarnings(False)
import json
"""Functions"""
def Ultrasonic():
    '''Returns Distance'''
    TRIG = 17
    ECHO = 27

    myGPIO.setup(TRIG,myGPIO.OUT)
    myGPIO.setup(ECHO,myGPIO.IN)

    myGPIO.output(TRIG, False)
    time.sleep(2)

    myGPIO.output(TRIG, True)
    time.sleep(0.0001)
    myGPIO.output(TRIG, False)
    while myGPIO.input(ECHO)==0:
        pulse_start = time.time()
    while myGPIO.input(ECHO)==1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance
def Loadcell():
    hx = HX711(dout_pin=5, pd_sck_pin=6)
    reading = hx.get_raw_data_mean()
    reading = hx.get_data_mean()
    hx.set_scale_ratio(reading)
    weight = hx.get_weight_mean(20)
    return weight
def Click():
    cmd = 'fswebcam 640x480 -pYUYV test.jpg'
```

```
os.system('fswebcam 640x480 -pYUYV currentNum/test.jpg')
def OCR(NumPlate):
    i = 1
    os.system("tesseract ../currentNum/test.jpg ../numTxt/output --dpi 300 --oem
1 --psm 6")
    currentFile = open("../numTxt/output.txt", "r")
    CurrentNum = currentFile.read()
    while i <= 5:
        os.system("tesseract ../validNum/"+ str(i) + ".jpg
../numTxt/output"+str(i)+" --dpi 300 --oem 1 --psm 6")
        existingFile = open("../numTxt/output"+str(i)+".txt", "r")
        ExistingNum = existingFile.read()
        if ExistingNum == CurrentNum:
            print("Authentication Is Successful")
            NumPlate.extend([str(CurrentNum)])
            return [True, NumPlate]
        i = i + 1
    NumPlate.extend([str(CurrentNum)])
    return [False, NumPlate]
def Open():
    print("Open Shutter")

"""Main"""
file = open("../txt/dic.json")
dic = json.load(file)
NumPlate = dic["NumberPlate"]
DateNTime = dic["DateandTime"]
Allow = dic["Allowed"]
Run = True
Distance = 0
Weight = 0
VFON = False
AUTH = False
while Run == True:
    Distance = Ultrasonic()
    print("Trying Authentication ...")
    while Distance <= 20:
        Click()
        tempList = OCR(NumPlate)
        AUTH = tempList[0]
        NumPlate = tempList[1]
        if AUTH == True:
            break
        else:
            Distance = Ultrasonic()
```

```

        print("Trying Authentication Again ...")
weight = Loadcell()
print("Trying Verification")
while weight >= 1:
    VFON = True
    if VFON == True:
        print("Verification is Successful")
        break
    else:
        weight = Loadcell()
        print("Trying Verification Again ...")
if AUTH == True & VFON == True:
    Open()
    Allow.extend(["True"])
    dt = datetime.datetime.now()
    dnt = dt.strftime("%d/%m/%y %H:%M")
    DateTime.extend([str(dnt)])
    Run = False
else:
    Allow.extend(["False"])
with open("../txt/dic.json","w") as file:
    json.dump(dic, file)

```

### 7.3.1.2 app.py

```

• from flask import Flask, request
• import main
• import json
• app = Flask(__name__)
• file = open("../txt/dic.json")
• dic = json.load(file)
•
• @app.get("/data")
• def get_data():
•     return dic
•
• @app.post("/openShutter")
• def MobileAppOpen():
•     request_data = request.get_json()
•     data = {"func": request_data["func"]}
•     func_name = data['func']
•     if func_name == 'Open':
•         main.Open()
•         return data, 201

```

### 7.3.2. Frontend

#### 7.3.2.1. main.dart

```
import 'package:flutter/material.dart';
import 'package:smartgarrage/Pages/HomePage.dart';
import 'package:smartgarrage/Pages/Login.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:get_it/get_it.dart';
import 'package:smartgarrage/Services/firebase_service.dart';
import 'package:smartgarrage/Pages/LogInfo.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  GetIt.instance.registerSingleton<FirebaseService>(
    FirebaseService(),
  );
  runApp(const SmartGarage( ));
}

class SmartGarage extends StatelessWidget {
  const SmartGarage({Key? key}) :super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Garage',
      theme: ThemeData(
        primarySwatch: Colors.red,
      ),
      initialRoute: 'Login',
      routes: {
        'Login': (context) => Login(),
        'Home':(context) => HomePage(),
        'Log':(context) => LogInfo(),
      },
    );
  }
}
```

## 7.3.2.2. Login.dart

```
// ignore_for_file: file_names
import 'package:flutter/material.dart';
import 'package:smartgarrage/Pages/HomePage.dart';
class Login extends StatefulWidget{
  const Login({super.key});
  @override
  State<StatefulWidget> createState(){
    return _LoginState();
  }
}
class _LoginState extends State<Login>{
  double? _deviceHeight, _deviceWidth;
  final GlobalKey<FormState> _loginFormkey = GlobalKey<FormState>();
  String? _psd;
  @override
  Widget build(BuildContext context) {
    _deviceHeight = MediaQuery.of(context).size.height;
    _deviceWidth = MediaQuery.of(context).size.width;
    return Scaffold(
      body: SafeArea(
        child: Container(
          padding: EdgeInsets.symmetric(horizontal: _deviceWidth! * 0.05,
          ),
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              mainAxisAlignment: MainAxisAlignment.max,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                _titleWidget(),
                _loginForm(),
                _loginButton(),
              ],
            ),
          ),
        ),
      ),
    );
  }
  Widget _titleWidget() {
    return const Text("Smart Garage",
      style: TextStyle(
        color: Color.fromARGB(255, 255, 0, 0),
        fontSize: 40,

```



```
        fontWeight: FontWeight.w600,
      )
    );
  }
  Widget _loginButton(){
    return MaterialButton(
      onPressed: _loginUser,
      minWidth: _deviceWidth! * 0.70,
      height: _deviceHeight! * 0.06,
      color: Colors.red,
      child: const Text("Login",
        style: TextStyle(
          color: Colors.white,
          fontSize: 25,
          fontWeight: FontWeight.w600,
        ),
      ),
    );
  }
  Widget _loginForm() {
    return SizedBox(
      height: _deviceHeight! * 0.20,
      child: Form(
        key:_loginFormkey,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          mainAxisAlignment: MainAxisAlignment.max,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            _admin(),
            _passwordTextField(),
          ],
        ),
      ),
    );
  }
  Widget _admin(){
    return const Text("Admin",
      style: TextStyle(
        color: Color.fromARGB(255, 0, 0, 0),
        fontSize: 35,
        fontWeight: FontWeight.w600,
      )
    );
  }
}
```

```

Widget _passwordTextField(){
  return TextFormField(
    obscureText: true,
    decoration: const InputDecoration(hintText: "Password..."),
    onSave: (_value){
      setState(() {
        _psd = _value;
      });
    },
    validator: (_value) => _value == "SmartGarage"
      ? null
      : "Please enter the correct password",
  );
}

void _loginUser(){
  print(_loginFormkey.currentState!.validate());
  if (_loginFormkey.currentState!.validate()) {
    Navigator.push(context, MaterialPageRoute(builder: (BuildContext _context){
      return HomePage();
    }));
  }
}
}

```

### 7.3.2.3.Homepage.dart

```

// ignore: duplicate_ignore
// ignore: file_names
// ignore_for_file: file_names
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter/material.dart';
import 'package:smartgarrage/Services/firebase_service.dart';
class HomePage extends StatelessWidget {
  //const HomePage({super.key});
  double? _deviceHeight, _deviceWidth;
  FirebaseServic? _firebaseServic;
  final usernameController = TextEditingController();
  final passwordController = TextEditingController();
  @override
  Widget build(BuildContext context) {
    _deviceHeight = MediaQuery.of(context).size.height;
    _deviceWidth = MediaQuery.of(context).size.width;
    return Scaffold(
      appBar: AppBar(

```

```

        title: const Text("Smart Garage"
    ),
    actions: [
        Padding(
            padding: const EdgeInsets.only(right: 20),
            child: GestureDetector(
                onTap: (){
                    Navigator.pushNamed(context, 'Log');
                },
                child: const Icon(Icons.feed_rounded),
            ),
        ),
        Padding(
            padding: const EdgeInsets.only(right: 40),
            child: GestureDetector(
                onTap:() async {
                    await _firebaseService?.logout();
                    Navigator.popAndPushNamed(context, 'Login');
                },
                child: const Icon(Icons.logout_rounded),
            ),
        ),
    ],
),
body: SafeArea(
    child: Container(
        padding: EdgeInsets.symmetric(vertical: _deviceHeight! * 0.3,
    ),
        child: Center(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                mainAxisAlignment: MainAxisAlignment.max,
                crossAxisAlignment: CrossAxisAlignment.center,
                children:[
                    _openButton(),
                ],
            ),
        ),
    ),
),
);
}
Widget _HomePage(){
    return Container(
        height: _deviceHeight!*0.3,

```

```
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.spaceBetween,  
          mainAxisAlignment: MainAxisAlignment.max,  
          crossAxisAlignment: CrossAxisAlignment.center,  
        ),  
      ),  
    );  
  }  
  Widget _openButton(){  
    return MaterialButton(  
      onPressed: () {  
        sendPostRequest();  
      },  
      minWidth: _deviceWidth! * 0.30,  
      height: _deviceHeight! * 0.05,  
      color: Colors.red,  
      child: const Text("Open",  
        style: TextStyle(  
          color: Colors.white,  
          fontSize: 25,  
          fontWeight: FontWeight.w600,  
        ),  
      ),  
    ),  
  );  
}  
  
Future<void> sendPostRequest() async {  
  final url = Uri.parse('http://192.168.1.139:5000/openShutter');  
  final headers = {'Content-Type': 'application/json'};  
  final body = jsonEncode({  
    "func": "Open"  
  });  
  final response = await http.post(url, headers: headers, body: body);  
  if (response.statusCode == 201) {  
    print('POST request successful');  
  } else {  
    print('POST request failed with status: ${response.statusCode}');  
  }  
}
```

## 7.3.2.4.LogInfo.dart

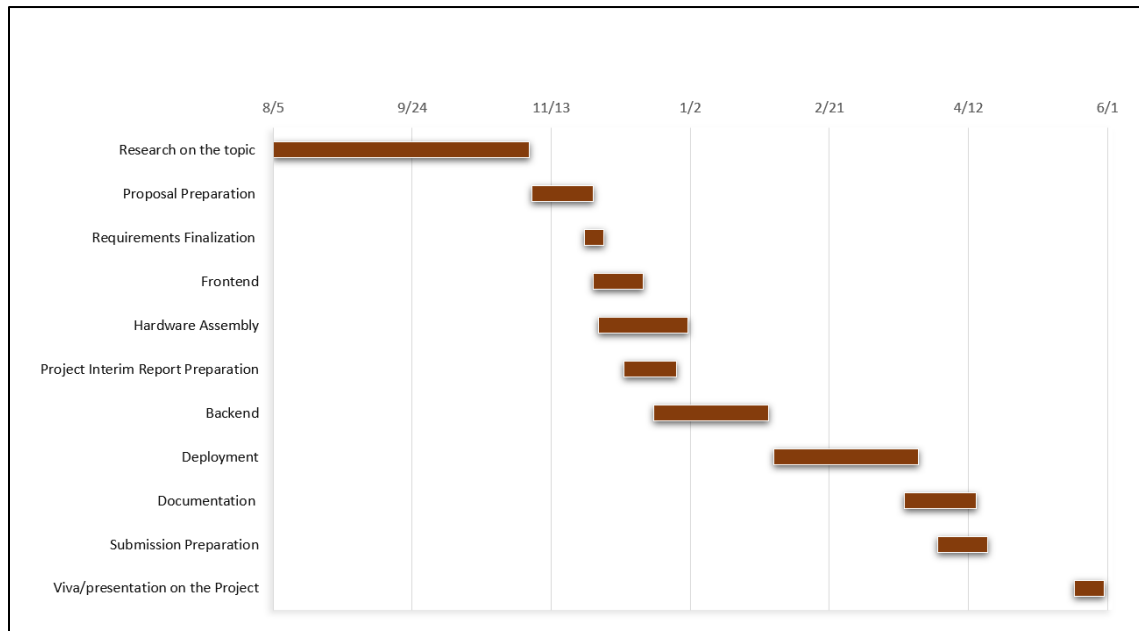
```
import 'dart:convert';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';
import 'package:smartgarrage/Services/firebase_service.dart';
import 'package:http/http.dart' as http;
class LogInfo extends StatefulWidget {
  const LogInfo({Key? key}) : super(key: key);
  @override
  State<LogInfo> createState() => _LogInfo();
}
class _LogInfo extends State<LogInfo>{
  String textFromFile = '';
  FirebaseServic? _firebaseServic;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Smart Garage"
      ),
      actions: [
        Padding(
          padding: const EdgeInsets.only(right: 20),
          child: GestureDetector(
            onTap: (){
              Navigator.pushNamed(context, 'Home');
            },
            child: const Icon(Icons.broadcast_on_personal_rounded),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(right: 40),
          child: GestureDetector(
            onTap:() async {
              await _firebaseServic?.logout();
              Navigator.popAndPushNamed(context, 'Login');
            },
            child: const Icon(Icons.logout_rounded),
          ),
        ),
      ],
    ),
    body: Center(
      child: Column(
```

```
mainAxisAlignment: MainAxisAlignment.start,
children: [
  const SizedBox(
    //Use of SizedBox
    height: 10,
  ),
  ElevatedButton(onPressed: (() {
    _fetchData();
    getData();
  })),
  child: const Text(
    'Get Data'
  ),
),
const SizedBox(
  //Use of SizedBox
  height: 5,
),
ElevatedButton(onPressed: (() {
  clear();
})),
child: const Text(
  'Clear'
),
),
const SizedBox(
  //Use of SizedBox
  height: 30,
),
Text(
  textFromFile,
),
],
)
),
);
}
getData() async{
  final directory = await getExternalStorageDirectory();
  final file = File('${directory!.path}/test.txt');
  final response = await file.readAsString();
  setState(() {
    textFromFile = response;
  });
}
```

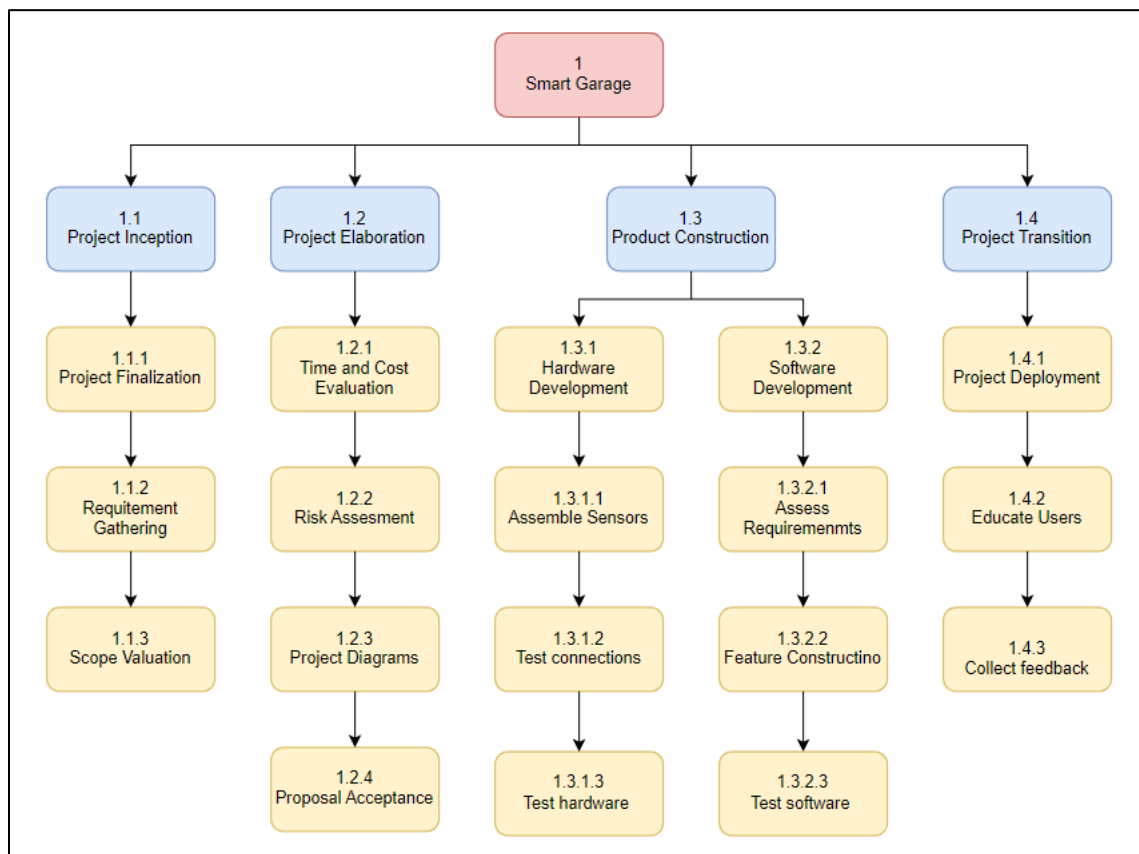
```
clear(){
    setState(() {
        textFromFile = '';
    });
}
void _fetchData() async {
    var url = Uri.parse('http://192.168.1.139:5000/data');
    var response = await http.get(url);
    final Map<String, dynamic> data = json.decode(response.body);
    String display = "          Date & Time          Numberplate          Allowed" ;
    List<dynamic> DateandTime = data['DateandTime'];
    int num = DateandTime.length;
    int i = 0;
    while (i < num) {
        String Allowed = data['Allowed'][i];
        String DateandTime = data['DateandTime'][i];
        String NumberPlate = data['NumberPlate'][i];
        display +=
"\n      $DateandTime          $NumberPlate          $Allowed";
        i++;
    }
    final directory = await getExternalStorageDirectory();
    File = File('${directory!.path}/test.txt');
    file.writeAsStringSync(display);
}
}
```

## 7.4. Appendix 4: Designs

### 7.4.1. Gantt chart

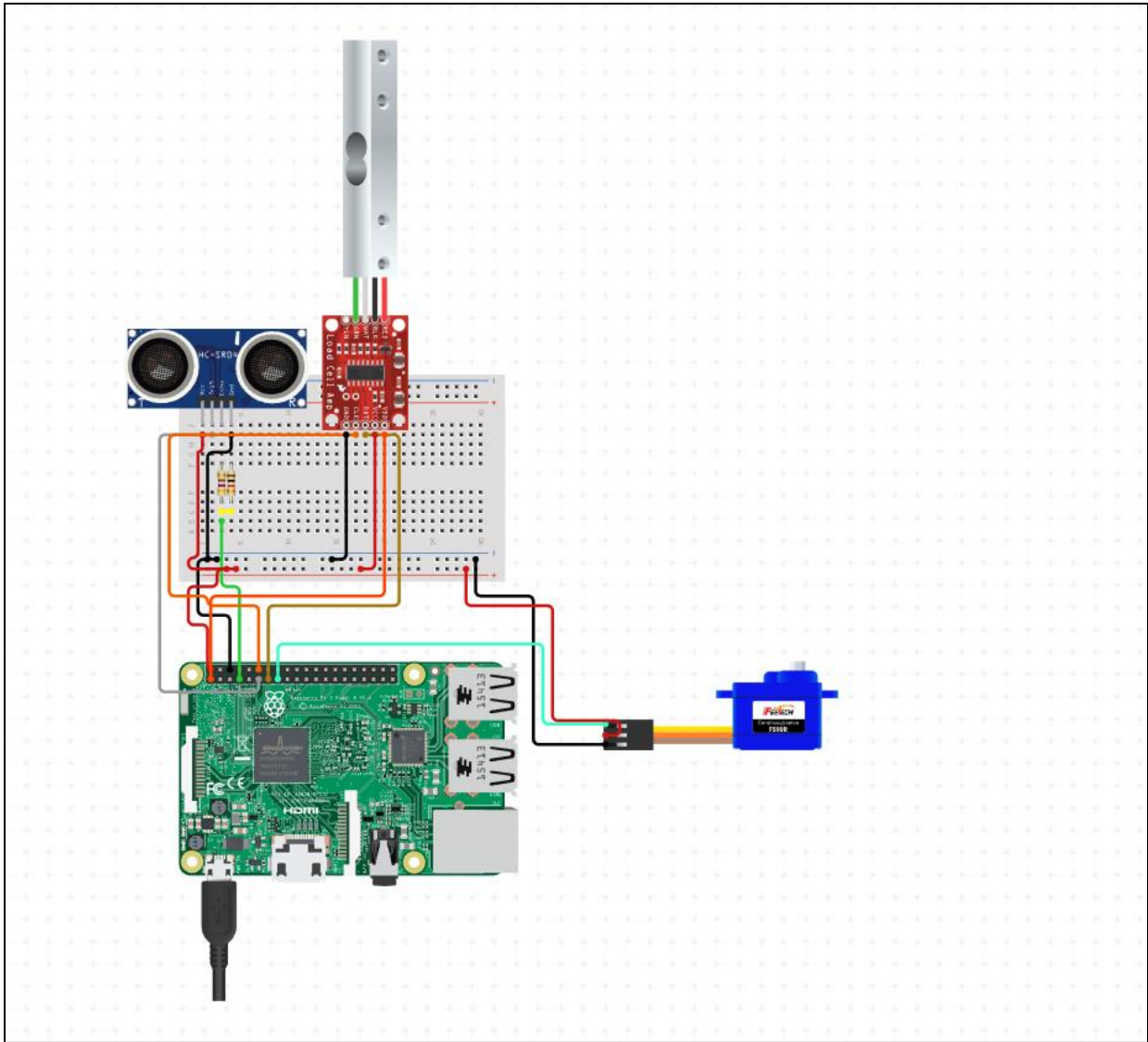


### 7.4.2. Work Breakdown Structure





### 7.4.3. Hardware Architecture



**7.4.4. Block Diagram**