Smart Café for Combating Covid-19

SC3-19

Supervision

DR. MAI RAMADAN

DEPARTMENT OF INFORMATION TECHNOLOGY

TA. ATRAB AHMED

DEPARTMENT OF INFORMATION TECHNOLOGY

Graduation Project

2021

Smart Café for Combating Covid-19

Submitted by:

1. Ahmed Ahmed AbuFoda
2. Abdelrahman Ramzy Dar
3. Aya Mostafa Elsheikh
4. Aisha Mohamed Najeeb
5. Toqa Ahmed Helal
6. Mohamed samy El-kharadly
7. Mohamed Ahmed Esmail
8. Mostafa kamel Mahmoud
9. Mahmoud Mohamed Abdelnabi
10. Maged Mohamed Ahmed

Supervised by:

Dr. Mai Ramadan

Eng. Atrab Ahmed

Acknowledgement

Thank you to the Allah the Almighty because of His power and blessings, we had able to complete our final year project entitled. Although we had undergone many obstacles and queries throughout the completion, we so glad that our final year project and report were carried out and conducted smoothly and successfully.

First and foremost, we would like to convey our highest appreciation and special gratitude to our respected supervisor, **Dr. Mai Ramadan** for her endless support, invaluable guidance and supervision throughout completion of our final year project. we would also like to express our appreciation to **Eng. Atrab Ahmed**, co-supervisor for her fully guidance and help. thank you very much for your support.

Apart from that, we would like to thank our entire friends for sharing knowledge, information and helping us in making this project a success.

To our beloved family, we want to give them our deepest love and gratitude for being very supportive and also for their inspiration and encouragement during our studies in this University

Last but not least, we hope this project will give a beneficial for this field of study especially in conducting for the future study.

Thank you

Smart Cafe Team Members

Abstract

Many infectious diseases that are spread by droplets and droplets require social distancing measures. According to the World Health Organization, the best way to avoid contracting COVID-19 is to maintain strict social distance. In a densely populated area, enforcing social distance is difficult and people are often unable to maintain adequate space from their neighbors. Also, Viruses such as Covid-19 are transferrable through touch and contact. Driven by the need for Automatic-efficient and cost-effective social distancing and noun touchable dealing, this project is presented.

One of the main advantages for societies is the Internet of Things (IoT) and Mobile Applications. As a result, many different types of companies use IoT technology for a variety of goals, including business automation, business intelligence, and marketing. IoT and Mobile Applications have a lot of potential in the cafeterias industry. However, in the cafeterias industry, these technologies are not widely deployed.

In This Project, a touchless is the best solution to combat covid-19. once customer enter a café, he should sterilize his hands. Then, once connected with Wi-Fi Customer can scan the barcode to open the E_menu and select a drink needed. The Smart Machine begins to make the drink without touching through ESP8266 Wi-Fi module with Arduino and 3 liquid pumps. Then The Smart Robot will deliver the Order in a few minutes and take the feedback from the customer.

Keywords: Covid-19, Touchless, robot, barcode, Sterilize, SC3-19

Table of Content

Chapter 1

Introduction

1.1 Introduction to project

The rapid increase of Covid-19 pandemic, there is a need to focus on personal hygiene and social responsibility. There is a high risk of the covid-19 virus transmission through manual use of sanitizers in Cafés. The contagion is primarily attributed to droplets of salvia generated by coughing, sneezing or nasal discharge.

For instance, when an infected person touches the manual dispenser, there is a high probability of virus being transmitted to other healthy individuals. Hence personal hygiene and protection is paramount. This can be achieved through the automating dispenser (touchless), where there is no need for manual operation.

In term of personal hygiene and protection, we need a smart automatic solution using IOT technology to redress the community spread of the disease. Although ‘the person who deliver drinks to customers’ in this time is a risk. We need a robot to deliver drinks to customers. In this case we avoid interfere with customers in café. Although ‘the paper menu’ is a risk. We need Android application that customer checks a barcode then drinks menu appeared to choose from it. Although hand sanitizer is important to sanitize hands of customers in Café.

1.2 Problem's definition

1. Manual Drinks Machine

The use of normal dispenser becomes very danger; press the bottle trigger, the virus may spread from this bottle.

Figure 1.1: Manual Drinks Machine

2. Waiter

A Person who deliver drinks to customers and Interfere with them is become danger.

Figure 1.2: Waiter in Cafe

3. Paper menus

Customers come in, pick them up and leave them for the next patron. Cafés need to rethink their menus. The days of passing around a paper menu are over and needed to adapt to keep their customers safe.

Figure 1.3: Paper Menu

4. Manual sanitizer

Hand sanitizers are an increasingly popular method to keep hands cleaned but it’s danger; you trigger a bottle to sterilize hand.

Figure 1.4: Manual Sanitizer

1.3 Alternative Solutions

1. Touchless; from android application you can make your drink without touching a machine.

Figure 1.5: Smart Drinks Machine

2. No interference with customers; a robot can deliver drink to customer.

Figure 1.6: Robot deliver drink to customer

3. E_Menu

Figure 1.7: Scan Barcode

4. Automatic hand sanitizer to keep customer's hands cleaned

Figure 1.8: automatic hand sanitizer

Chapter 2
System analysis

2.1 Introduction

Usually, users are not able to clearly define their problems or requirements. They have a vague idea of what they want .so system analyst need to gather user requirements. After gathering requirements and analyzing them, problem statement must be stated clearly. Problem definition should unambiguously state what problems need to be solved.

2.2 Analysis model

Our project follows Waterfall model in which software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete.

Waterfall model is SDLC in which outcome of one phase acts as the input for the next phase sequentially.

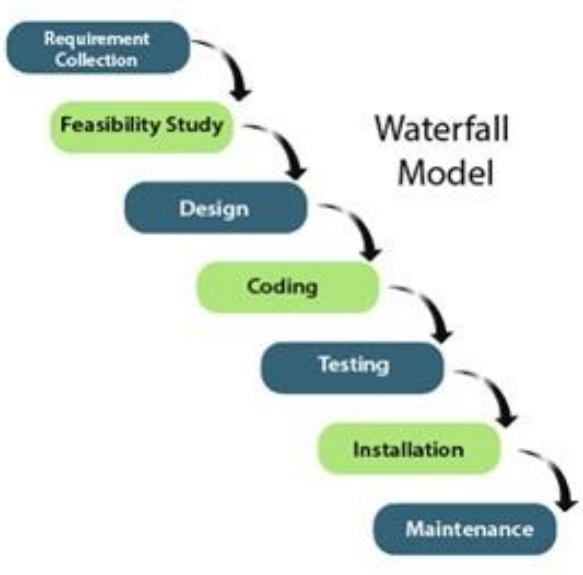

Figure 2.1: waterfall Model

Figure 2.2: Software development life Cycle

Stage 1: planning

This is the first phase in SDLC. At this stage, you make a feasibility study for the project and execute the project plan. The planning stage is actually the most important stage in project. the idea is revealed and discussed.

Stage 2: Analysis

In this phase system analyst collect information from user about the product. All details of the product must be discussed with the customer. The requirements gathered then analyzed for their validity and the possibility of incorporating them into the software.

Stage 3: Design

The system design phase begins with the completion of the requirements. Software product design is the activity of determining the characteristics and interfaces of the software product to meet needs of the customer.

Stage 4: Development

This stage is the process in which the product to be delivered to the customer is developed. Quality coding is very important. Good code is a simple code that is easy to read and maintain. According to the principle “keep it simple”, you must have a certain coding quality standard.

Stage 5: Testing& integration

The testing phase is an essential part of the development process. It is the stage where the developed software is tested whether it meets the expected requirements. How it reacts to correct and incorrect data entries. The performance of the software is measured and unexpected errors and sending to development phase to be eliminated.

Stage 6: Maintenance

After the testing phase is completed, a usable and deliverable version of software is created.in addition; the user guide document is prepared to user, in this phase. You can add new features to the product, resolving errors and monitoring the performance of the software. Based on the feedback from the users, new improvements can be made and existing problems can be solved.

2.3 Study of the system

2.3.1 GUI’S

SCCC-19 has Android application from which customer scan a barcode and select drink needed. Once selecting a drink, Order page notified with this order. Once trigger a drink, the machine begins to make this drink.

2.3.2 Entities Involved in the Project

1. **Customer**, The person who come in Café and scanning a barcode to select a drink needed.

2. **Admin**, The person who notified with orders and begins to make a drink by clicking a trigger.

3. **Waiter robot**, the robot that takes a drink and delivers it to customer.

2.4 Software Requirement Specification (SRS)

2.4.1 Introduction to SRS

SC3-19 deals with four problems include touch machines, interfere with customers in café and hands to be cleaned and paper menu to ensure safety of customers.

2.4.1.1 purpose

The main purpose of this document is personal hygiene and social responsibility to combat covid-19 virus in Café.

2.4.1.2 Scope

The key objectives are as follow:

- From android application you can make your drink without touching a machine using Internet of things (IOT).
- A robot can deliver drink to customer.
- Customer scan barcode and select drink needed.
- Automatic hand sanitizer to keep customer's hands cleaned.

2.4.2 External specific requirements:

2.4.2.1 Hardware Components

2.4.2.1.1 Smart Drinks Machine

- ESP8266 Wi-Fi Module

The NodeMCU ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (microcontroller unit) capability produced by the Shanghai-based Chinese manufacturer, Espressif Systems.

The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

This serial Wi-Fi module adds wireless Ethernet connectivity to your microcontroller through serial TTL pins (Rx and Tx).

Features

- Wi-Fi communication through serial AT commands
- Baud rate:115200
- Operating voltage:1.7 to 3.6(3.3 is standard)
- Integrated low power 32-bit CPU could be used as application processor
- On board build it antenna

Figure 2.3: NodeMCU Esp8266 Wi-Fi Module

- 3 Liquid pump

This is high performance brushless DC pumps with long-life, small size, high efficiency, low noise and low power consumption and adopt high-performance stainless-steel shaft.

It is submersible and entirely waterproof (IP68) with flow rate of 4L/MIN. the axis is enclosed with static sealing, not dynamic, which can avoid leaking problems.

Can be used in many fields such as cooling system. Very easy to use, just connect red wire to 12V and black wire to ground.

Figure 2.4: Three Liquid pumps

- ## 4 channel Relay module

  a relay is an electrically operated switch that can be turned on or off, letting the current go through or not, and can be controlled with low voltages, like 5V provided by the Arduino pins.

  This relay module has four channels powered with 5V which is appropriate to use with an Arduino.

Figure 2.5 : 4channel Relay Module

- ## Jumper Wires

Figure 2.6: Jumper Wires

- Power supply

A power supply is an electrical device that supplies electric power. It converts AC into multiple DC voltages. A power supply gives three voltages. 12 volts represent by yellow wire, 5volt represents by red wire and 3.3 volt represents by orange wire. Which are used by the chips and components.

Figure 2.7: Power Supply

Circuit diagram

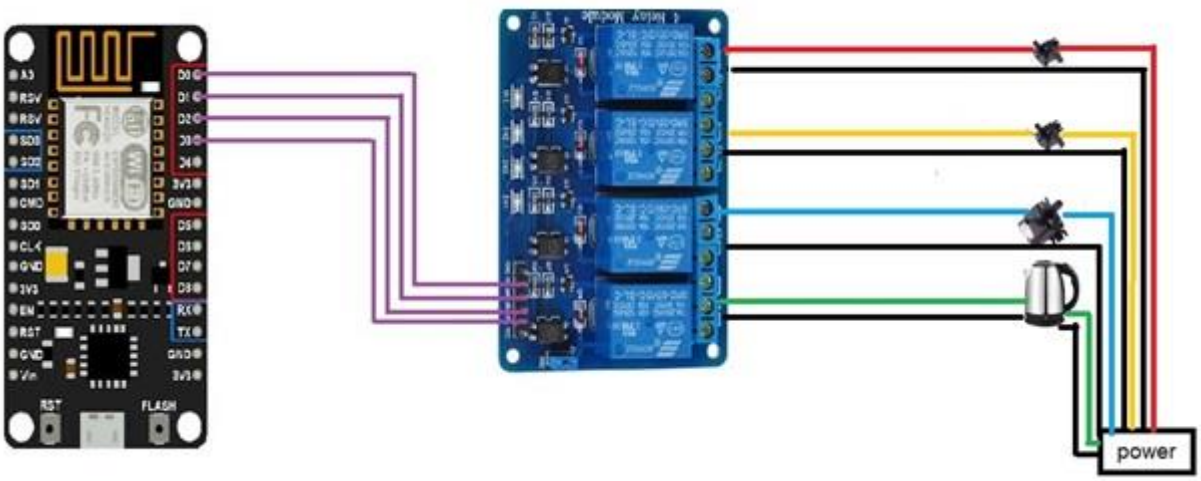

Figure 2.8: Touchless Drinks Machine Circuit

2.4.2.1.3 Automatic Hand Sanitizer

* IR Sensor

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver.

Figure 2.9: IR Sensor

*Arduino UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output pins that may be interfaced to various expansion boards and other circuits.

Figure 2.10: Arduino Uno

*Water Pump

DC pump with long-life, small size, high efficiency, low noise and low power consumption and adopt high-performance stainless-steel shaft.

Figure 2.11: Liquid Pump

*One channel Relay

Figure 2.12: One Channel Relay

Circuit Diagram

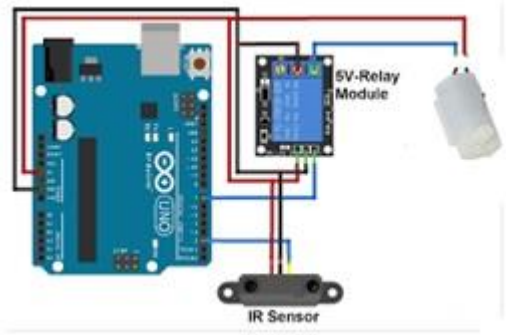

Figure 2.13: hand sanitizer Circuit

2.4.2.1.3 Waiter Robot

* RFID

RFID systems are composed of tags and a reader to read them. Each RFID tag has a processor and an antenna. Tags can be active and use a battery or passive and use power from the reader. Readers can be handheld or stationary. They vary in size and strength based on their purpose.

Figure 2.14: Radio Frequency Identification

DC motor:

(Direct current machine) refers to a rotating electric machine that can convert direct current electrical energy into mechanical energy or convert mechanical energy into direct current electrical energy. It is a motor that can convert between DC electrical energy and mechanical energy.

Figure 2.15: DC Motor

Line follower 3 channel:

For the move of Robot to specific place, and the use of line following robotic vehicle is transport the materials from one place to another place in the industries

Figure 2.16: Line Follower 3 Channel

Battery 12 V:

In order to gain the power to make the robot move

Figure 2.17: Battery 12 V

Keypad:

An exploited 16-button keypad can provide a useful human interface with the Arduino. It represents a simple way to deal with variety options. A combination of four rows and four columns in the keypad matrix can be helpful for the microcontroller to manage the button states.

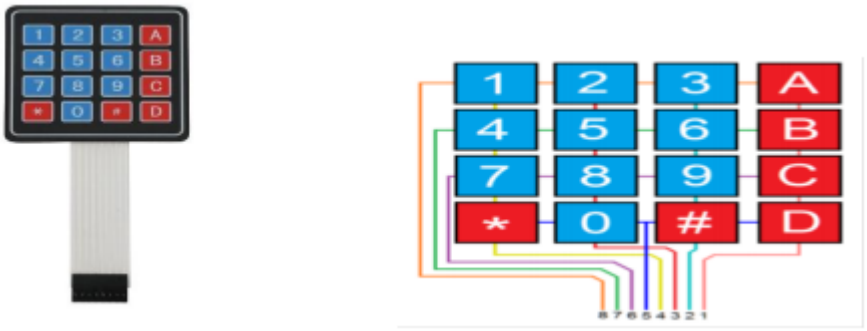

Figure 2.18: Keypad Map

Motor 12 V:

DC motor converts direct current electrical energy into mechanical energy. At the most basic level, DC motors work well in robotics because they allow the robot to be battery powered, which offers great advantages for a variety of robotic applications, particularly mobile and collaborative robots.

Figure 2.19: Motor 12 V

SD Card Module

The SD and micro SD card modules allow you to communicate with the memory card and write or read the information on them. The module interfaces in the SPI protocol.

Figure 2.20: SD Card Module

Voice Recognition Module

Is a compact easy-control speaking **recognition** board? It is a speaker dependent **module**

Figure 2.21: Voice recognition Module

Speaker

Figure 2.22: Speaker

Circuit Diagram

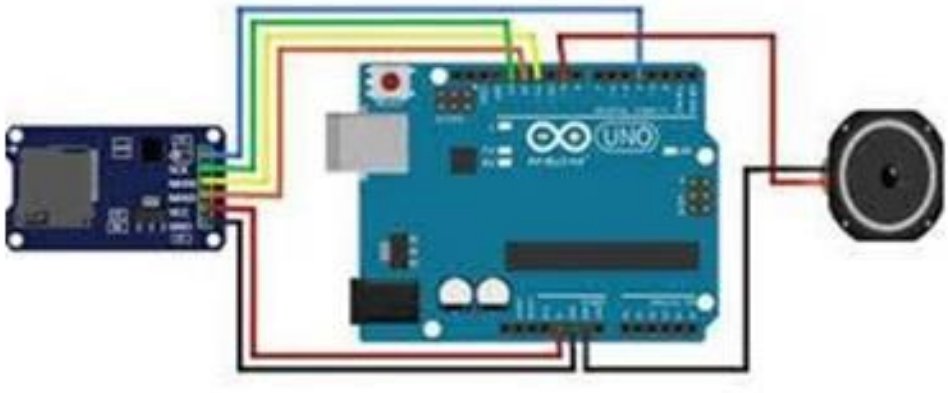

Figure 2.23: Voice Recognition Circuit

2.5 Performance requirements

The system must be interactive and the delays involved must be less. So in every action-response of the system, there are not immediate delays. In case of Scanning a barcode, the menu must appear much below 2 seconds.

2.6 Proposed System

2.6.1 Functional Features

The entire Scope has been classified into four parts Touchless drinks Machine via Cloud, Waiter Robot, scanning a barcode and sterilize a hand. The proposed Software is reliable to customers to order drinks and admin receive these orders and trigger them to be done.

2.6.2 Input and output

The main inputs, outputs and major functions of the system are as follows

Inputs:

- Customer Sterilize a hand.
- Customer Scan barcode and order a drink.

Outputs

- Admin receiver orders.
- Admin make a drink by trigger a drink.
- Waiter robot delivers a drink to customer.

Chapter 3
SC3-19 Design

3.1 Introduction

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages. Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

Software analysis and design includes all activities, which help the transformation of requirement specification into implementation. Requirement specifications specify all functional and non-functional expectations from the software. These requirement specifications come in the shape of human readable and understandable documents, to which a computer has nothing to do.

Software analysis and design is the intermediate stage, which helps human readable requirements to be transformed into actual code.

3.2 DFD - Data Flow Diagrams

Data Flow Diagram (DFD) is a graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow, and stored data. The DFD does not mention anything about how data flows through the system. There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. It does not contain any control or branch elements.

3.2.1 Types of DFD Data Flow Diagrams

Are either Logical or Physical.

- **Logical DFD -** This type of DFD concentrates on the system process, and flow of data in the system.

- **Physical DFD -** This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation

3.2.2 DFD Components

DFD can represent source, destination, storage, and flow of data using the following set of components_

Figure 3.1: DFD Components

- **Entities** - Entities are sources and destinations of information data. Entities are represented by rectangles with their respective names.
- **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.

- **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination**.**

3.2.3 Constructing a DFD

Several rules are used in drawing DFD'S:

1. Each process should have at least one input and an output.
2. Each data store should have at least one data flow in and one data flow out.
3. Data stored in a system must go through a process.
4. all processes in a DFD go to another process or a data store.
5. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.

3.2.4 Features of DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.

2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.

3. The sequence of events is not brought out on the DFD.

3.2.5 Rules governing the DFD's

3.2.5.1 Process

- No process can have only outputs.

- No process can have only inputs. If an object has only inputs than it must be a sink.

- A process has a verb phrase label.

3.2.5.2 Data store

- Data cannot move directly from one data store to another data store, a process must move data.

- A data store has a noun phrase label.

3.2.5.3 Source or sink

- Data cannot move direly from a source to sink it must be moved by a process.

- A source and /or sink has a noun phrase land.

3.2.5.4 Data flow

1. A Data Flow has only one direction of flow between symbols.
2. A data flow cannot go directly back to the same process.
3. A Data flow to a data store means update (delete or change).
4. A data Flow from a data store means retrieve or use.

3.2.6 DFD Diagrams

Customer DFD

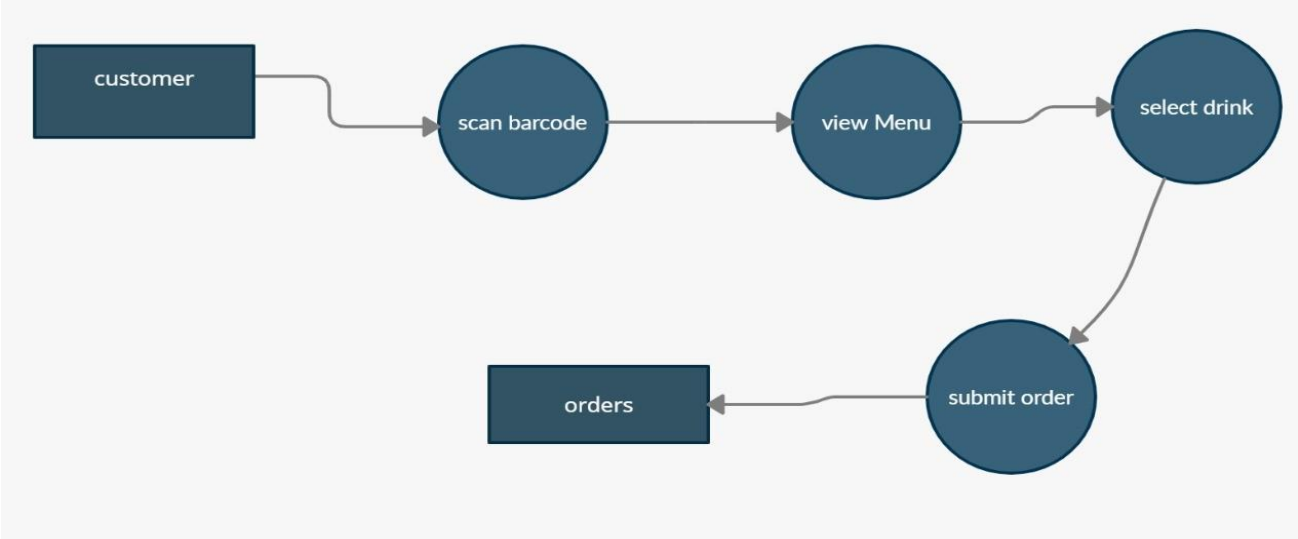

Figure 3.2: Customer DFD

Admin, Waiter robot DFD

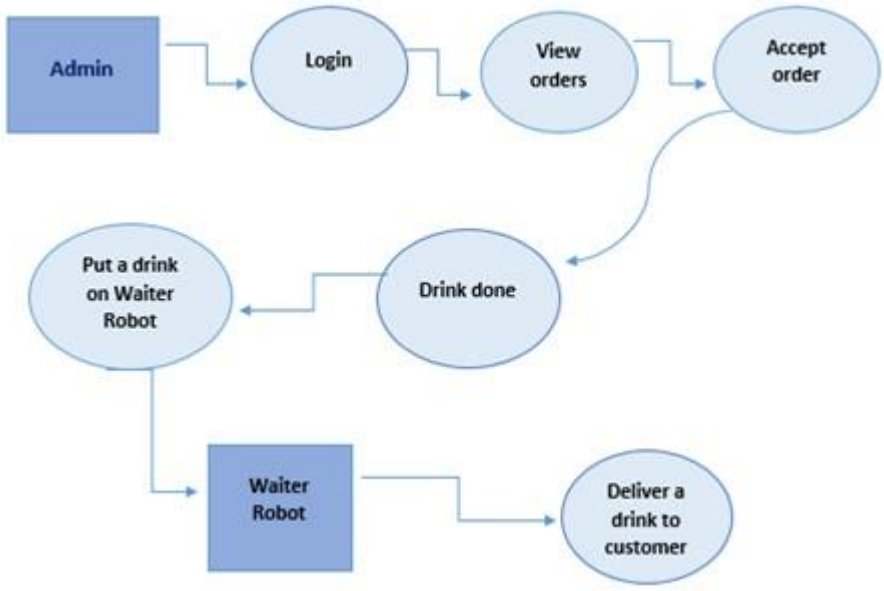

Figure 3.3: Admin, Waiter DFD

3.3 Unified Modeling Language (UML)

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

The Unified Modeling Language (UML) is a set of notations and conventions used to describe and model an application. The UML is intended to be a universal language for modeling systems, meaning that it can express models of many different kinds and purposes, just as a programming language or a natural language can be used in different ways.

The use of visual notation to represent or model a problem can provide us several benefits relating to clarity, familiarity, maintenance, and simplification.

UML is a language used to:

- Specify the software system and help building unambiguous and complete models.

- Construct the models of the software system that can directly communicate with a variety of programming languages.

UML provides different kinds of models for modeling the system. These diagrams are:

- **interaction models**, where you model the interactions between a system and its environment or between the components of a system.

- **Structural models**, where you model the structure of the data that is processed by the system

- **behavior models**, where you model the dynamic behavior of the system and how it responds to events.

3.3.1 Interaction models

*** Use Case Diagram**

Which is mostly used to model interactions between a system and external actors (users).

***Sequence Diagram**

Which are used to model interactions between System Components.

3.3.2 Structural models

*** Class Diagram**

Class Diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.

3.3.3 Behavior models

*** Activity diagrams:**

These diagrams describe the behavior of a class in response to internal processing rather than external events. Activity diagrams describe the processing activities with in a class.

3.4 UML Diagrams

3.4.1 Use Case Diagram

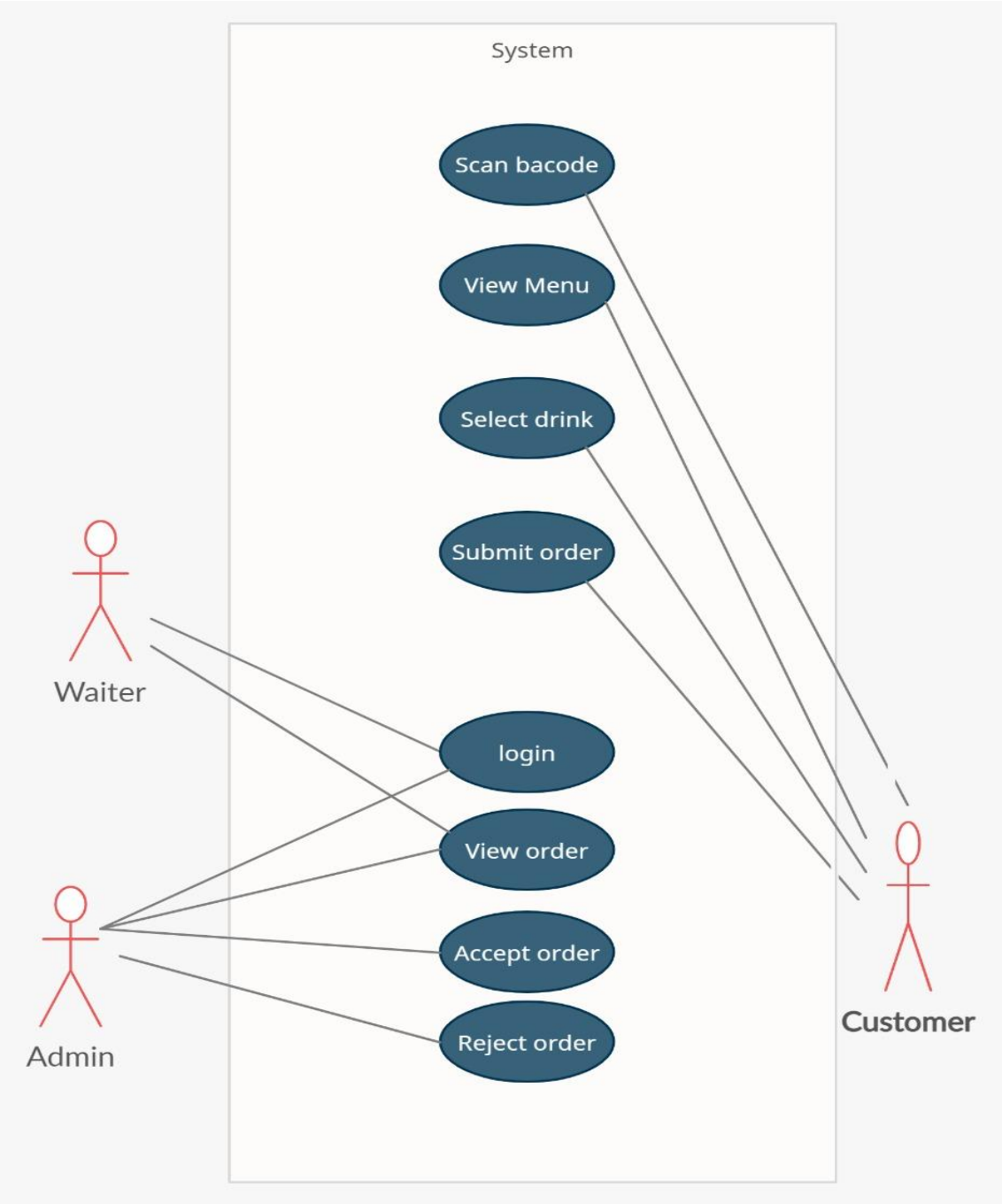

Figure 3.4: Use Case Diagram

3.4.2 Sequence Diagram

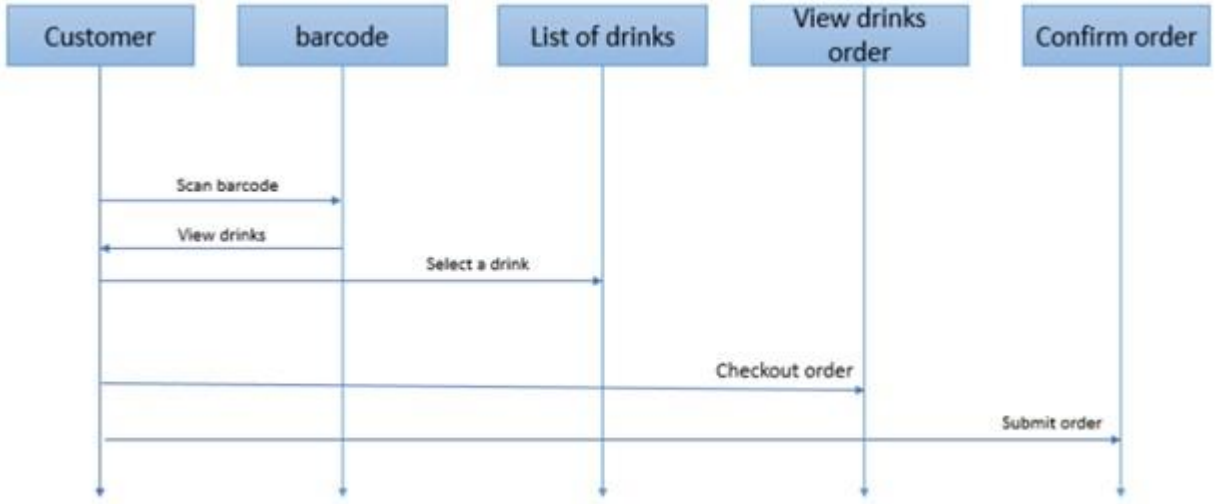

Figure 3.5: Customer Sequence Diagram

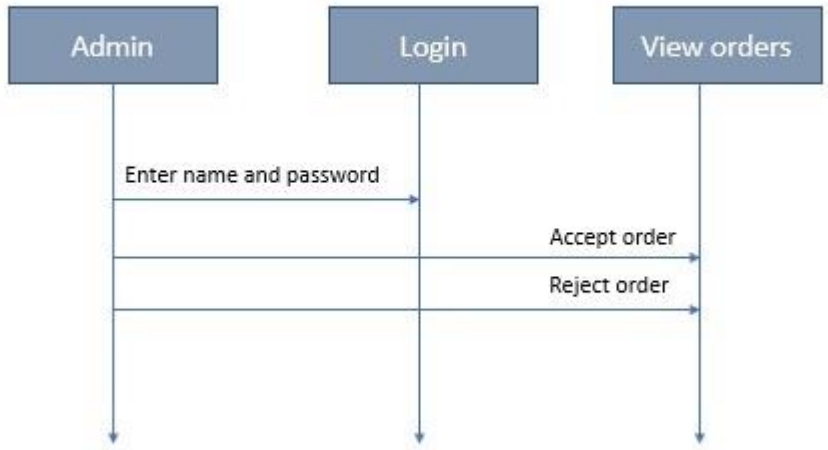

Figure 3.6: Admin Sequence Diagram

3.4.3 Activity Diagram

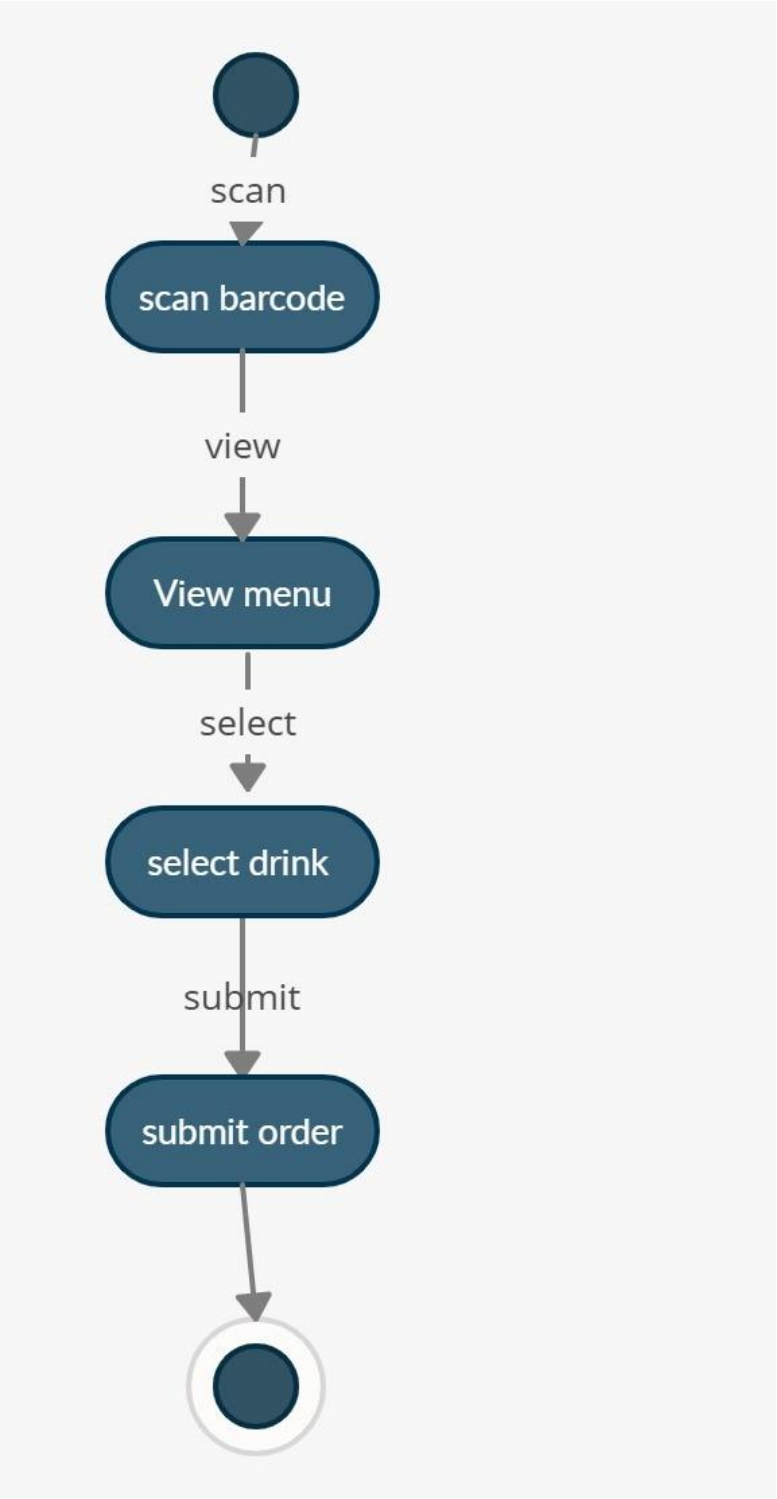

Figure 3.7: Activity Diagram of Customer

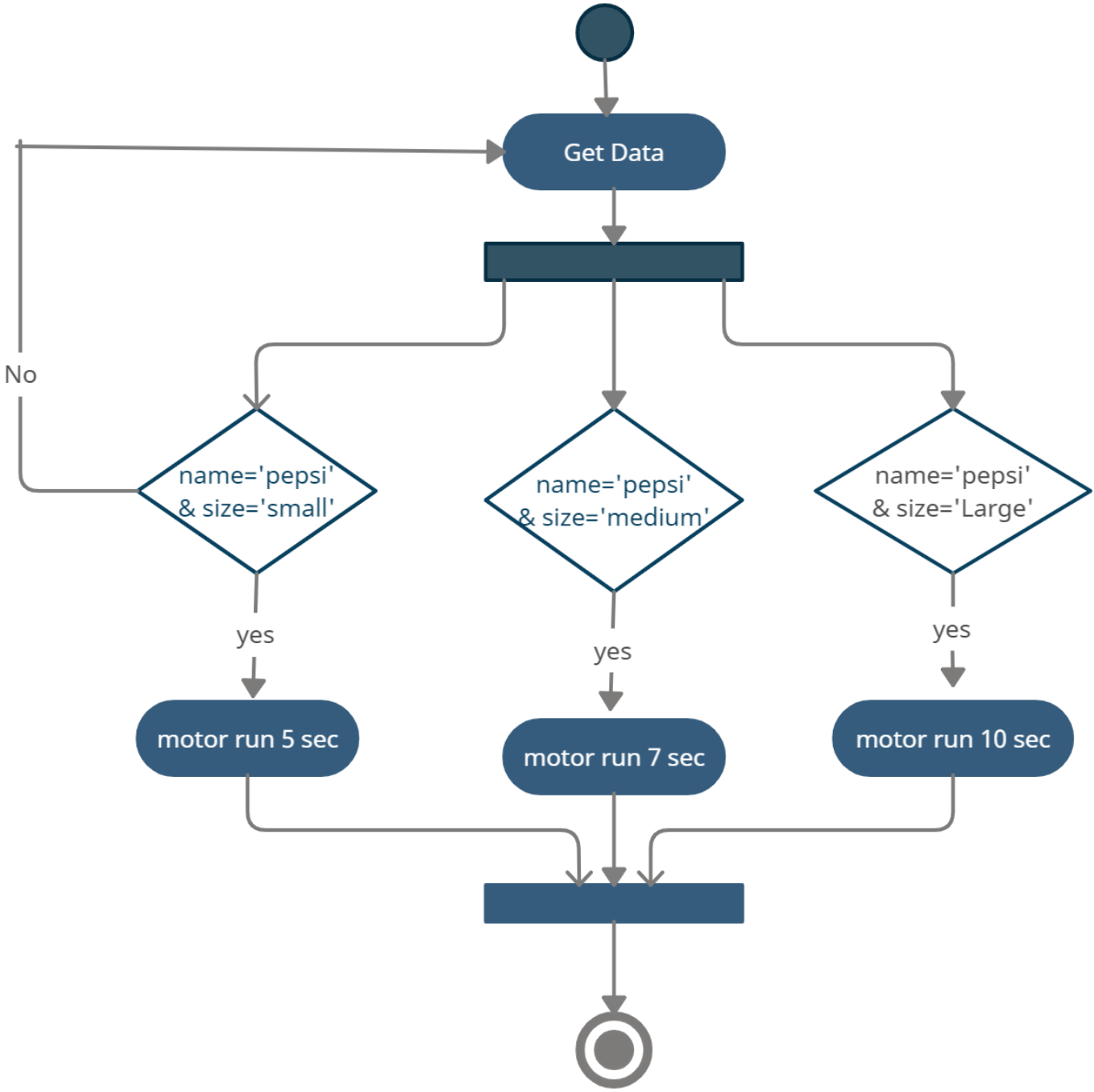

Figure 3.8: Smart Drinks Machine Activity Diagram

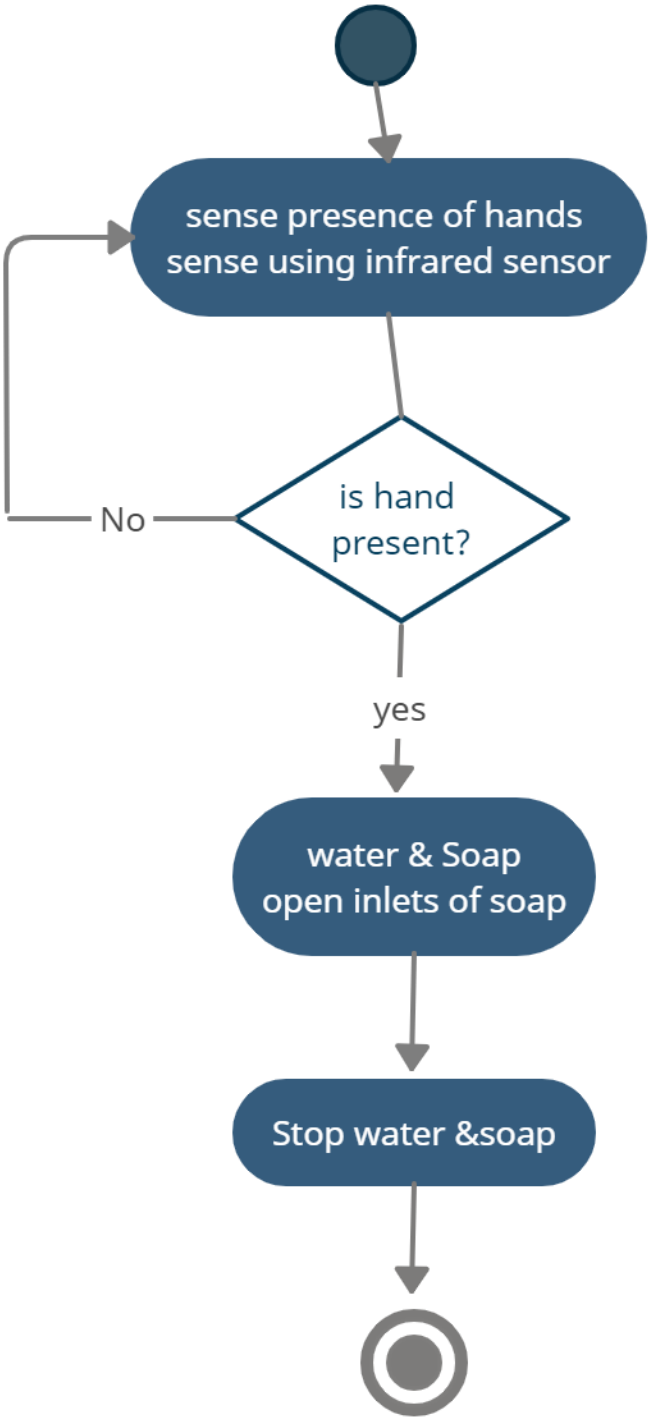

Figure 3.8: Activity Diagram of Hand sanitizer

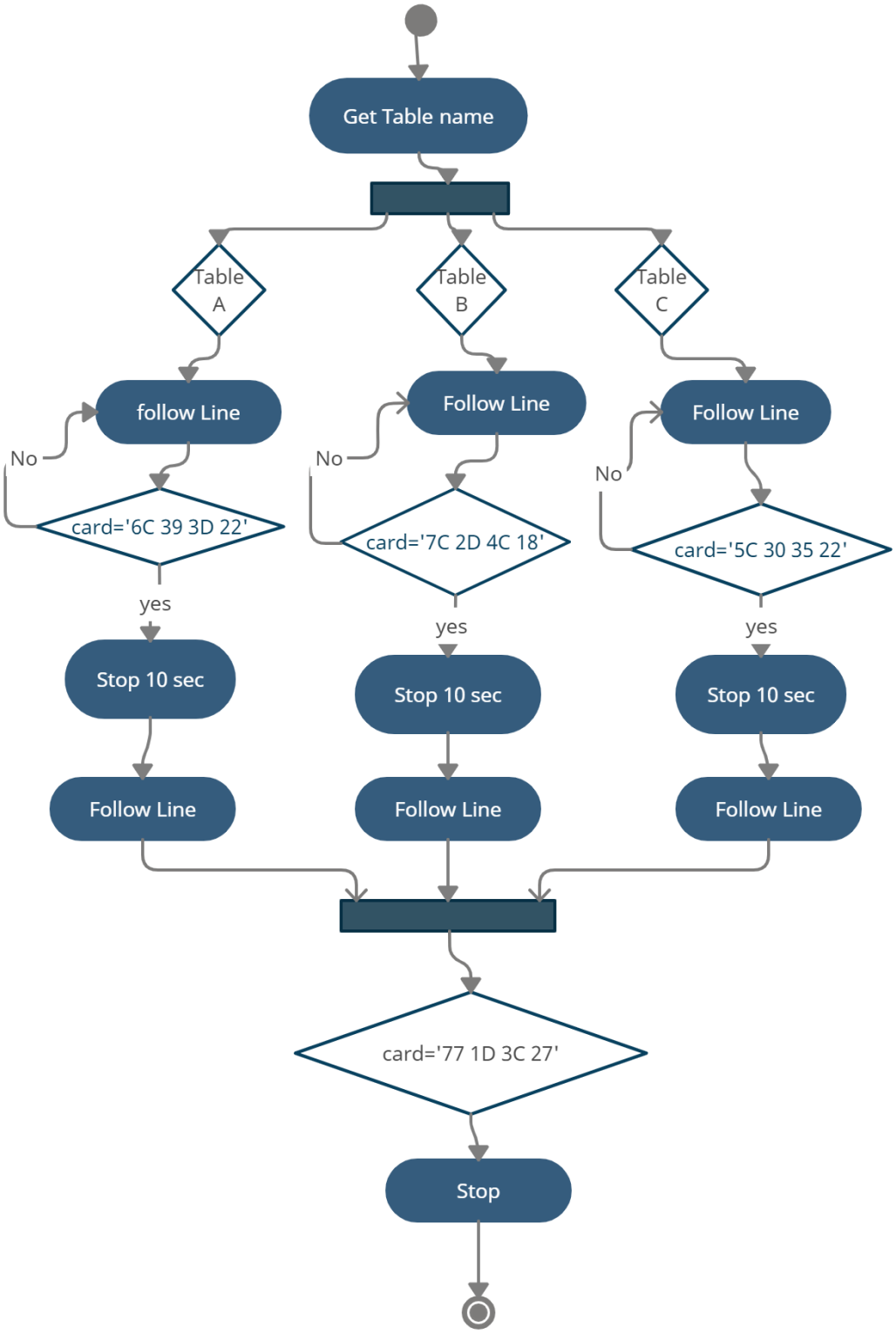

Figure 3.9: Waiter robot activity Diagram

3.5 Class Diagram

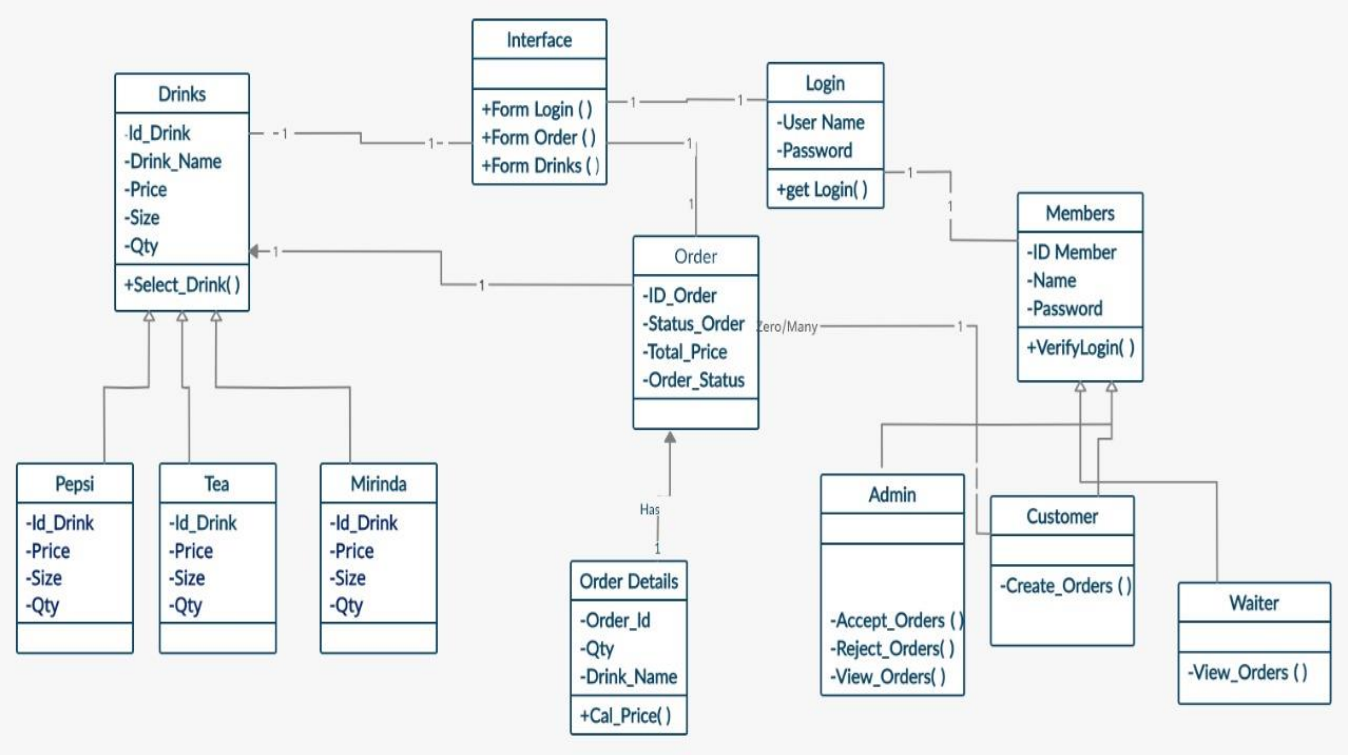

Figure 3.10 Class Diagram of E_Menu App

3.4.5 ER Diagram

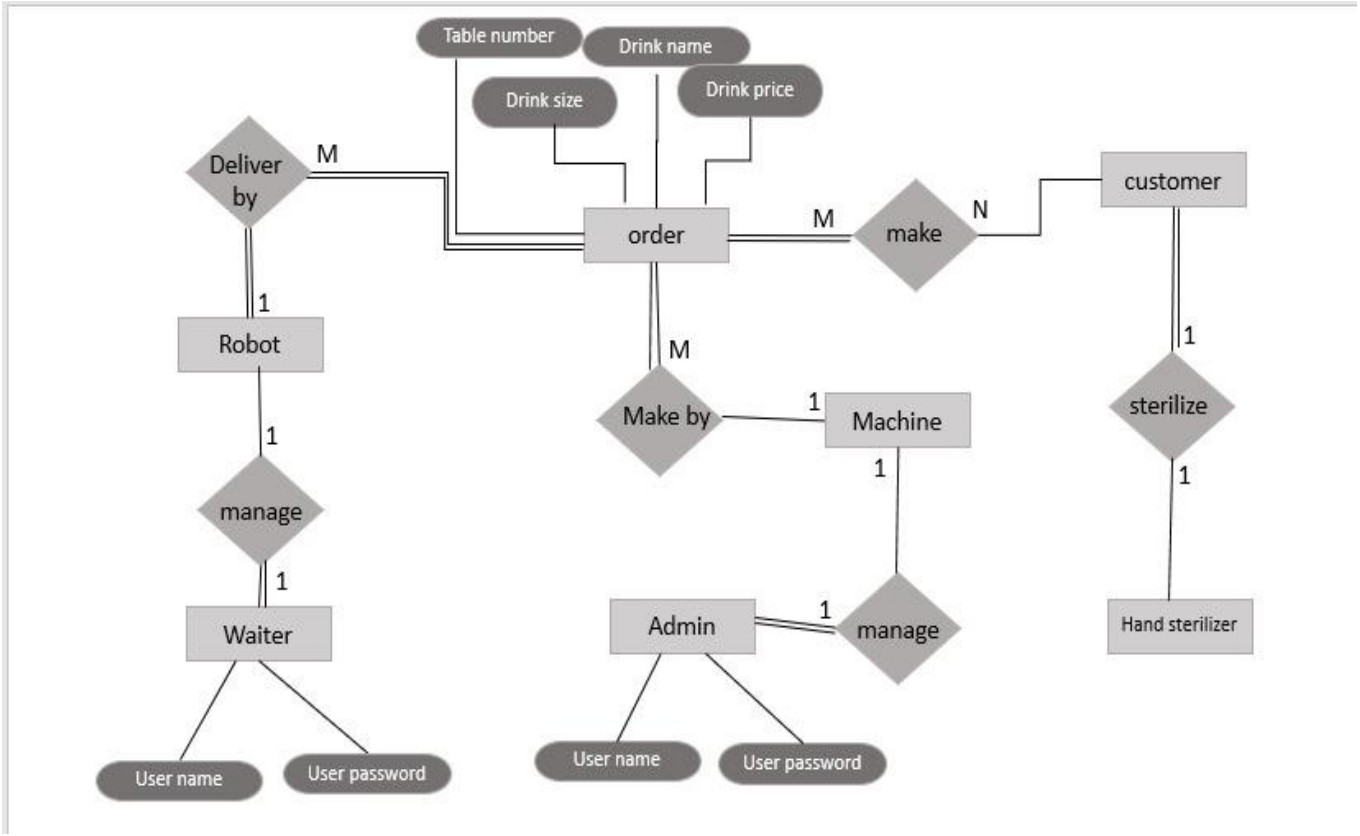

Figure 3.11 ER Diagram

3.5 Data Dictionary

3.5.1 Smart Drinks Machine

Chapter 4

SC3-19 Software

4.1 Arduino Software

Figure 4.1: Arduino IDE Components

4.1.1 IDE

Arduino IDE is an open-Source software that is mainly used for writing and compiling the code into the Arduino Module. It is an official Arduino Software making compilation too easy. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus.

4.1.2 Sketch

A sketch is the name that Arduino uses for a program. It is the unit of code that is uploaded to and run on Arduino board.

Verify

The verify button will check through your code line by line, ensuring there are no programming errors. It will be handy for us to begin troubleshooting issues with sketches using this button.

Upload

The upload button will send you verified code to your Arduino. You will see an uploading progress bar appear in your IDE and a success message when upload is complete.

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Coding Area

The Coding Area is where the real magic happens! This is where you will type your instructions and comments.

Debugging Console

The debugging console will provide you with the line and character number that is causing your program to malfunction. It will also output a program compilation was the successful message.

4.1.3 Sketchbook

The default location where Arduino sketches you write will be saved is called sketchbook, the folder where Arduino IDE stores sketches. This folder is automatically created by the IDE when you install it. You can change the location of the sketchbook location from with the Preferences dialog.

4.1.4 Bootloader

The bootloader is a little program that runs when turn the Arduino on, or press the rest button. Its main function is to wait for Arduino software on your computer to send it a new program for the Arduino. It allows you to upload code without using any additional hardware.

4.1.5 Libraries

The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from sketch > import Library.

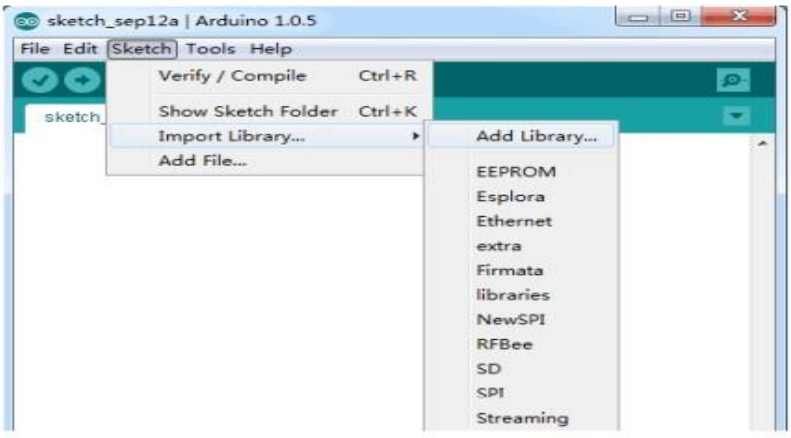

Figure 4.2: add Libraries in Arduino

4.1.6 Serial Monitor

The serial Monitor button will allow you to see data that is transferred to and from the Arduino board and in some cases allow us to instruct the Arduino in real-time. This is essential in project, so we will cover the ins and outs of the serial monitor.

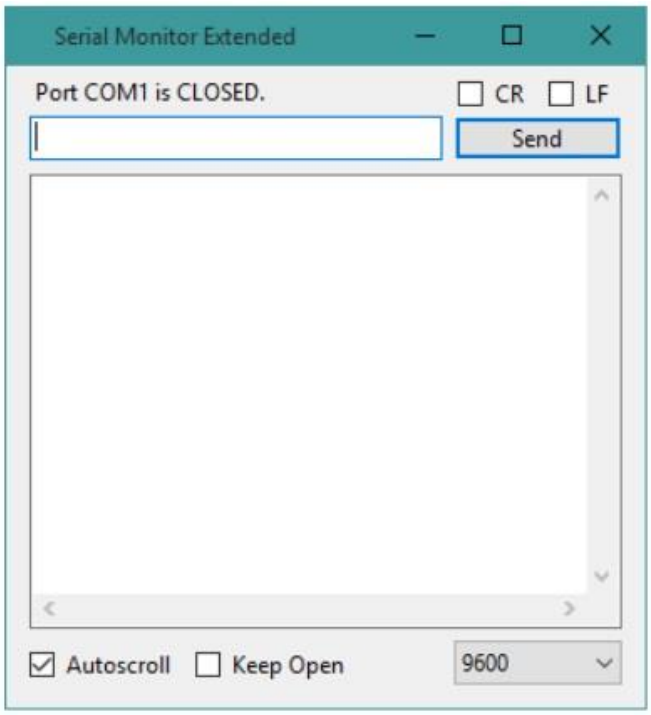

Figure4.3: Serial Monitor

4.1.7 Boards

The core is available in the Arduino IDE software under the menu Tools > Boards > Boards Manager. Once Boards Manager is open, you can just search the name of your board and install it. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection, you'll want to check it before burning the bootloader.

4.2 Smart Café – ANDROID MOBILE APPLICATION

4.2.1 Introduction to Programming Languages, IDE'S, Tools and Technologies used for This Implementation: -

1. Java:-

As the project is developing an Android Application, the default programming language is Java. All Android applications are built using Java in Android Studio or Eclipse or both. Java is a popular and widely used language throughout the world. Java is one of the powerful programming languages like C, C++. Developed by Sun Microsystems which has many powerful features as described below. After the development of C, C++, Java has come into evolution by addressing their drawbacks .It is one of the open source projects [5] that could be easily installed in our machine. The language is also easy to learn, understand and implement. Java is used in various kinds of applications like Web, Desktop, Mobile, and Big Data. Many powerful features are supported by Java including various libraries, application services, graphics library for 2D/3D applications. The language is flexible enough to maintain code complexity, test, implementation, integration and support. Apart from these, there are other key features which make Java more special. It is object oriented programming language, one of the important hierarchies in the programming languages which is used to implement real time applications, it provides for code reusability, it has a platform independence feature including any virtual machines(Write Once Read Everywhere), as in no need to write the 20 code for different OS as the Java Compliers convert the java source files to byte code and this could be interpreted by any machine and the actual code is compiled irrespective of any machine, OS. It is more secured as the compilers are designed efficiently to figure out any kind of errors.

2. IDE's, Tools and Technologies:-

2.1 Android Studio

Android Studio is exclusively designed for developing Android applications. It consists of all Android SDK tools to design, develop, maintain, test, debug and publish our app. The IDE is designed very efficiently which makes the developer's job easy. It also supports the IntelliJ IDE, the main idea behind this IDE is that it automatically senses the variables, methods, classes, built-in functions or it could be anything else when we press the first letter of it. Say, suppose we declared few variables or methods that starts with an 'S', it automatically senses everything that starts with an 'S' and makes suggestions. It also supports Git as a version control system to maintain the app changes and push them into github. All java files, layout files (for design) are integrated into a single project easily. After the completion of project, the whole application could be put as an .APK (Android Package) file, in which we can run that APK file in any device and use the application. Other main tools include Android SDK, ADB, and Cradle Build.

Figure4.4: Android Studio IDE

2. Android Software Development Kit (SDK):-

One of the main tools used in developing android applications, as it packages many core features into one SDK and it can be used in the application easily. This helps us to avoid writing lot of code, and building applications faster.

3. Android Debug Bridge (ADB):

Android SDK uses ADB tool as a connection device which allows us to connect the Android Devices or Emulator with the machine via USB. After developing or while developing applications, we can connect with the device to check how the application runs. Later, we can debug and run the applications.

4. Gradle Build:

Gradle Scripts are the recent feature that is added to Android Studio. It is basically an automated build system which is used to automate the various phases involved in designing an application that includes design, development, test, debug, and publish. We need to configure the project and modules by mentioning all the supported jar files, SDK's, version name, level, compiled SDK version, build tools version. to ensure that the developed app is compatible with the testing device/emulator. Gradle is also similar to Ant and Maven which helps in maintaining java projects (repositories).

5. Android Device Monitor:

If we want to access all the hidden files that are generated when we run the application, we can use the monitor. We can select any project and explore the files that are related to that project. But, as they are hidden files, we need root permissions to access them. Suppose, if we run the app in device, we need to root the device and run commands in adb shell to get permissions.

6. SDK Manager:

It is one of the main tools to maintain the updates of all the installed components required to run the project. It also notifies us when the project is not compatible with device or any other compatibility issues and to download any component that is required.

7. AVD Manager:

It is used to create virtual devices of any desired API level to support higher level SDK's incase our device does not support. Using emulators to test the application is difficult as it might be little slower when compared to real device.

8. Firebase:-

- Firebase Real time Database is a cloud-hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON format and any changes in the data, reflects immediately by performing sync across all the platforms & devices.

- Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

Figure4.5: Firebase

4.2.2 Security and Permissions in Android

Security notions in Android are quite high. Whenever a new Android Application is created, a unique user and group ID. This makes the maintenance of the application in an easier way to avoid any security or privacy issues. As the application is created uniquely, it becomes private and no one can access other's applications.

Permissions are another important concept which is included in AndroidManifest.XML configuration file. This is required if the application wants to access the external features. For ex, if the application wants to access the Internet, Camera or it could be any feature, it requires permissions. It is included within the tags as it is an XML file. Permissions are automatically created for the basic applications at the time when we create the application. If the app uses higher level API or SDK we must explicitly mention the permissions inside uses-permissions tag to access the features or component.

4.2.3 User Interface Representation

To make the application interactive, different controls have been used and designed using the layout file. Following are the important controls that are designed and used in this application:

- **Text View**: The text view component belongs to the view group as a part of GUI. It displays the text or content view of any activity to the user and allows them to edit.
- **Edit Text**: This allows itself to be editable in the text box.

- **Button**: One of the important components in which the application needs. It is mainly associated with action when the user clicks it. We can represent the button using any text which holds the action class on it.
- **Image Button**: Suppose, if we want to have an image for the button which we have designed, we can include using this control by adding the source or path of the image file within the tags in the layout file.
- **List View**: This is a key component under the view group which helps in displaying the information about anything when we click the action button. It also allows us to scroll through the screen and have a look about the information displayed. Using the list adapter, the content is pulled from the database.
- **Checkbox**: It is the control component which allows us to use or make use of the function by just clicking on the check box button. When we include check box widget in the application.

Chapter 5
SC3-19 Implementation

5.1 Introduction to system implementation

Implementation is the process of defining how the information system should be built. The process of installing and maintaining a new system and making sure it operates correctly. Involves handling System to ensuring that the information system meets quality standard.

5.2 Implementation of SC3-19 Android App

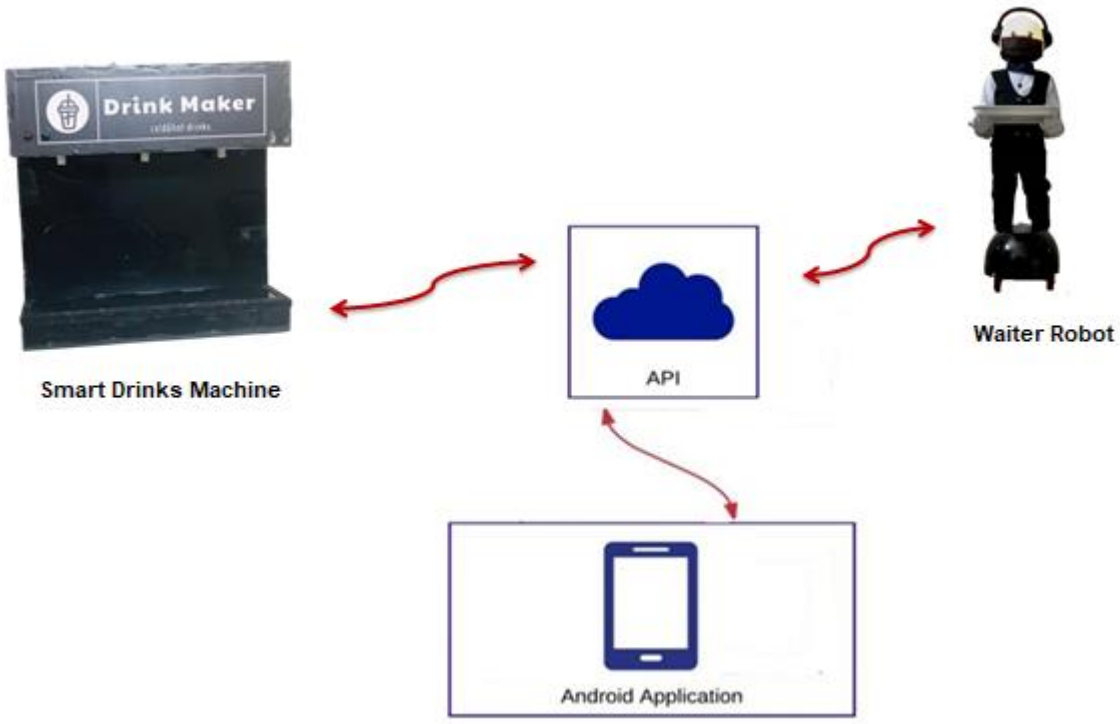

Figure 5.1: Implementation of SC3-19 android app

5.2.1 Smart Drinks Machine Android App Steps

- **Step1: Opening application**

The home page contains two buttons:

- the first one for the customer
  - If the user wants to use the Smart Café, the user must download the application from the store, install it, choose user button.
- The second for the admin who receives orders from the customer

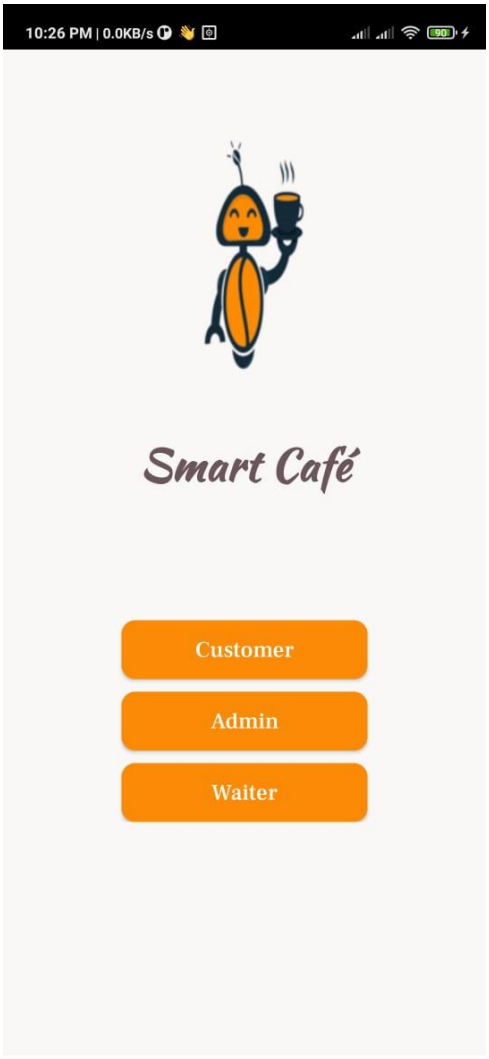

Figure 5.2: Home page

- **Step2: Scanning Barcode**

  This screen will appear for the customer after the home page , customer should scan the bar code which is on the table to view drinks

Figure 5.3: Scan Barcode

Step3: List of drinks

After scanning barcode, this screen will appear which containing drinks. Customers can choose any drink they want

Figure 5.4: List of drinks

In this project, we have used library to scan the barcode on items and it supports many kinds of barcode formats- UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Code 128, QR Code, Coda bar. To use this scanning feature in our project, we should jar files. This feature helps people by allowing them to scan the barcode that is available on the item. Once the user scans the item, they can see all the information about the item like barcode number, name, quantity, price, net price,

serial number. Users can scan any number of items they wish and keep adding to their physical shopping cart as shown in the Figure 5. Later, they can request any item they want by checking the items in the cart. It also has the option to select quantity of the items and suppose if the drink is not available for any items, “drink not available” message is also shown.

- **Step4: Checkout and Payment**

  Checkout is made in an easy step to avoid hassle in this application. The user can just check in with the checkbox from the physical shopping cart.

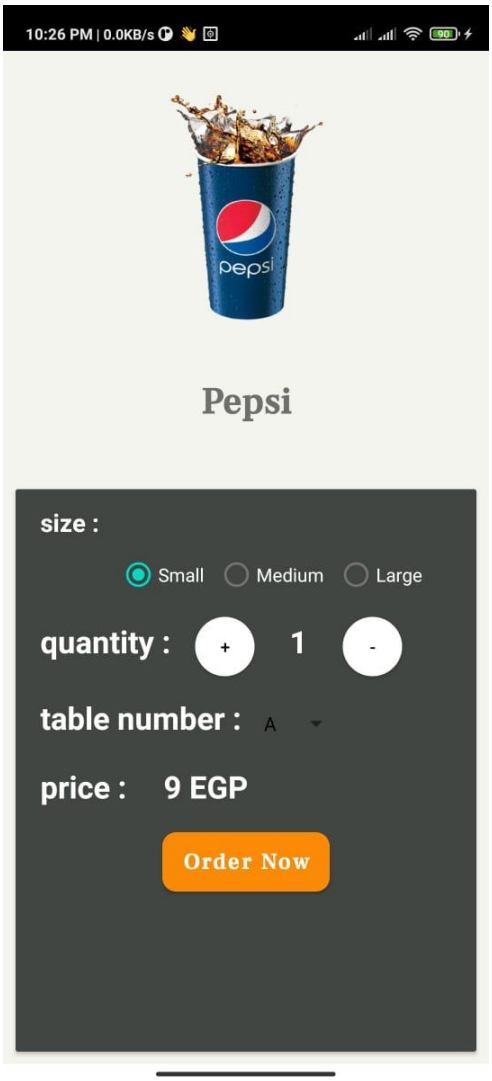

Figure 5.5: Order Screen

- **Step5: Submit Order**
  This is the last step for customers to confirm the order they choose.

Figure 5.6: Submit Order

- **Step6: Staff Login**

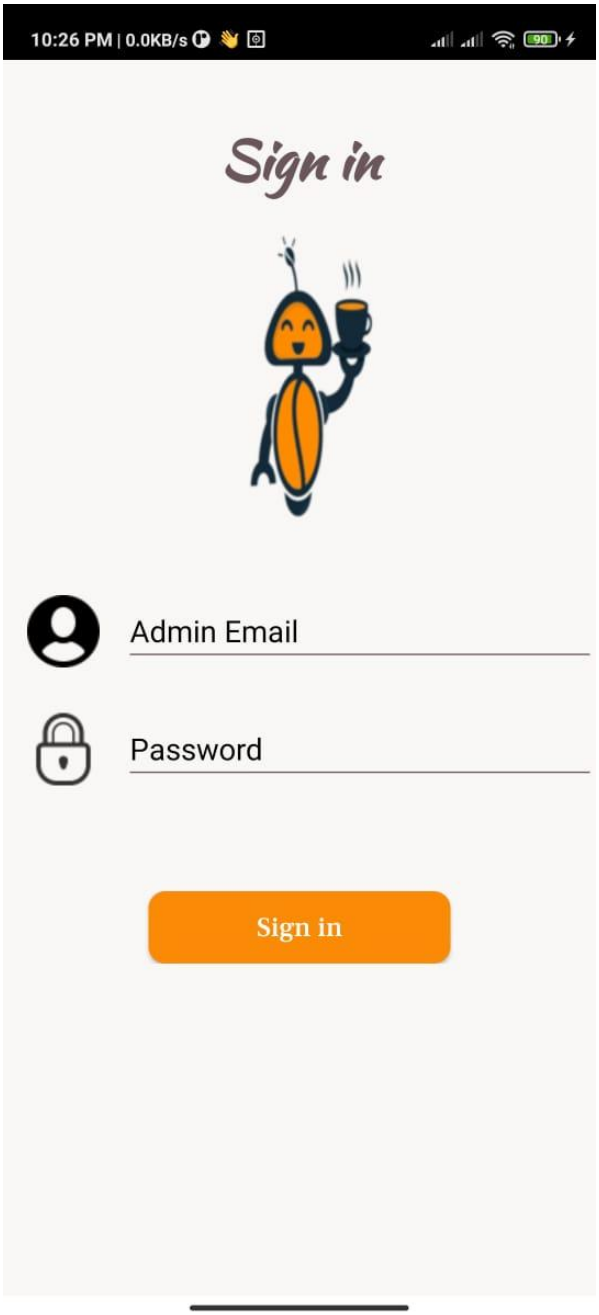

Figure 5.7: Staff Login

- **Step7: View Orders**

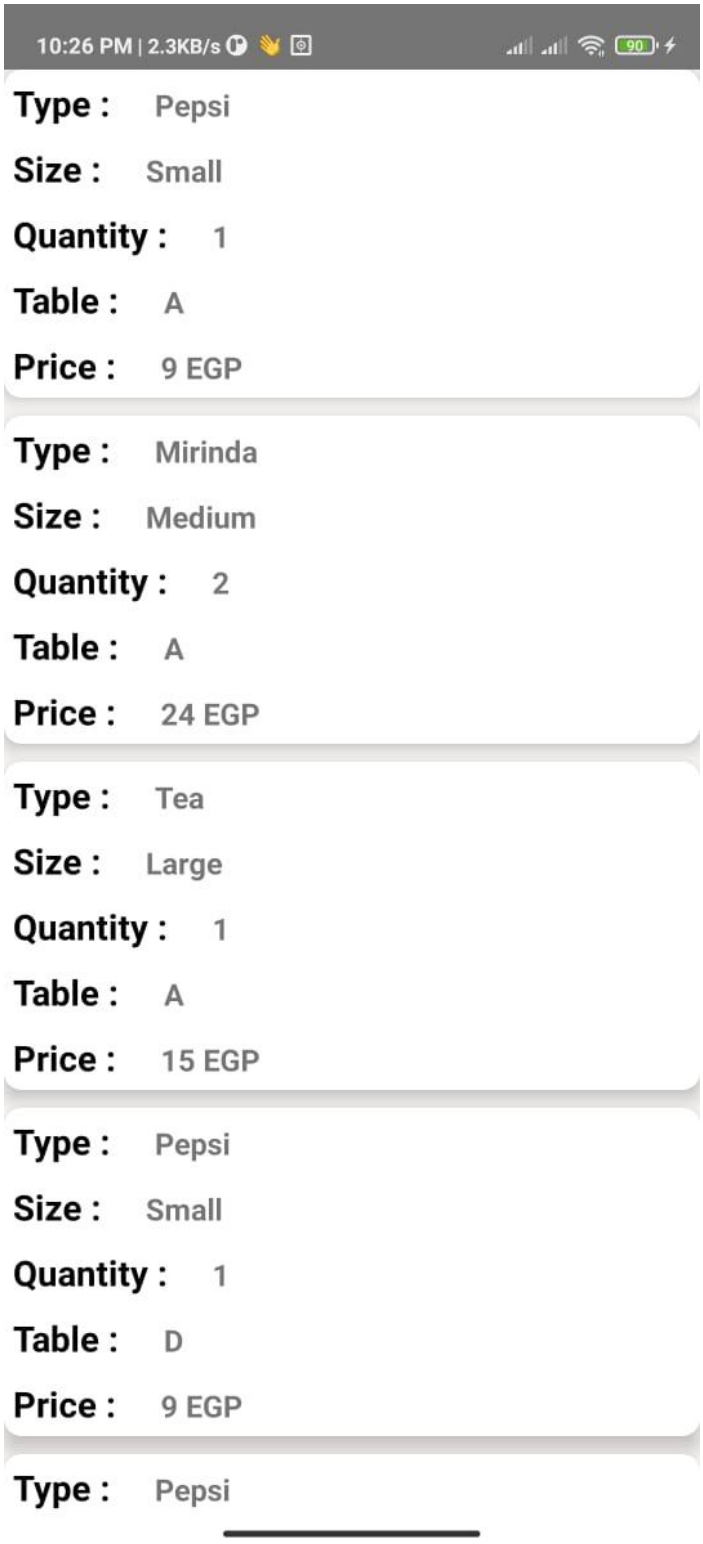

Figure 5.8: view order Screen

Step 8: Confirm Order

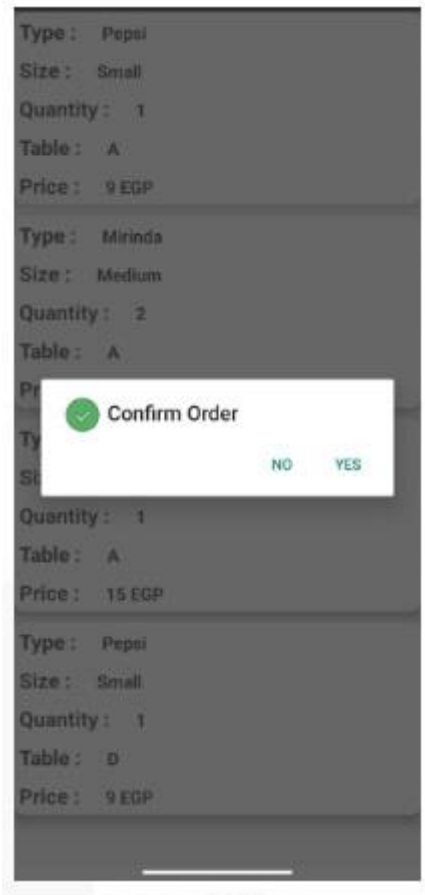

Figure 5.9: Confirm order Screen

5.3 Implementation of SC3-19 Hardware partition

5.3.1 Smart Drinks Machine

As shown in the figure Touchless Dispenser Consists of NodeMCU Esp8266 Wi-Fi Module , 4 Relay Channel , 3 water pump , Cattle and Power Supply.

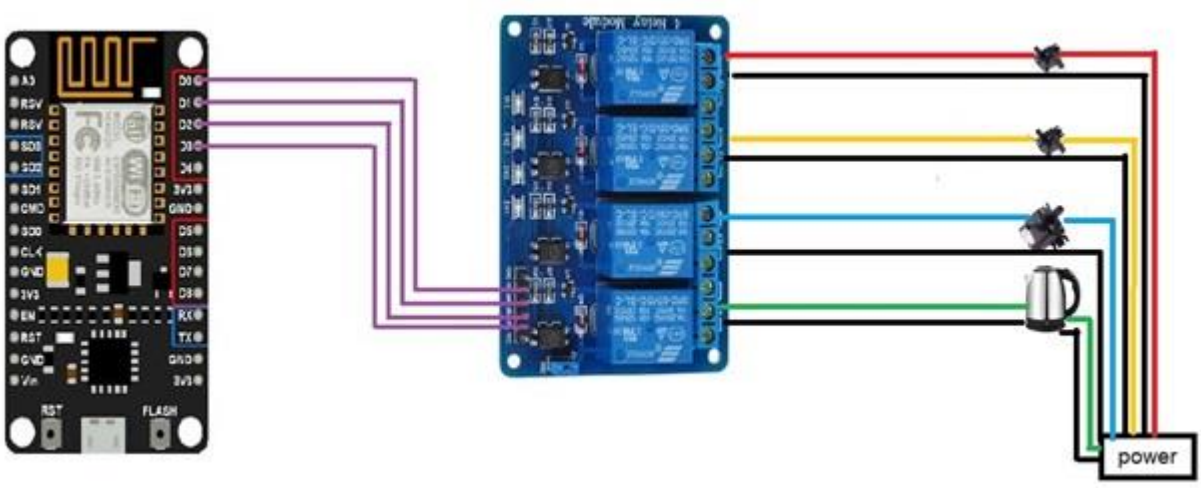

Figure 5.10: Smart Drinks Machine Circuit

Connection Pins

Motor 1	**D0**
Motor 2	D1
Motor 3	D2
Cattle	D3
Green Led	D4
Red Led	Power supply

Figure 5.11: Smart Drinks Machine

Figure 5.12: Smart Drinks Machine

5.3.2 Automatic hand sanitizer

AS shown in Figure, automatic hand sanitizer consists of Arduino UNO, IR Sensor, Water pump and Relay.

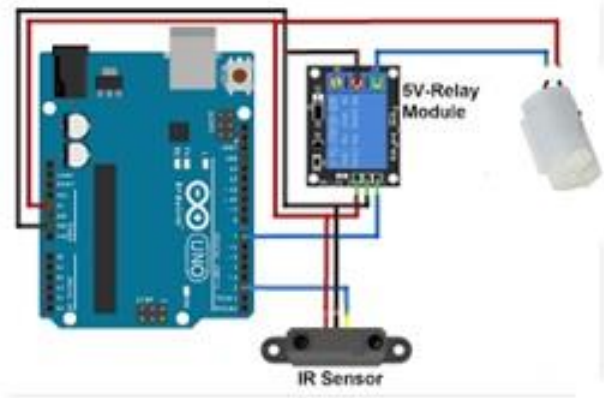

Figure5.13: Hand Sanitizer Circuit

Connection Pins

IR Sensor	2
Water pump	**7**

Figure5.14: Hand Sanitizer

Figure5.15: Hand Sanitizer

5.3.3 Waiter Robot

The Structure of waiter robot as shown in Figure 5.16 has one tray to put drinks on it, 4 wheels to move it , RFID to detect table , Line follower 3 channel to track a line , Arduino Mega , electrical and Mechanical Components .

Figure5.16: Physical Design of Waiter Robot

As Shown in Figure 5.17, 4 motors&wheels at bottom side , Line follower 3 channel at front side. From this configuration, the RFID Fixed at left side .

Figure5.17: Motors , Wheels and RFID

The robotics technology is replacing manual work at a fast pace throughout the world. In classical café, the customers face a lot of problems due to congestion at peak hours, unavailability of waiters and due to manual order processing. These shortcomings can be handled by using a restaurant automation system where, The desired order is also transmitted on wireless network to the kitchen via eMenu. The eMenu is based on Keypad . The customer places the order using eMenu. This order is sent to the kitchen and reception using communication network. The waiter robot then transfers drinks from the kitchen to the customer.

Figure 5.18: Waiter Robot deliver Drink to Customer

Figure5.19: waiter Robot in a Café

Chapter 6

SC3-19 Testing

6.1 Introduction

The testing phase is an essential part of the development process. It is the stage where the developed software is tested whether it meets the expected requirements. How it reacts to correct and incorrect data entries. The performance of the software is measured and unexpected errors and sending to development phase to be eliminated.

6.2 Features to be tested

This project should satisfy some features. Features to be tested as follows:

- Scan barcode to view Menu.
- Android application make drink without touching a machine.
- IR sensor work well and detect hand.
- A robot deliver drink to customer..

6.3 Testing Tools and Environment

- To test NodeMCU, we require software called Arduino IDE.
- The NodeMCU must connect first to the WIFI hotspot.

6.4 Test Cases

6.4.1 E_Menu

Feature Name	Story ID	Test Case ID	Summary	Execution Steps	Expected Result
Install	Mob-1214	Mob-1214_1	Verify that application should be Installed successfully.	1. Click on install button. 2. Navigate to the menu and click on the newly installed app.	The application should be Installed successfully.
scan Barcode	Mob-1214	Mob-1214_2	Enter The List Of Drinks	1. Click on App. 2. Scane Gaven Barcode	The application should be scan successfully.
Create Order	Mob-1214	Mob-1214_3	Customer Can Craete Order	1. Open the application. 2. Scane Gaven Barcode 3. Choose Your Drink. 4. Subemit Your Order.	Order Will Be In the Screen Order .
Accept_Order	Mob-1214	Mob-1214_4	Admin View Order and Accept It.	1. Open the application. 2. Scane Gaven Barcode 3.Log In 4. View Order . 5. Accept Order.	Order Accepted.
Reject_Order	Mob-1214	Mob-1214_5	Admin View Order and Reject It If Order Not Found .	1. Open the application. 2. Scane Gaven Barcode 3.Log In 4. View Order . 5. Reject Order.	Order Rejected.
Login	Mob-1214	Mob-1214_6	Admin And Waiter Asssign User And Password	1. Open the application. 2. Scane Gaven Barcode 3.Log In	Login Sucessfully

6.4.2 Automatic Hand Sanitizer

T1: Infrared Sensor

From the Table A, it has been set on the work sensitivity of the infrared module, by adjusting it according to the hand distance requirements of the infrared sensor.

Table 6.2: Hand Distance Experimental Result of Infrared Sensor

Distance(mm)	Sensor Information
10	Sensor Detection
20	Sensor Detection
30	Sensor Detection
40	Sensor Detection
50	Sensor Detection
60	Sensor not Detection
70	Sensor not Detection

The automatic hand sanitizer testing can run smoothly with a minimum detection error of transferring data.

6.4.3 Waiter Robot

T1: Actual Robot Construction and Layout

The robot was tested on the cafe model. The testing was done by simulation, i.e., the robot is simulated in a place that has been made to resemble the real situation. The café model was made of white material sheet with black line as trajectory printed on it. The café model consists of four tables.

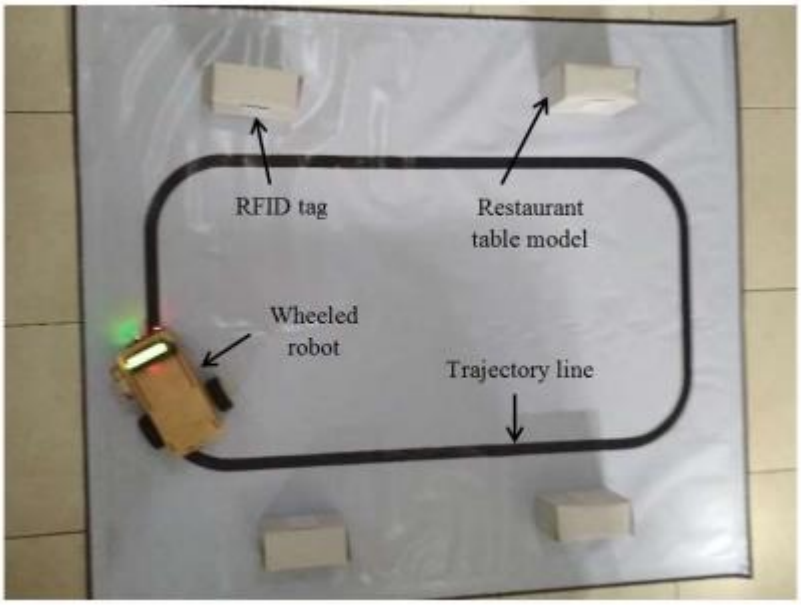

Figure 6.1: Café Model Layout Used for Testing

T2: Line Sensors Testing

There are two-line sensors used by the robot. The robot will move following the black line. Testing was done by simulating it on the black line that has been made. Table B presents the test results conducted more than ten laps of café model layout of Figure a. All tests were successful which suggested that the line sensors were work consistently.

Table B. Line Sensors Test Results

No.	Left Sensor	Right Sensor	Robot Motion
1.	0	0	Forward
2.	1	0	Turn right
3.	0	1	Turn left
4.	1	1	Stop

T3: RFID Reader and Robot Stop Testing

The Keypad has four buttons namely table A, table B, table C, and Table D. Each table has its own RFID tag mounted on it with unique RFID code. RFID reader detects and reads RFID code of every table that is passed by the robot. When the RFID code is matched then the robot will stop in front of the table abruptly otherwise it will continue to run passing the table and detect RFID code of the next table and so on until it finds the correct RFID code. In the experiment, the RFID reader has specifications only be able to detect and to read the tag which has distance within 2 cm from it. The experiment was carried out as follows. The robot was given command to find the intended table. After that robot runs by following the trajectory line to find the intended table and then stops in front of it. Then the distance of the RFID reader of the robot to the axis of the RFID tag of the table was measured. After stopping in the set time, in this publication was ten seconds, the robot was expected to continue. For every table, there were five experiments conducted. The results showed that robot always able to find intended table correctly without mistake and stop in front of it, as can be seen in Figure b, but with varying distance between RFID reader of robot and RFID tag of table were in the range from 0.5 to 1.5 centimetres as can be seen in Table C. These results suggested that the robot worked consistently.

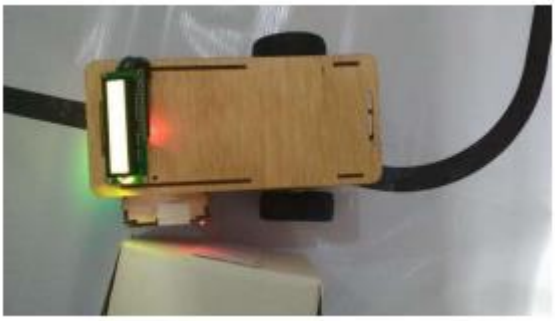

Figure 6.2: Robot Stops in Front of Intended Table

Table 6.4 RFID Reader and Robot Stop Testing Results

Button	Table Number (RFID Code)			
	A (6C 39 3D 32)	B (7C 39 3D 32)	C (5C 39 35 22)	D (77 1D 3C 27)
Table A	Stop ± 0.5-1.5 cm	Passed	Passed	Stop ± 0.5-1.5 cm
Table B	Passed	Stop ± 0.5-1.5 cm	Passed	Stop ± 0.5-1.5 cm
Table C	Passed	Passed	Stop ± 0.5-1.5 cm	Stop ± 0.5-1.5 cm
Table D	Passed	Passed	Passed	Stop ± 0.5-1.5 cm

The distance was caused by the delay time from the RFID reader detecting the RFID code and then Arduino providing a command for the DC motor to stop where the delay time. After stopping for designated time, the robot will continue moving forward to return to base, i.e. table D.

Chapter 7
SC3-19 Security

7.1 Introduction

The protection of computer-based resources that includes hardware, software, data, procedures and people against unauthorized use or natural Disaster is known as System Security.

System Security can be divided into four related issues:

- Security
- Integrity
- Privacy
- Confidentiality

SYSTEM SECURITY refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

DATA SECURITY is the protection of data from loss, disclosure, modification and destruction.

SYSTEM INTEGRITY refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

PRIVACY defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

CONFIDENTIALITY is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

7.2 Android Security Model

Android has a predefined set of permissions. This permission must be specified by the application in the manifest file of the application it requests for. At the time of installation of the application all the permissions are displayed to the user. If user allows all the permission, then the application installed successfully otherwise it is rejected completely. But in this case user do not have any choice to approve selected permission. There are near about 134 system defined permissions and developer can also create their own user defined permissions.

Javed Parvez et al proposed an enhanced Android Security Framework (AFS) which applies Advanced Encryption Standards (AES) algorithm to all the files stored in media and if any malicious app that want to modify the file found to be vulnerable then it will restrict access to the file. Malware detection can be done using 3 techniques as

1. Signature Matching
2. Heuristics
3. Hashes

A new security enhancement in the package installer system is proposed by Aparna Bhonde et al in which it is suggested that Android system should be able to identify the malware before the application that is being installed. When an application gets installed Package Manager should be able to detect the vulnerability along with the authenticity of the android application. For this purpose, a provision is suggested in android operating system structure.

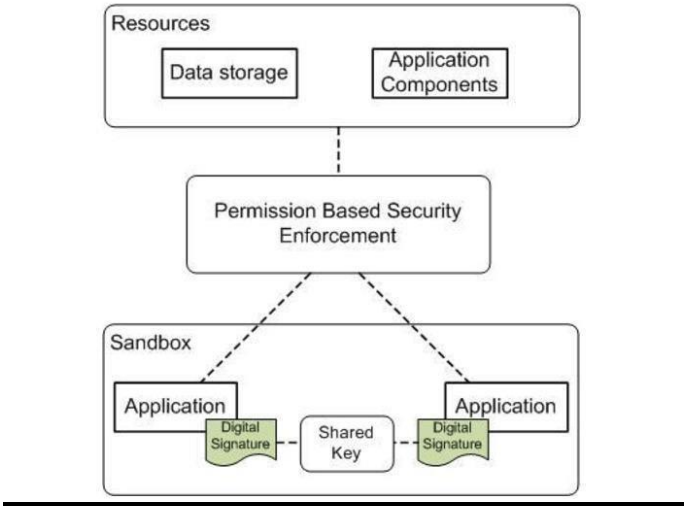

Figure 7.1 Android Security model

7.3 Firebase Security

Fingerprint Authentication

What are the advantages of scan authentication?

- can be used via Intents (little code required).
- can be embedded in an Activity, for advanced customization of UI .
- scanning can be performed in landscape or portrait mode.
- Camera is managed in a background thread, for fast startup time.

These are several steps to follow to enable scan authentication. But still, you have to write lots of code to implement that.

Steps of implementation scan authentication in our app:

1) Add Dependency to Gradle:

Add this dependency to your module's Gradle file.

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.journeyapps:zxing-android-embedded:4.2.0'
    implementation 'androidx.appcompat:appcompat:1.0.2'
}

android {
    buildToolsVersion '28.0.3' // Older versions may give compile errors
}
```

Figure 7.2 Scan Authentication Step_1

2) Hardware Acceleration:

Hardware acceleration is required since TextureView is used. Make sure it is enabled in your manifest file:

```
<application android:hardwareAccelerated="true" ... >
```

Figure 7.3 Scan Authentication Step_2

3) Android Permissions:

The camera permission is required for barcode scanning to function. It is automatically included as part of the library. On Android 6 it is requested at runtime when the barcode scanner is first opened.

When using Barcode View directly (instead of via IntentIntegrator / CaptureActivity), you have to request the permission manually before calling BarcodeView#resume(), otherwise the camera will fail to open.

Chapter 8
Conclusion and Future Work

8.1 Conclusion

We build four Solutions that mitigate the effect of manual Drinks Machine, Paper Menu, manual hand sanitizer and waiter inn café.

- **E_Menu (electronic menu)**

  Is an interactive **Cafe** ordering system and an innovative alternative to traditional paper **menus**. With new **digital menus** your Guests can quickly look through all the items and make their choice.

  E_Menu has too many features that you won't imagine it's available. let's take a look:

  - Scanned by QR Code.
  - Responsive and fits all devices and screens.
  - Allow customer to see and order through your portal.
  - You Receive the order

  - **No Touches or virus spread,**

    Keep your customers safe from the touching and sharing of physical menus.

  - **Keep your staff safe,**

    It’s not just the consumers you must take care of, also your staff members are also at danger of getting affected as they serve dine-in consumers. By joining a eMenu and adding QR code for restaurant menu on the table or the wall where it’s quickly scannable by your clients, your staff members no extended have to hand in physical menus and describe the specials.

- **Smart Drinks Machine**

  From android application you can make your drink without touching a machine.

- **Waiter Robot**

  The robot was able to move by following the trajectory line and recognized the intended table by reading the RFID code of each table and stopping in front of it precisely with the distance of between 0.5 and 1.5 centimeters from the RFID tag axis.

- **Automatic hand sanitizers** are an important asset in combating the spread of **COVID-19**. These dispensers show your customers that you care about their health and the health of your staff.

  Sanitizers are typically liquid or gel structures that may or may not contain alcohol. People use it **to get rid of germs, viruses, and bacteria** that pile up on our hands and surfaces around us. In the wake of the COVID-19 pandemic.

  Touchless hand sanitizer work greatly for a variety of **locations**, including:

  - Hospitals and medical offices
  - Restaurants, Cafes, grocery stores, retailers, and malls
  - Education campuses
  - hotels
  - Manufacturing, industrial, logistics, and distribution facilities

8.2 Future Work

E_Menu

The following section describes the work that will be implemented with future releases of the software.

- Customize orders: Allow customers to customize drink orders.
- Enhance User Interface by adding more user interactive features. Provide Deals and promotional
- Offer details to home page. Provide Recipes of the Week/Day to Home Page
- Payment Options: Add different payment options such as PayPal, Cash, Gift Cards etc. Allow to save payment details for future use.
- Allow to process an order as a Guest.
- Delivery Options: Add delivery option.
- Order Process Estimate: Provide customer a visual graphical order status bar.
- Order Status: Show only Active orders to Restaurant Employees.
- Order Ready notification: Send an Order Ready notification to the customer.
- Restaurant Locator: Allow to find and choose a nearby restaurant.
- Integrate with In store touch screen devices like iPad.

Smart Drinks Machine

This Machine can be modified by adding Coin Detector to it. Once a customer select a Drink needed then inserts currency or credit into the machine, The machine begins to make this drink. For instance, Smart Drinks Machine increasingly found to encroach cafes nowadays, which reduces the time and also reduce the human effort.

Waiter Robot

There are various advanced features that we can add to the system. We could work on field mapping of the restaurant, for which we can use rotary encoders. With the help of field mapping, the error in traversal can be minimized and the time for traversal of the robot can also be reduced to a great extent. As we can see in recent times, everyone is trying to avoid human contact in the COVID-19 pandemic. These robots can be modified as per the requirement and deployed in any field, like medical, industries, households, etc. Robots like these can be used in the isolation wards to serve the medicine or food to the affected person. It can also be used in industries as a helping hand.

Hand Sanitizer

The system can be improved by adding

- Oxygen level and temperature measurement.
- Smart electro-magnetic lock which ensures only after sanitizing hands, the person is allowed inside.
- Alternative power sources such as solar can be installed to power up the system in case of the failure or cut off of the conventional power source.
- A camera module with image processing technique and a database can be created to maintain the history of how many people went through the process in a cafe, thereby providing us the statistics which will help plan well.

References

1. Ing-K L, Chih-C W, et al February 2020 Effective Strategies to Prevent Coronavirus Disease-2019 (Covid-19) Outbreak in Hospital Journal of Hospital Infection

2. Zakir K, Khayal M, Ali A, Hazir R March 2020 Coronavirus Outbreaks: Prevention and Management Recommendations, Drugs & Therapy Perspectives

3. T. S. Hong et al., “Systems-Level Quality Improvement A Hand Hygiene Compliance Check System : Brief Communication on a System to Improve Hand Hygiene Compliance in Hospitals and Reduce Infection,”

4. E. Tartari et al., “Train-the-Trainers in hand hygiene : a standardized approach to guide education in infection prevention and control,”

5. Afsheen, S., Fathima, L., Khan, Z., & Elahi, M. (2018). A self-sufficient waiter robot for serving in restaurants. International Journal of Advance Research and Development

6. Ammanagi, N., Joshi, M., Daswani, U., Dondapati, S., & Jayaram, V. (2016). Automated Service System for a Restaurant Using a Line follower Robot. International Journal of Innovative Research in Computer and Communication Engineering

7. Bankar, A., & Suresh, S. S. (2015). Intelligent Restaurant - Menu Ordering System. IOSR Journal of VLSI and Signal Processing

8. Chin, W. J., Lim, M., Yong, J. C. E., Al-Talib, A. A. M., & Chaw, K. H. (2019). Service Time Performance Analysis of Improved Automated Restaurant by Layout Reconfiguration and Conveyor System. IOP Conf. Series: Materials Science and Engineering

9. Gurav, S., Khot, P., Potadar, D., Shelke, S., & Chougula, B. (2017). Remote controlled Waiter Robot for Restaurant Automation. International Journal of Application or Innovation in Engineering & Management

10. Hamid. A., Hamdany, S., Albak, L.H., Rafi, R., & Al-Nima, O. (2019). Wireless Waiter Robot. TEST Engineering and Management

11. Mirgal, D., Parab, P., Puro, A., & Dakhare, B. (2018). Smart Automated Restaurant. International Journal of Engineering Science and Computing

12. Nataliana, D., Hadiatna, F., & Fauzi, A. (2019). Rancang Bangun Sistem Keamanan RFID Tag menggunakan Metode Caesar Cipher pada Sistem Pembayaran Elektronik

13. Oriolo, G. (2014). Wheeled Robots. Encyclopedia of Systems and Control

14. Supriyono, H., & Hadi, A. N. (2017). Designing A Wheeled Robot Model For Flammable Gas Leakage Tracking

15. Shiny, J. S., Kumar, A. M., Nanthagopal, V., & Raguram, R. (2017). Automation of Restaurant (Ordering, Serving, Billing). International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Appendix

SC3-19 Coding

Smart Drinks Machine

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

// Set these to run example.
#define FIREBASE_HOST "smartcoffee-e221a-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "AD3GR3XsJpcSjEH2WttWULL772qh0S2hjLh0ijrY"
#define WIFI_SSID "home"
#define WIFI_PASSWORD "bbb123123"

void setup() {
  Serial.begin(9600);
  pinMode(D3,OUTPUT);
pinMode(D2,OUTPUT);
pinMode(D1,OUTPUT);
pinMode(D0,OUTPUT);
pinMode(D4,OUTPUT);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.setString("Machine/1/name","hers");
  Firebase.setString("Machine/1/size","ser");



}


String b="";
String c="";
```

```
void loop() {

  b=Firebase.getString("Machine/1/name");
  c=Firebase.getString("Machine/1/size");


   if (b=="Captchino" && c=="Small") {
      Serial.println("led on");
      digitalWrite(D2,HIGH);
      digitalWrite(D4,HIGH);
      delay(3500);
      digitalWrite(D2,LOW);
      digitalWrite(D4,LOW);
      Firebase.setString("Machine/1/name","hers");
      Firebase.setString("Machine/1/size","ser");


  }
 else if (b=="Captchino" && c=="Medium") {
      Serial.println("led on");
      digitalWrite(D2,HIGH);
      digitalWrite(D4,HIGH);
      delay(7000);
      digitalWrite(D2,LOW);
      digitalWrite(D4,LOW);
      Firebase.setString("Machine/1/name","hers");
      Firebase.setString("Machine/1/size","ser");
  }
   else if (b=="Captchino" && c=="Large") {
      Serial.println("led on");
      digitalWrite(D2,HIGH);
      digitalWrite(D4,HIGH);
      delay(9000);
      digitalWrite(D2,LOW);
      digitalWrite(D4,LOW);
      Firebase.setString("Machine/1/name","hers");
      Firebase.setString("Machine/1/size","ser");


  }
```

```
 else if (b=="Cafe Latte"&&  c=="Small") {
     Serial.println("led on");
     digitalWrite(D3,HIGH);
     digitalWrite(D4,HIGH);
     delay(3500);
     digitalWrite(D3,LOW);
     digitalWrite(D4,LOW);
     Firebase.setString("Machine/1/name","hers");
     Firebase.setString("Machine/1/size","ser");

 }
  else if (b=="Cafe Latte"&& c=="Medium") {
     Serial.println("led on");
     digitalWrite(D3,HIGH);
     digitalWrite(D4,HIGH);
     delay(7000);
     digitalWrite(D3,LOW);
     digitalWrite(D4,LOW);
     Firebase.setString("Machine/1/name","hers");
     Firebase.setString("Machine/1/size","ser");

 }

 else if (b=="Cafe Latte"&&  c=="Large") {
    Serial.println("led on");
    digitalWrite(D3,HIGH);
    digitalWrite(D4,HIGH);
    delay(9000);
    digitalWrite(D3,LOW);
    digitalWrite(D4,LOW);
    Firebase.setString("Machine/1/name","hers");
    Firebase.setString("Machine/1/size","ser");

}
```

```
else if (b==" Iced cafe "&&  c=="Small") {
    Serial.println("led on");
    digitalWrite(D0,HIGH);
    digitalWrite(D4,HIGH);
    delay(3500);
    digitalWrite(D0,LOW);
    digitalWrite(D4,LOW);
    Firebase.setString("Machine/1/name","hers");
    Firebase.setString("Machine/1/size","ser");

}
 else if (b==" Iced cafe "&& c=="Medium") {
    Serial.println("led on");
    digitalWrite(D0,HIGH);
    digitalWrite(D4,HIGH);
    delay(7000);
    digitalWrite(D0,LOW);
    digitalWrite(D4,LOW);
    Firebase.setString("Machine/1/name","hers");
    Firebase.setString("Machine/1/size","ser");

}

else if (b==" Iced cafe "&&  c=="Large") {
   Serial.println("led on");
   digitalWrite(D0,HIGH);
   digitalWrite(D4,HIGH);
   delay(9000);
   digitalWrite(D0,LOW);
   digitalWrite(D4,LOW);
   Firebase.setString("Machine/1/name","hers");
   Firebase.setString("Machine/1/size","ser");

}
```

Hand Sanitizer

```
int motor = 4;  // motor connected to digital pin 4


void setup() {
  pinMode(motor, OUTPUT);  // sets the digital pin 4 as output
}

void loop() {
  val = digitalRead(inPin);   // read the input pin
  if(val==1)
  {
    digitalWrite(motor,HIGH);
  }
  else
  {
    digitalWrite(motor,LOW);

  }
}
```

Waiter Robot

```
#include<Keypad.h>
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 9
#define RST_PIN 8

#define input1  2
#define input2  3
#define input3  4
#define input4  5
#define speed1 6
#define speed2 7
#define center_sen A1
#define Right_sen A0
#define Left_sen A2
int key;

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
char Key[4][4]={
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}

  };
  byte rowpin[4]={30,31,32,33};
  byte colpin[4]={34,35,36,37};
  Keypad mykeypad=Keypad(makeKeymap(Key),rowpin,colpin,4,4);

void setup() {
  pinMode(13,OUTPUT);
  Serial.begin(9600);   // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
  Serial.println("Approximate your card to the reader...");
  Serial.println();

  pinMode(input1, OUTPUT);
  pinMode(input2, OUTPUT);
  pinMode(input3, OUTPUT);
  pinMode(input4, OUTPUT);
  pinMode(speed1,OUTPUT);

}
```

```
void loop() {
key=mykeypad.getKey();
while(key=='A')
{

   robot();
   rfid();
  rfid3();



}
while(key=='B')
{
    robot();
   rfidl();
  rfid3();
}
while(key=='C')
{
   robot();
   rfid2();
   rfid3();


}

}
```

```
void robot()
{
      int Right_val=digitalRead(Right_sen);
      int Left_val=digitalRead(Left_sen);
      int center_val=digitalRead(center_sen);
Serial.println(Right_val);
      if(Right_val==1&&Left_val==1)
     {
      Up();

     }

         else if(Right_val==1&&Left_val==0)
         {
          right();

         }

         else if(Right_val==0&&Left_val==1)
         {
          left();

         }

          else if(Right_val==0&&Left_val==0)
         {
          Stop();
         }       }
}
```

```
void rfid()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "6C 39 3D 22") //change here the UID of the card/cards that you want to g
  {
    Serial.println("Authorized access");
    Serial.println();
    Stop();
```

```
void rfid1()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "7C 2D 4C 18") //change here the UID of the card/cards that you want to give access
  {
    Serial.println("Authorized access");
    Serial.println();
    Stop();
    delay(10000);
```

```
void rfid2()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "5C 30 35 22") //change here the UID of the card/cards that you want to give access
  {
    Serial.println("Authorized access");
    Serial.println();
    Stop();
    delay(10000);

void rfid3()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();

 while(content.substring(1) == "77 1D 3C 27") //change here the UID of the card/cards that you want to give acces
  {
    Serial.println("Authorized access");
    Serial.println();
    Stop();
```

```
void Up()
{
  digitalWrite(input1,HIGH);
  digitalWrite(input2,LOW);
  analogWrite(speed1,100);
  digitalWrite(input3,HIGH);
  digitalWrite(input4,LOW);
  analogWrite(speed2,120);
}
void Back()
{
  digitalWrite(input1,LOW);
  digitalWrite(input2,HIGH);
  analogWrite(speed1,100);
  digitalWrite(input3,LOW);
  digitalWrite(input4,HIGH);
  analogWrite(speed2,120) ;
}
void left()
{
  digitalWrite(input1,LOW);
  digitalWrite(input2,HIGH);
  analogWrite(speed1,100);
  digitalWrite(input3,HIGH);
  digitalWrite(input4,LOW);
  analogWrite(speed2,120) ;
}

void right()
{
  digitalWrite(input1,HIGH);
  digitalWrite(input2,LOW);
  analogWrite(speed1,100);
  digitalWrite(input3,LOW);
  digitalWrite(input4,HIGH);
  analogWrite(speed2,120) ;
}
void Stop()
{
  digitalWrite(input1,LOW);
  digitalWrite(input2,LOW);

  digitalWrite(input3,LOW);
  digitalWrite(input4,LOW);
}
```