

Secure Food Delivery – Design Manual



Contents

Introduction.....	4
Hardware Design	5
Sensors	5
RFID Sensor.....	6
Door Sensor	7
Actuators	8
Solenoid Door Lock.....	8
LCD Display	9
Relay	10
LEDs	11
Piezo Buzzer	11
Development Board - ESP32 Microcontroller.....	12
Additional components	13
Buck Converters.....	13
Battery Pack	13
Battery Charging Module (BMS-30A-4S).....	14
3D Model	15
3D View of the Box	15
3D Circuit Design	17
Circuit diagram	18
Pinout Diagram	18
Power, Control, and Data Flow.....	18
Software Design.....	19
Database.....	19
DBMS	19
ER Diagram	20
Tables.....	20
Mobile Application	21
Technology – Flutter.....	21
Quick Lookup Into SFD App.....	21

Functionalities.....	22
Security	23
API.....	24
Main Technologies.....	24
Communication Protocols.....	24
Architecture.....	25
Security	26
Cloud Deployment.....	27

Introduction

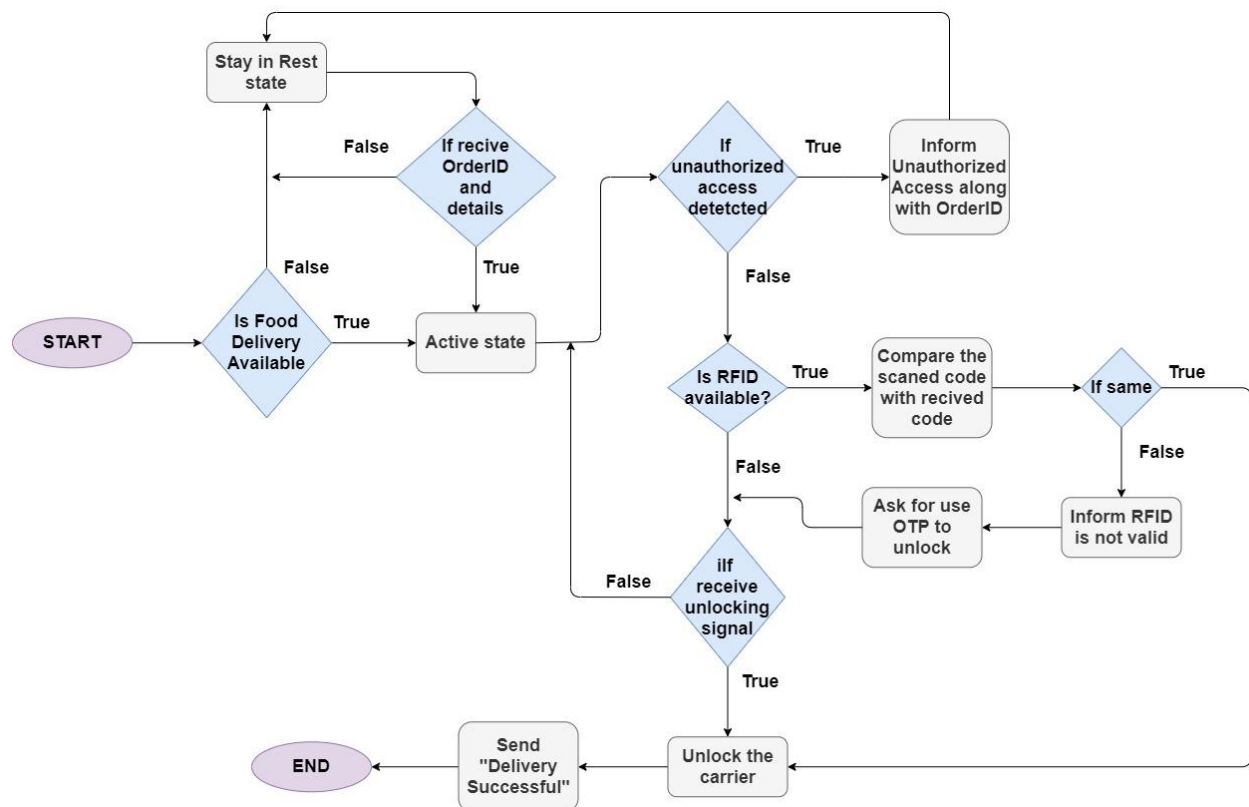
Secure Food Deliver is and new way of securing the food throughout the delivery process. It is a basically an IoT driven solution to address some problems that have been raised in the online food delivery context. In current situation the delivery process is more abstract to the customers, therefore with that, nowadays customers are not very satisfied, and they do not trust the process until the delivery has arrived at their houses. Specially in pandemic situations, people more tend to order the foods to their houses rather visiting to the restaurant. Reason behind the loosing the customer's satisfaction and the trust is that customer is not always 100% trust on the rider since he/she has full access to the foods throughout the delivery process. There were many incidents have been recorded, that were saying the rider has stolen the foods. As a new team who are seeking to solve real world problem with the help of innovative technology, we saw that as a opportunity to solve and address.

Here we mainly focused to give the fully ownership of the delivery box to the customers. Customers can unlock the carrier by using their mobile application with the provided OTP or the RFID tag card that has issued by our service. When those events occur, customer always gets a notification SMS to their mobile phone. If the delivery box has opened the without permission provided by the API, it will indicate as a unauthorized access, and then will notify to the customers as well as the food delivery company. With those features, it can be state that, this solution has addressed the issue occurred throughout the delivery process.

Hardware Design

This section of the Design Manual will explain the overall hardware design in the scope of sensors, actuators, development board, communication devices, 3D model of the hardware, and Circuit diagram with the design decisions, specifications, and wirings to the development board.

Hardware Control Flow:



Sensors

The overall solution of Secure Food Delivery is based on the security of this product. Trustworthiness is the focus area in both software and hardware design. Apart from software, the security of the hardware is achieved through several sensor modules. There are 2 sensors to sense the features of the delivery box.

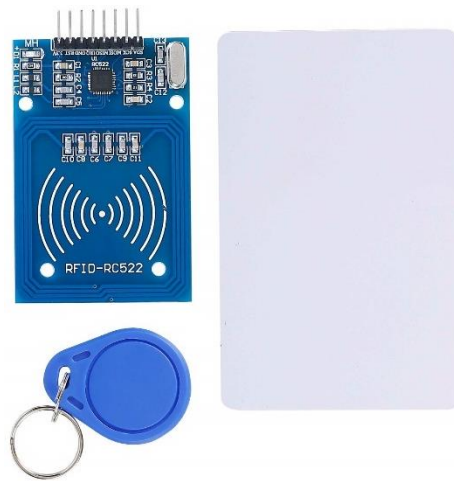
One is to sense the opening and closing of the door. Here we are using a door sensor to sense the event of opening and closing of the delivery box's lid. Secondly, we are providing an unlocking feature via RFID cards. Therefore, the RFID reader is a sensor that we use in the hardware design to ensure that only authorized parties can access the locking device.

RFID Sensor

For the RFID sensor module, we have used **RFID-RC522 Sensor**.

Design Decisions:

- RFID is provided to the registered customers of the Secure Food Delivery service.
- Once the customer has registered via the mobile application, the corresponding RFID card is issued to the customer and the RFID Code is stored in the database.
- When the user tries to unlock the device using RFID, the data will be published to the relevant MQTT topic which the microcontroller itself has subscribed already to.



Specification:

- MFRC522 chip-based board
- Operating frequency: 13.56MHz
- Supply Voltage: 3.3V
- Current: 13-26mA
- Read Range: Approx 3cm with supplied card and fob
- SPI Interface
- Max Data Transfer Rate: 10Mbit / s
- Dimensions: 60mm × 39mm

Wiring:

RC522	ESP32
3.3V	3V3
RST	D32
GND	GND
IRQ	Not Required
MISO	D19
MOSI	D23
SCK	D18
SDA	D5

Door Sensor

To identify the door opening and closing event, an **MC-38 Magnetic Door Sensor** is used.

Design Decisions:

- The door sensor is mainly used to identify any unauthorized access. If there is an ongoing order in the delivery box, it cannot be opened without any predefined way. (Either using OTP or RFID card)
- If unauthorized access is detected via the sensor, the event is published to the API and then, an SMS will be sent to the relevant customer addressing the issue.
- Also, when any event is detected via the door sensor, it is displayed using LEDs. If the box is opened, a red color LED will light up showing the open event of the box. Similarly, Green LED is for closed events, and Blue LED is for unauthorized access.



Specification:

- Rated current: 100mA
- Power rating: 3W
- Operating distance: 15 - 25mm
- Cable Length: 30.5cm \pm 12mm
- Sensor Output: Normally Closed (NC) (Switch is closed when the switch and magnet are together)

Wiring:

MC-38	ESP32
Signal	D15
Ground	GND

Actuators

The operations based on the decisions are done using actuators. In the design of the hardware, there are several units to handle the outputs such as showing a message, opening the lid of the delivery box, showing status, alerting the users, controlling the power supply, etc.

To achieve these tasks, the actuators that we have used are a solenoid door lock, LCD display, relay, LEDs, and piezo buzzer. The below part of the manual will explain the design of the hardware based on actuators.

Solenoid Door Lock

Unlocking and locking of the delivery box lid are done using a **12V Solenoid Door Lock**.

Design Decisions:

- The lid is opened when the customer enters the OTP or RFID. In that scenario, the door lock is powered 12V using a battery pack and when 12V has supplied, the lock will be unlocked.



Specification:

- Voltage: 12VDC
- Current: 0.6A
- Size: 53 x 26 x 23mm
- Weight: 142g
- Unlocking time: 1S
- Temperature: -40 ~ +50

Wiring:

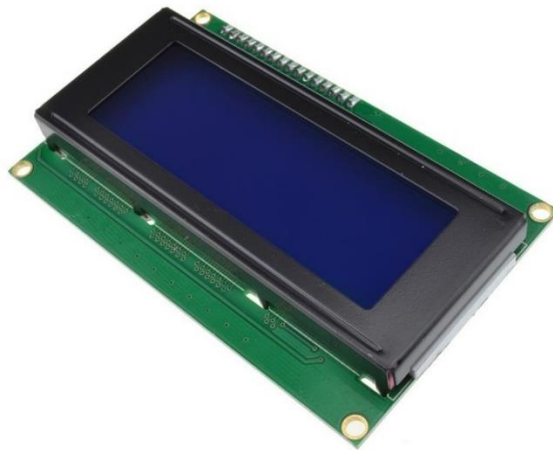
- The door lock needs 12V to operate. Therefore, it is powered using a battery unit. Also, a relay is used as a switch to turn the lock on and off. Therefore, the door lock is connected to the battery unit via the relay's normally open contact.

LCD Display

An LCD display is used to show important messages to the customers and riders when necessary. For that purpose, a **20X4 Character LCD** is used.

Design Decisions:

- The LCD display is displaying several messages at each step of the delivery process.
- On the first line of the display, it shows the name of the service “*Secure Food Delivery*”.
- An unlock event is also shown on the display in both unlocking events (OTP and RFID) as “*OTP is validated*” and “*RFID is validated*”.
- If an unauthorized event has occurred, it will also show on the display.



Specification:

- Display format : 20 characters x 4 lines
- Dot matrix (w x h) : 5 x 8 dots
- Character size (w x h) : 2.95 x 4.75 mm
- Character pitch (w x h) : 3.55 x 5.35 mm
- Lcd driver is : sitronix st7066u (or equivalent)
- Interface: 4-bit parallel and 8-bit parallel
- Operating temperature: -20 ~ 70°C

Wiring:

LCD	ESP32
GND	GND
VCC	3V3
SDA	D21
SCL	D22

Relay

A **5V Single Channel Relay** was used to control the solenoid door lock.

Design Decisions:

- The 12V power supply needed to be isolated from the microcontroller and at the same time there was the need to turn on and off the power supply to the lock.
- When the lock is to be unlocked, make the relay control pin LOW. Then the normally open contact will be closed powering the lock.
- Power will be disconnected after 15s delay.



Specification:

- Normal Voltage is 5V DC
- Normal Current is 70mA
- AC load current Max is 10A at 250VAC or 125V AC
- DC load current Max is 10A at 30V DC or 28V DC
- Operating time is 10msec
- Release time is 5msec
- Maximum switching is 300 operating per minute

Wiring:

Relay	ESP32	Solenoid Lock
VCC	3V3	-
GND	GND	-
IN	D33	-
COM	-	+ve
NO	-	+ve
NC	-	-

LEDs

Design Decisions:

- Delivery statuses are indicated using LEDs as it is simple to identify and understand. 3 indications are performed using LEDs.
- If the door is opened - Red LED ON and Green LED OFF
- If the door is closed – Green LED ON and Red LED OFF
- Unauthorized access – Blue LED ON



Wiring:

LED	ESP32
Red	12
Green	14
Blue	13

Piezo Buzzer

Design Decisions:

- The door unlock event is notified using the piezo buzzer.



Wiring:

Buzzer	ESP32
+ve	D26
GND	GND

Development Board - ESP32 Microcontroller



for the development board, an ESP32 microcontroller is used. The components that are connected to the microcontroller are listed below.

- I2C interface
- RFID module
- Relay
- Door Lock Sensor
- Sim module
- Buck converter
- LEDs
- Buzzer
- Switch

Some libraries must be installed to work properly. They are,

```
• #include <HttpsOTAUpdate.h>
• #include <Update.h>
• #include <WiFi.h>
• #include <PubSubClient.h>
• #include <ArduinoJson.h>
• #include <LiquidCrystal_I2C.h>
• #include <MFRC522.h>
```

Additional components

Buck Converters

Design Decisions:

- Different components in the hardware needs different voltages to operate. In that scenario, a 5V and 12V power supply were needed. 5V for the relay and ESP32 board while 12V for the Solenoid Door Lock.
- Therefore, 2 buck converters were used. The battery supply voltage is ~15V and that is converted into 5V and 12V separately and connect to relevant components.



Specification:

- Input voltage: 3-40V
- Output voltage: 1.5-35V(Adjustable)
- Output current: Rated current is 2A, maximum 3A
- Switching Frequency: 150KHz
- Operating temperature: Industrial grade (-40 to +85)
- Conversion efficiency: 92%(highest)

Battery Pack

The power supply was resolved using 4 **3.7V 18650 Batteries**.



Battery Charging Module (BMS-30A-4S)

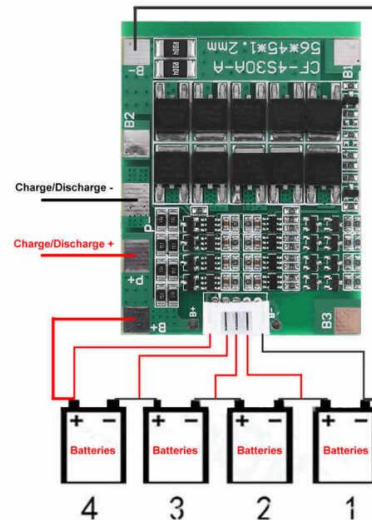
To charge the batteries, the **BMS-30A-4S** module is used.

Design Decisions:

- When the batteries drain, they should be recharged. But in the hardware solution, the battery removal is not practical as it is built as a single component. Therefore, this module was used to charge the batteries when they are drained.



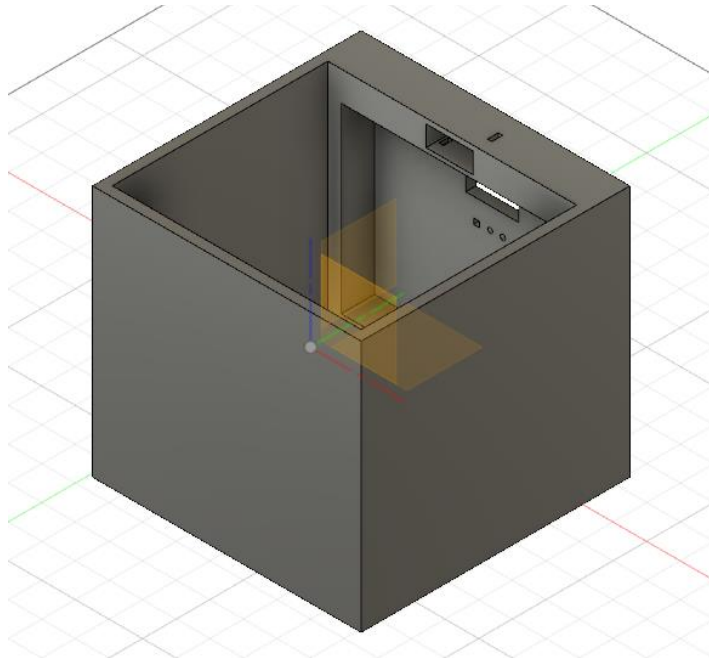
Wiring:



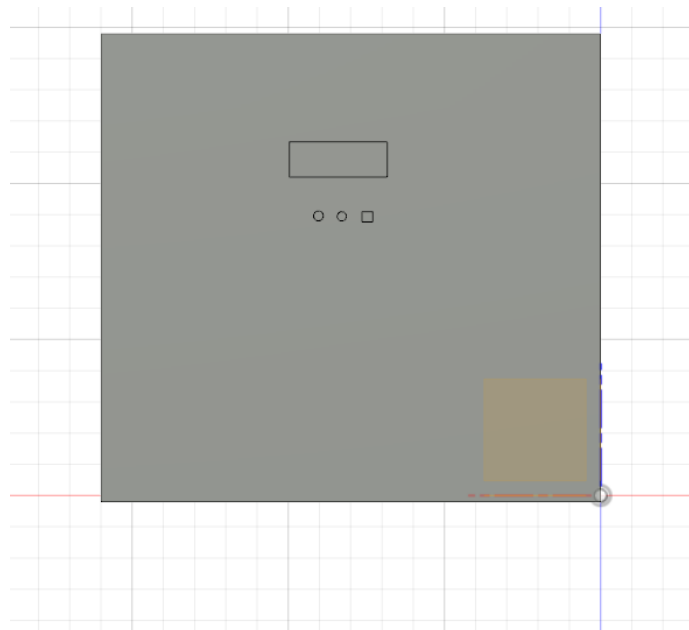
3D Model

3D View of the Box

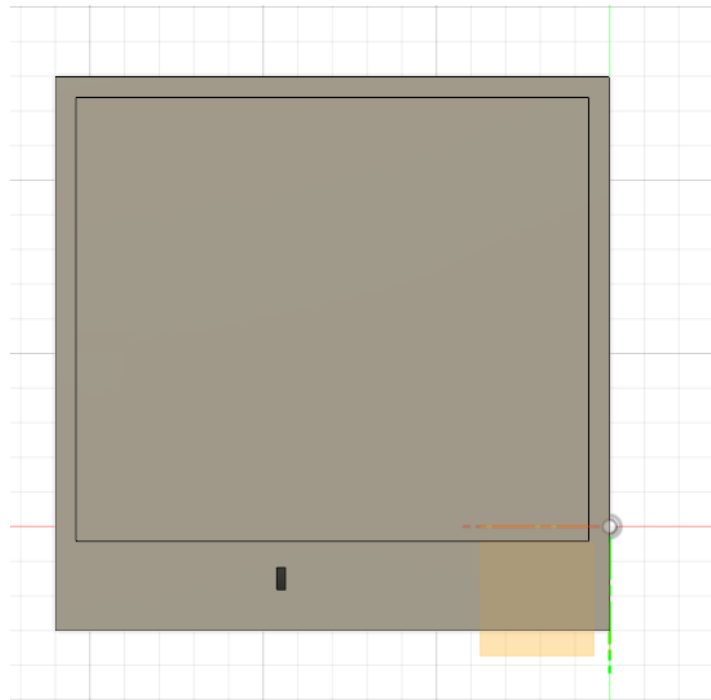
3D View:



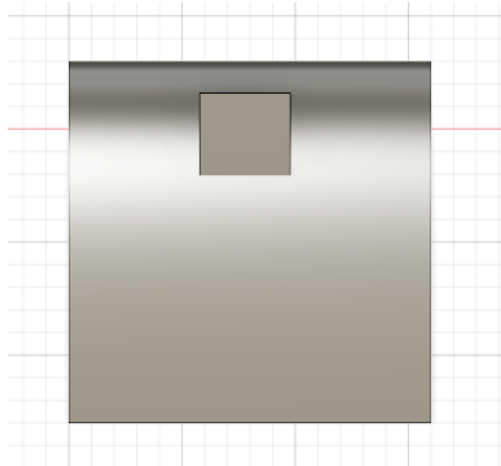
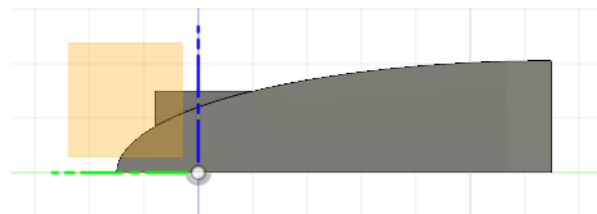
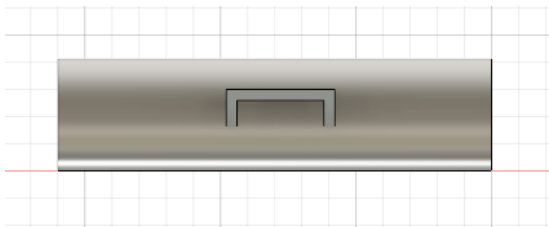
Front View:



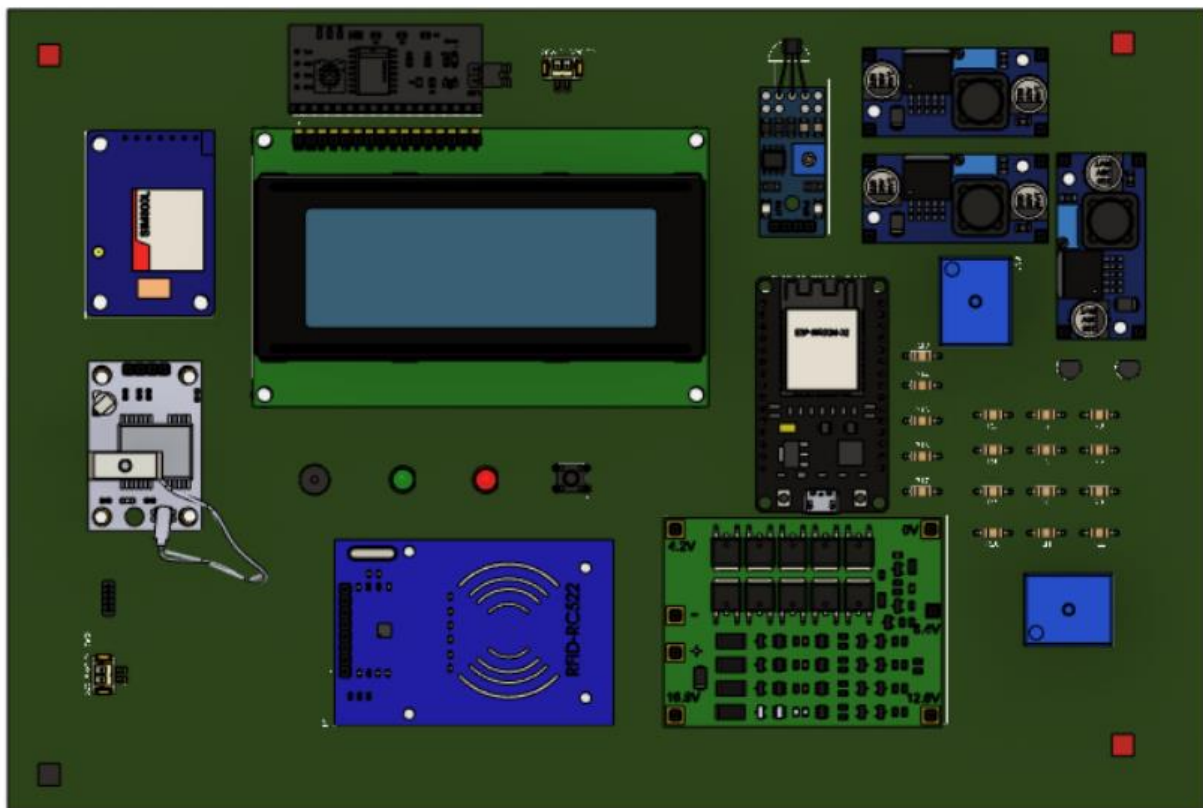
Top View:



Door View:

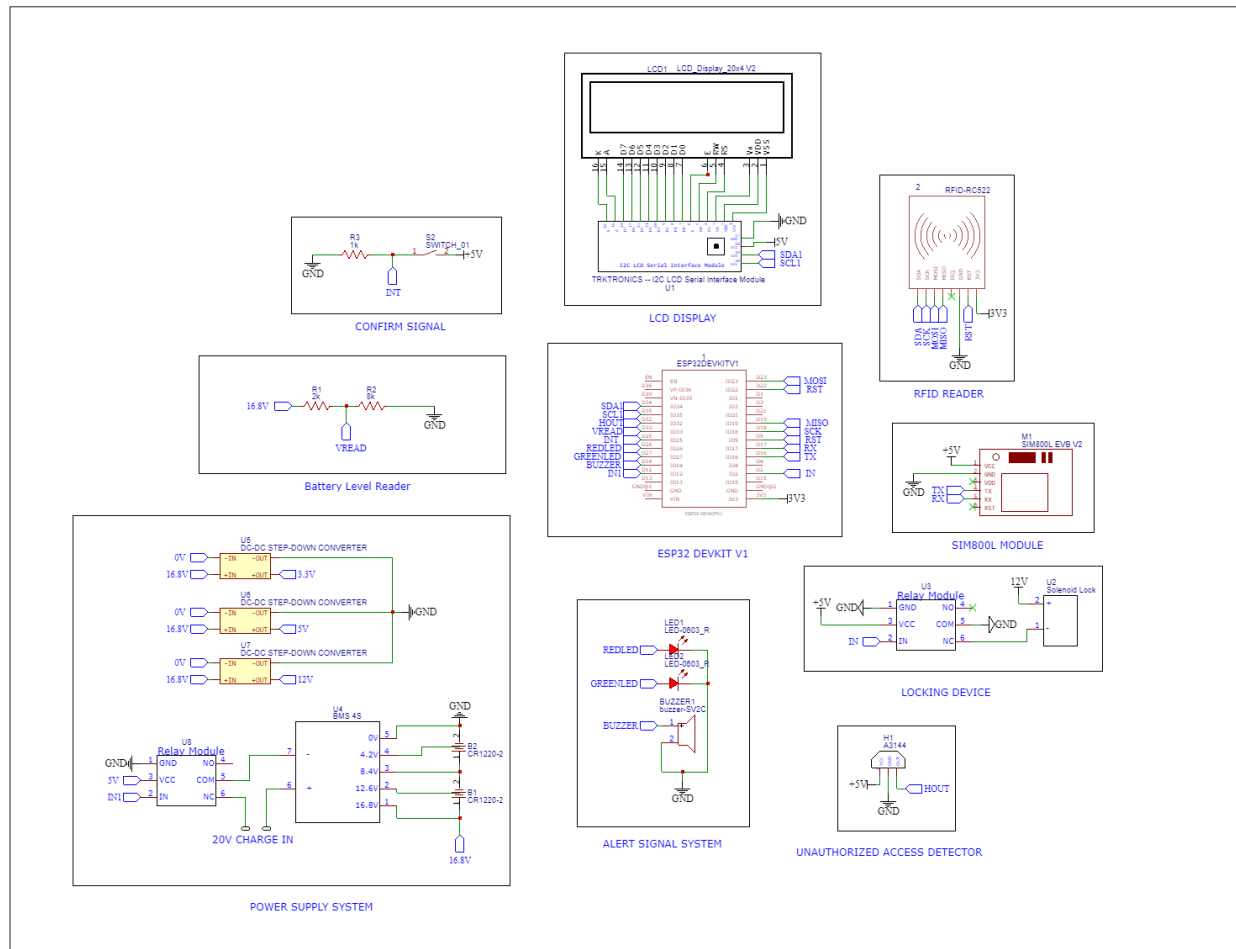


3D Circuit Design

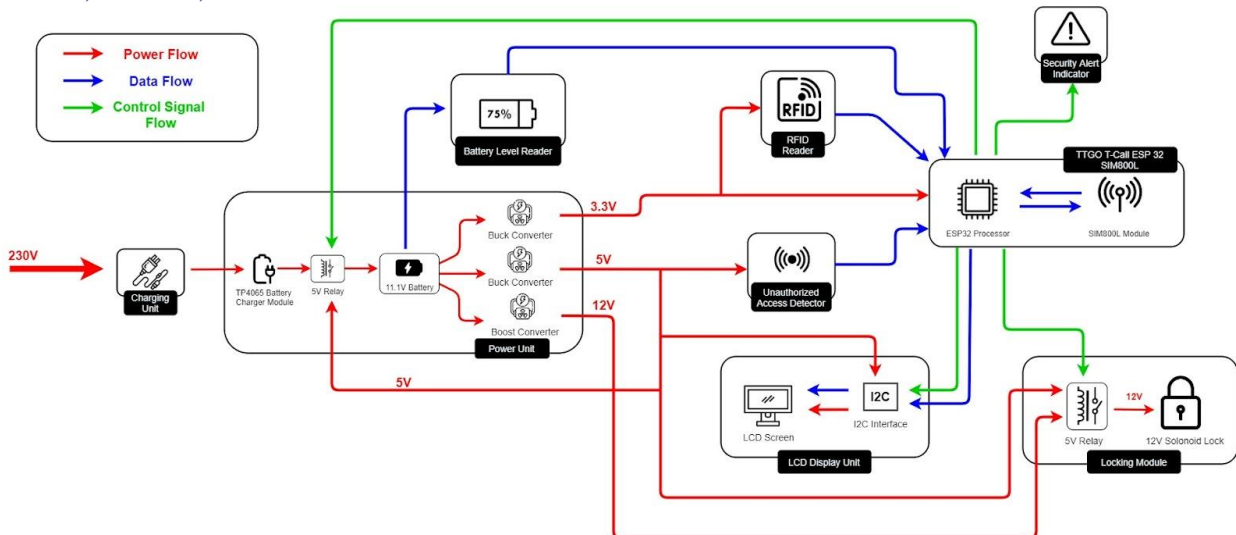


Circuit diagram

Pinout Diagram

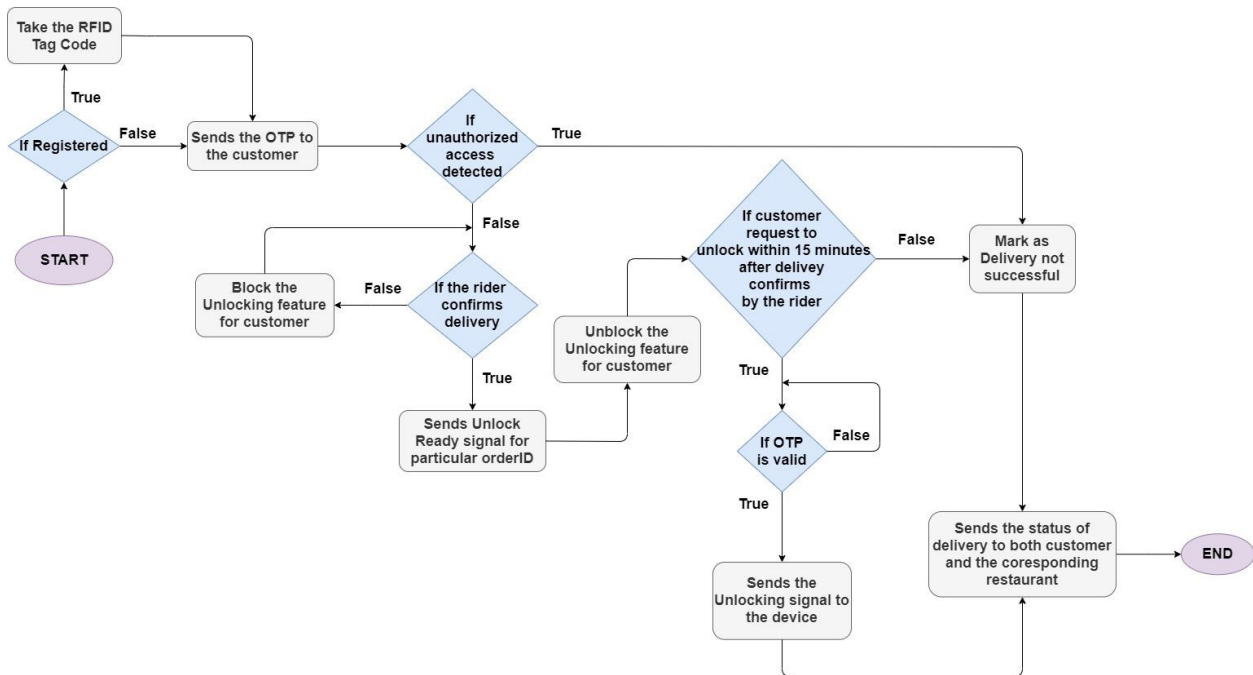


Power, Control, and Data Flow



Software Design

Software Control Flow:



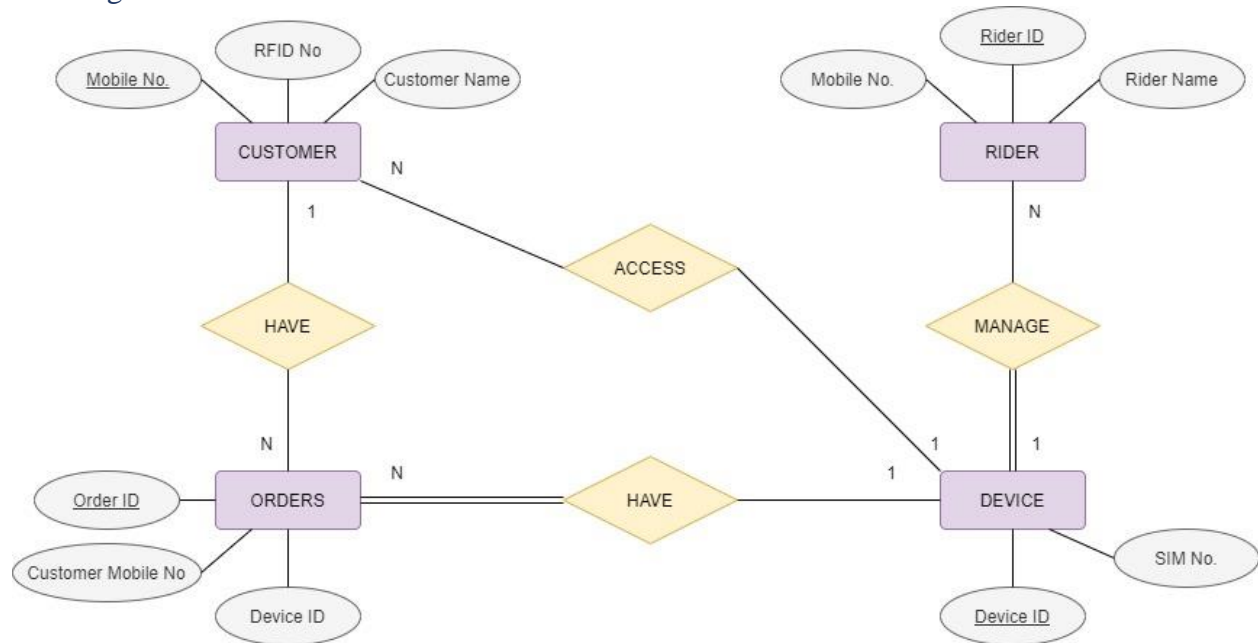
Database

DBMS

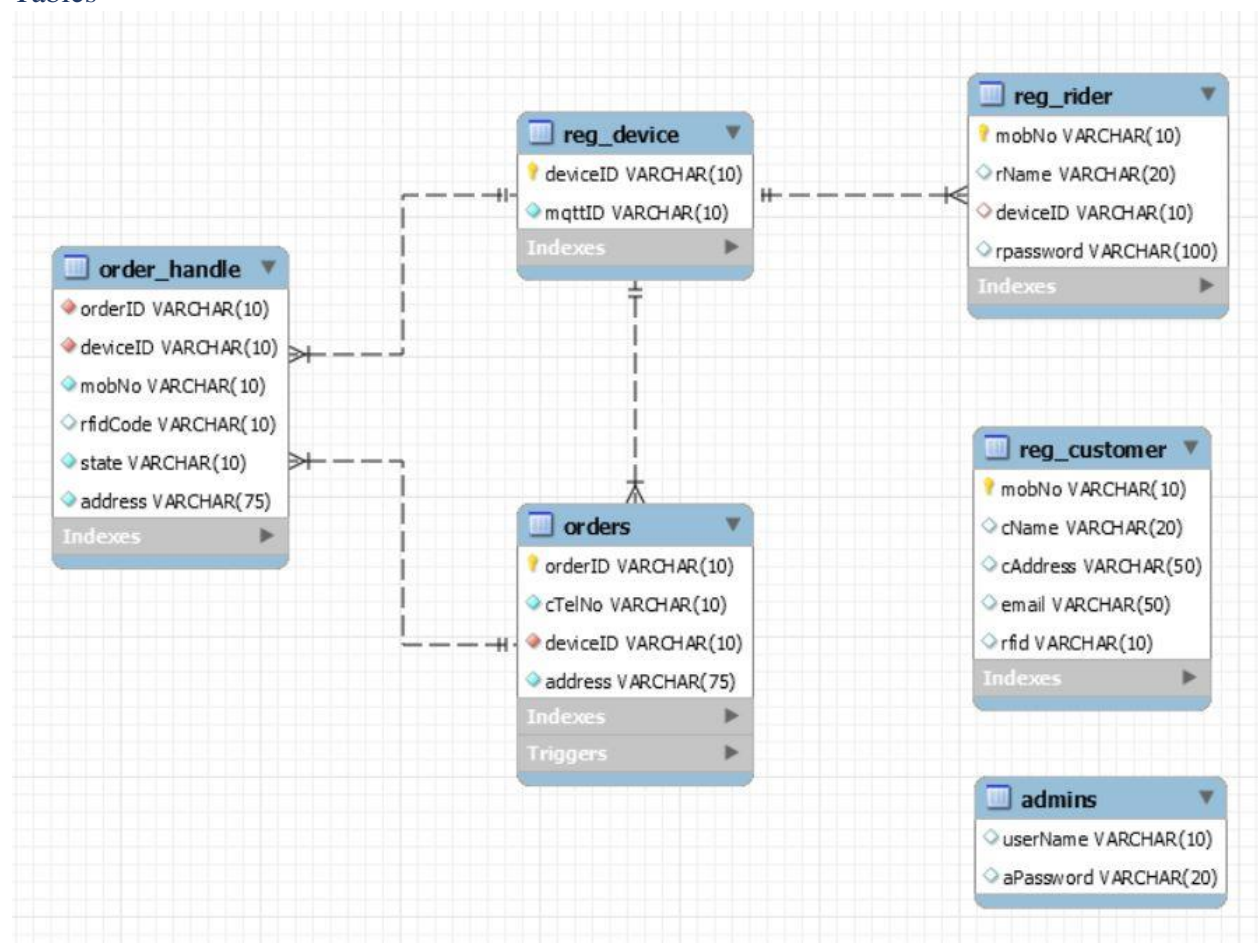
For the database management system, **MySQL** was used. The overall solution has several entity relations that must be implemented. Therefore, choosing a relational database was the most suitable for the solution.



ER Diagram



Tables



Mobile Application

The Secure Food Delivery service is providing a mobile application to make it easy to use the service. It is designed for both delivery riders and food customers.

The delivery riders can use the mobile application to register for the Secure Food Delivery service and after registration, they can use it for delivery service. There is a login interface that shows allocated deliveries for each rider. The riders can see a list of delivery orders and they can confirm the orders when they are ready to hand over the food to the customer.

The food customers can register for Secure Food Delivery service via the mobile application. But without registration, the customers can log into the mobile app to get the SFD service. Therefore, both registered and non-registered customers can log in to the mobile app. The advantage of registering into the mobile app is that after registration into the SFD service, we provide an RFID card to the users.

When the customer logs into the mobile app, they will be asked to enter the OTP sent as an SMS which ensures that 2-factor authentication is an advanced security feature. Then, by providing the OTP, customers can unlock the device.

Technology – Flutter

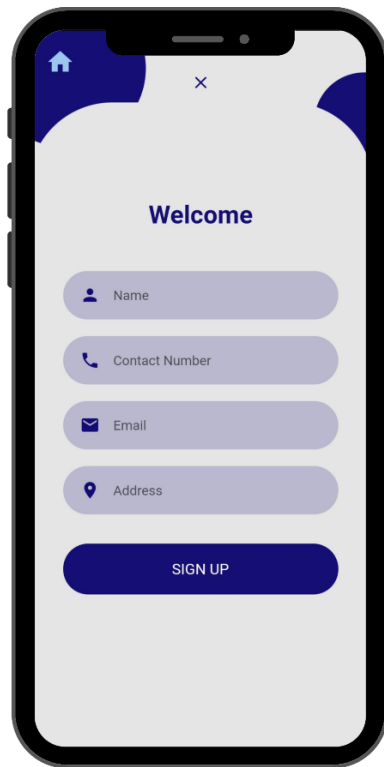
For mobile application development, **Flutter** was used.



Quick Lookup Into SFD App

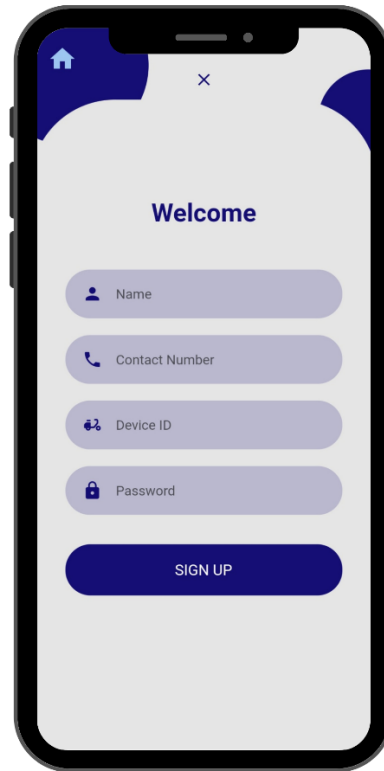
The design structure of the mobile application can be seen below.

Functionalities



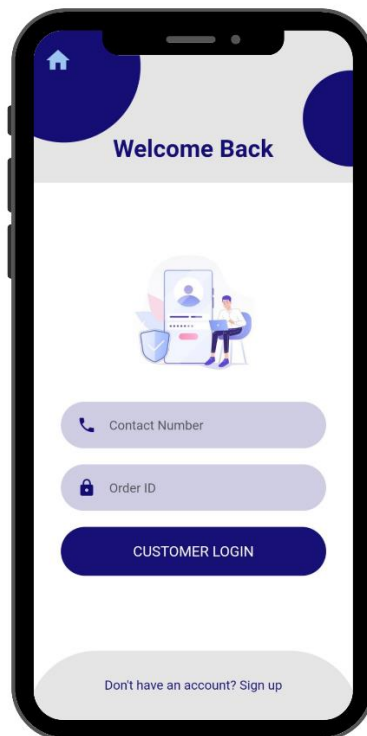
A mobile app screen for Customer Signup. At the top, there is a home icon and a close 'X' button. The title 'Welcome' is centered. Below it are four input fields: 'Name' (with a person icon), 'Contact Number' (with a phone icon), 'Email' (with an envelope icon), and 'Address' (with a location pin icon). At the bottom is a large blue button labeled 'SIGN UP'.

Customer Signup



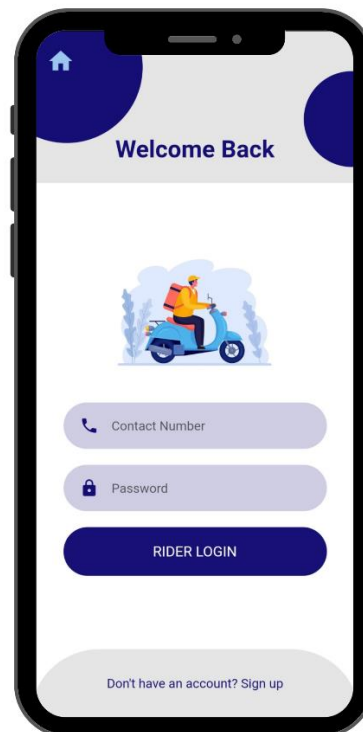
A mobile app screen for Rider Signup. At the top, there is a home icon and a close 'X' button. The title 'Welcome' is centered. Below it are four input fields: 'Name' (with a person icon), 'Contact Number' (with a phone icon), 'Device ID' (with a device icon), and 'Password' (with a lock icon). At the bottom is a large blue button labeled 'SIGN UP'.

Rider Signup



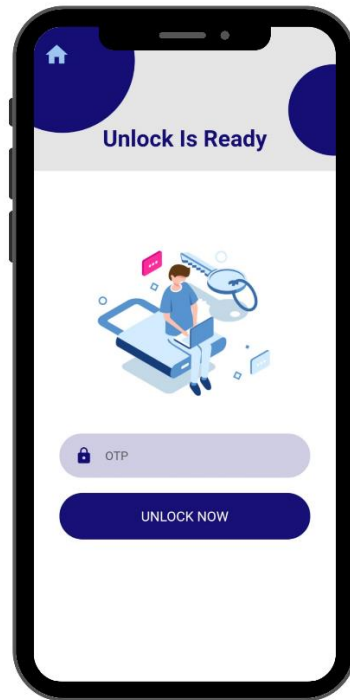
A mobile app screen for Customer Login. At the top, there is a home icon. The title 'Welcome Back' is centered. Below it is an illustration of a person sitting at a desk with a laptop. Underneath the illustration are two input fields: 'Contact Number' (with a phone icon) and 'Order ID' (with a lock icon). At the bottom is a large blue button labeled 'CUSTOMER LOGIN'. At the very bottom, there is a link that says 'Don't have an account? Sign up'.

Customer Login



A mobile app screen for Rider Login. At the top, there is a home icon. The title 'Welcome Back' is centered. Below it is an illustration of a person riding a scooter. Underneath the illustration are two input fields: 'Contact Number' (with a phone icon) and 'Password' (with a lock icon). At the bottom is a large blue button labeled 'RIDER LOGIN'. At the very bottom, there is a link that says 'Don't have an account? Sign up'.

Rider Login



Unlock Interface

Security

From the front-end side, the mobile application has input data validation and 2-factor authentication as security features. Additionally, data communication is based on HTTPS protocol.

Data validation

User input data is validated at this stage. Here, data validation has happened in different data fields. They are listed below,

- Name validation
- Contact number validation
- Email validation
- Password validation
- Address validation

The above fields are validated to check whether the user inputs the correct data in the correct format. By validating, it is prohibited to input invalid characters for each field. Therefore, it is not possible to have attacks like SQL injection.

2-factor authentication

After the user login, the users have to enter the OTP to unlock the device. Here we consider 2 factors. One is the order ID which is used to log in to the mobile app. Second, we ask for the OTP as the second factor.

API

For the backend services, a REST API has been developed to serve the necessary services to the mobile application and the hardware nodes. Considering several key factors, the design concern of having REST API is to map the requirements of the solution as well as the technical capabilities.

Main Technologies

- **Node.js Runtime Environment**
For the API, the underlayer technology is Node.js. With the single-threaded event-driven architecture, the application is fast enough to serve thousands of requests per second.
Used Node.js version – v10.19.0
- **Express.js**
Express is the web framework that is used to build the REST API on top of the Node.js runtime. With the ES6, Express.js is more flexible as well as it meets the design concerns of the solution.
Used Express.js version – 4.17.1
- **PM2**
For the application deployment, the PM2 package is used to serve the facilities such as load balancing, data logging, startup scripting, and many more. PM2 is an advanced process manager for production-grade Node.js applications.
Used PM2 version – 5.2.2
- **MySQL**
For the database requirements, a relational database has been used to cooperate with data. For that, MySQL has been used to serve the requirements.

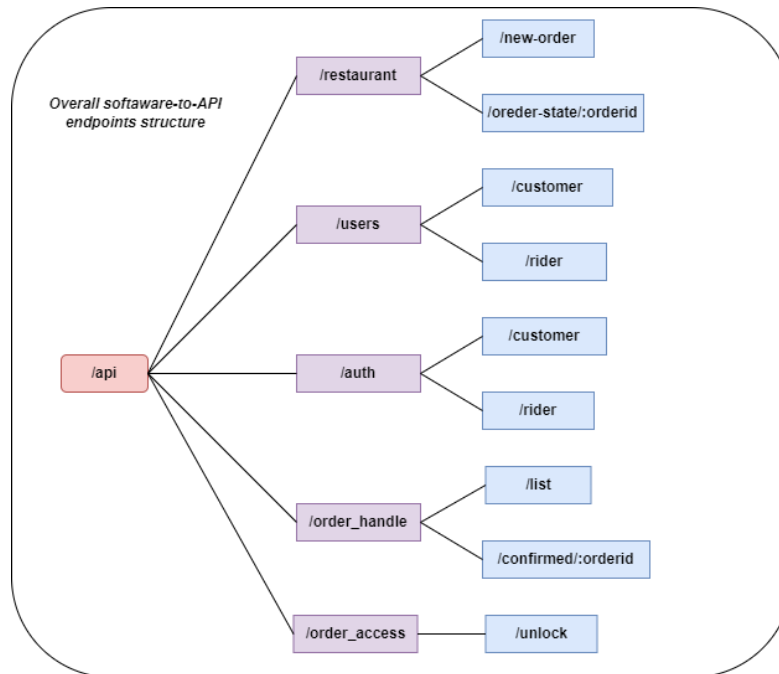
Communication Protocols

- **HTTPS**
With the SSL/TLS transport layer services, HTTPS communication protocol has been used to communicate between frontend software nodes and the backend API. Considering the security concerns, an encrypted data transfer through HTTPS is highly advantageous. For SSL certificates, OpenSSL has been used.
- **MQTT**
Between the hardware node and the API communication, MQTT messaging protocol have used with a private HiveMQ Broker configuration.
Pub-Sub messaging protocols have been used to communicate efficiently with the hardware nodes through the MQTT broker.

Architecture

The overall architecture of the API can be separated into two parts, Software components to API communication and Hardware components to API communication.

For the Software components to API communication, several endpoints have been created to service requirements. For the format of messages to pass between nodes, JSON has been used.



The overall solution is introduced as a 3rd party service for the existing food or any online delivery services. Therefore, in the API, there is a separate route for serving new orders to govern the solution.

Since the overall API architecture is based on the REST API style, all the data transactions were served through the REST pattern.

In the API, there are separate routes for customer and rider logging and registration. When receiving a new order from 3rd party delivery service, the API will send a one-time password (OTP) through SMS to the customer's mobile phone to access the carrier. For delivering the SMS to the customer, the API has used The wilio SMS service.

Upon receiving the OTP, customersomer now can log in to the app using their mobile number and enter the OTP code to unlock the carrier. But, until the rider has confirmed the delivery process, the customer won't be able to unlock it by using OTP or the RFID card.

When after a time up of 20 minutes of confirming the delivery has arrived by the rider, the OTP will be invalidated if the customer has not opened the carrier either by using an RFID card or OTP.

The underline data structure that has been used is Maps. Those are used to store the order objects on the API dynamically. When a delivery process been has completed, corresponding order objects will be cleared.

Security

Regarding the security concerns, here network and application security aspects were highly considered. To achieve such an SCEA scenario, several mechanisms have been used.

- **Input Validation**

User input validation has been done on both front-end mobile applications and in the backend API. In the backend API, user input validation has been done by using the *npm* library *joi*. If found abnormality in a user inputted data, the API will reject the data and will send a response to the client.

Ex: -

```
let schema = Joi.object({
  ...mobno: Joi.string().pattern(/^[0-9]+$/).min(10).max(10).required(),
  ...password: Joi.string().min(5).max(10).required()
});

let { error } = schema.validate(req.body);
if (error) return res.status(400).send("incorrect credentials");
```

- **JWT**

To manage the authentication and authorization aspects of the users of the system, JSON Web Tokens have been used to fulfill those. In the the API, *jasonwebtoken* library is used. When successful login of the user is, a newly created token with necessary details on the body will be sent to the user. Then, for next accessing the API, the user has to pass the issued token to validate whether the user has been authorized or not for the corresponding resource on the API.

Ex: -

```
const rtoken = jwt.sign({
  ...mobno: req.body.mobno,
  ...deviceid: result[0].deviceId,
  ...role: "rider"
}, config.get('jwtPrivateKey'));
return res.send(rtoken);
```

- **HTTPS**

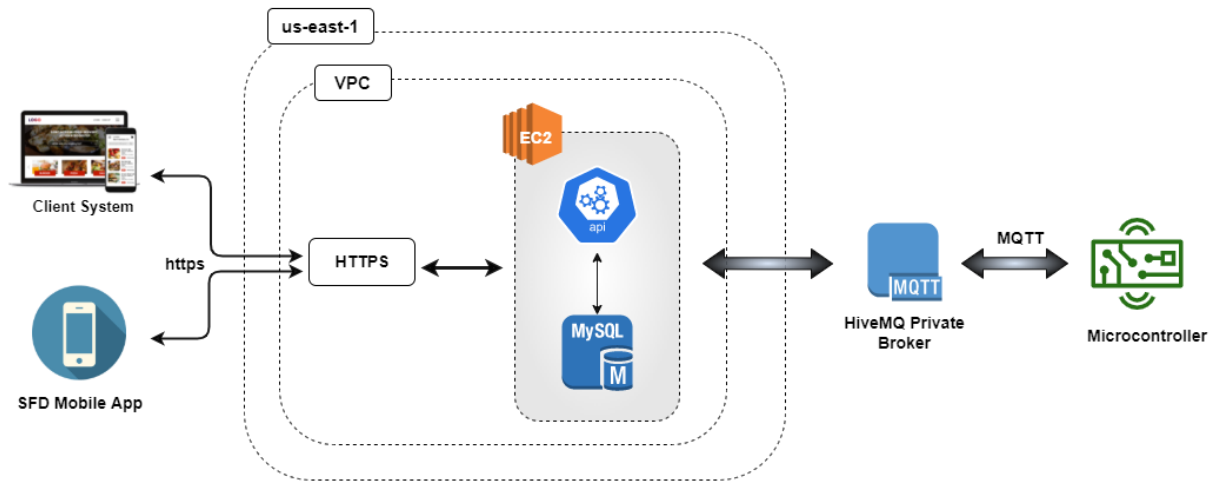
For message delivery, since adversary attacks can happen in middle, it is needed to be concerned about confidentiality, integrity, and non-repudiation of the data. For that, HTTPS protocol will be sufficiently provided the necessary facilities to ensure those. With SSH/TLS certificates, a secure connection between the software nodes can be established. We here have used SSL certificates that are signed by OpenSSL.org.

- **Password Hashing**

When storing sensitive data on the database, here we have used hashing mechanism to convert a string to a randomized custom string. With that, it is not possible to read what has been stored in the database if there is hashing mechanism is being used. To achieve that, here we have used the *npm* library called, *bcrypt* to achieve this in API.

Cloud Deployment

The overall cloud deployment structure is as follows.



For cloud deployment, AWS services such as EC2 and Firewall features have been used to run the backend services.

In a single EC2 instance which is an Ubuntu machine, on port 443, the node.js application is listening to the concurrent requests coming from several software nodes. In the same instance on port 3306, a MySQL server is running.

Through the firewall features that are provided by the AWS, all the inbound connections other than the HTTPS(443), and SSH(22) are allowed to isolate the API from the public internet.

For the MQTT communication, the HiveMQ broker has been used to deliver the messages that are published by the both API and the hardware nodes.