Smart Restaurant POS system - SafEat

Y.C.W.Arachchi [180042E], K.R.H.M.D.M.Kularatne [180330K], R.M.A.S.Rathnayake [180534N], H.M.A.M.Bandara [180060G], A.U.P.H.Athukorala [180051F], K.R.Jeyanthan [180292T], M.G.S.Thanujaya [180634V], Y.A.A.W.Rajakaruna [180508N]

***Abstract*—The system in this report is focused on developing an automated Point of Sales system that will enable customers at restaurants to order without human contact while also protecting from nay power disruptions through an uninterruptible power supply. This also allows for automated inventory management. Several hardware and software technologies have been implemented in this project including Raspberry Pi, custom designed charging module, Node MCU, Node Red, Firebase and web servers and we hope the system will be a viable alternative to the current system with human contact.**

I. Introduction

THE Covid-19 pandemic has evolved lifestyle choices and how business works in multiple avenues, may it be from fashion, grocery shopping, food delivery and public transport. Another such area which has evolved to minimize human interaction has been the restaurant industry, with many international businesses relying on their custom applications to handle orders. However, in a local context most Sri Lankan business still rely on a traditional model with direct interaction between waiters and customers. This increases the risk of being exposed to diseases from customers from other parts of the restaurant whom the same waiter might have interacted with and also involves the handling of a lot of shared material such as cash or invoices. With this problem in mind we identified the following key factors as major driving forces towards our proposed solution,

- A simple interface which even individuals without extensive technical expertise could easily use.
- Simple installation and hardware
- Attractive User Interface
- Uninterruptable Power Supply to ensure that power disruptions do not negatively impact the application
- Inventory management as an added feature
- All in one platform to handle and monitor everything that happens inside the shop.

II. Overview

Our proposed solution is a Real time end-to-end IoT, Point of Sale (PoS) system which can be installed in local restaurants with minimum effort. This proposed system consists of both hardware and software technologies and involves a custom electronic device which will be installed at each table of the restaurant.

This electronic device will involve authentication from the customer using a QR code while for loyalty customers, this the device also offers RFID authentication through a restaurant provided loyalty/membership card. Following authentication, the customer will be led to the application on an attached touch screen or which can be accessed through his mobile phone through which he can place orders, see which items are in stock, see the prizes of the menu items and other functionalities. Power to the system is given through an Uninterruptible Power Supply which accommodates two Li-ion batteries and USB charging.

All these individual electronic devices at each table will be connected to a central controlling system which will be managed by the restaurant staff and which will communicate which tables require which orders. This also enables the provision of real time inventory management as items will be updated as soon as orders are placed.

Our belief is that this system will offer an enhanced customer experience while also minimizing human interaction and thus reducing the risk of the spread of infection. The automated inventory management feature also offers restaurant owners a degree of comfort and enables them to make relevant inventory related decisions promptly.

III. Methodology

A. System architecture

The following image [Fig. 1] explains the system architecture of the SafeEat system with the Raspberry Pi acting as the central hub, while custom electronic devices are connected to the central hub. The images at top extreme display the NodeRed dashboard.

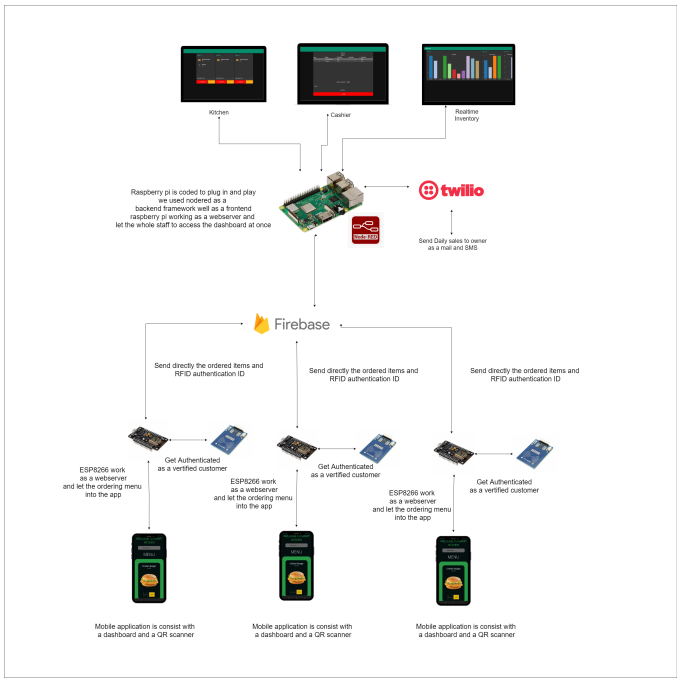

Fig. 1: The system architecture of the SafEat system

B. List of system devices and functions

Electronic device

Each table in the restaurant will be equipped with a custom electronic device which consists of an NodeMCU ESP8266 module which will function as the device handling the customer side application of the system. This NodeMCU was configured to host a ESP8266 Web Server which hosts the the web application (more details given in system implementation)

The device also accompanies a RFID reader which offers RFID authentication to loyalty/membership cards which are owned by customers. This RFID module is also connected to the ESP8266 module and coded using available Arduino libraries.

The differentiating factor offered by the electronic device is the attached Uninterruptible Power Supply which makes the device resistant to power disruptions. To store electric energy two 18650 Li-ion batteries are used in parallel configuration with a P-channel MOSFET being used to switch between battery and charger power as required. As a result, the customer can keep the charger connected to device and when power is available LOAD is driven by the charger power and at power failure LOAD is driven by battery power. The specialty of the system is that LOAD can be driven from charger while charging too and as a result power is managed intelligently.

The ESP module possesses a 3.3V internal regulator while our power supply provides 5V when the charger is providing power, and a voltage in the range of 3.7V-4.2V when driven by batteries.

To control the charging of the batteries, TP4056 constant current/constant voltage linear charger IC is used which features the following,

- Constant current / constant voltage charging method
- 2.9V trickle charge threshold for deeply discharge batteries
- Automatic recharge
- Overcharge protection.

DW01A Battery Protector IC is used along with FS8205A dual N channel MOSFET to obtain

- Short circuit protection
- Over current protection
- Over discharge protection

To identify the status of batteries, two indicator LEDs are used.

- When charging, only red LED is glowing.
- When battery is charged, only blue LED is glowing.
- If both LEDs are off, then input voltage is too low.

All the relevant circuits were designed and PCB layout was done using Altium Designer. The initial intention was to build a compact PCB utilizing Surface Mount (SMD) components, however the current situation and associated complications did not allow for this option. Ultimately, the PCB design

consists of two separate PCB designs, one using SMD components while the other uses through hole components. Refer Fig. 2, Fig. 3, and Fig. 4 for the PCB layouts.

[Fig. 5] 3-D design of the enclosure was completed using SolidWorks and the enclosure was 3-D printed.

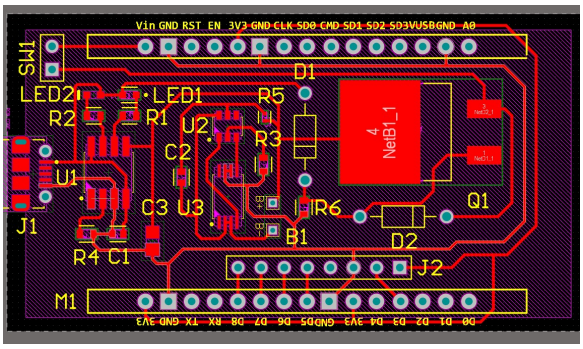

Fig. 2: PCB layout of the charging module and Node MCU

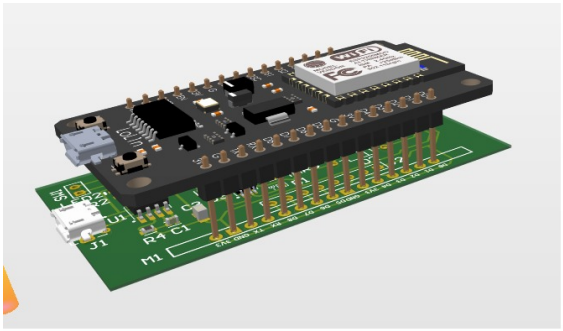

Fig. 3: 3D view of the charging module and Node MCU

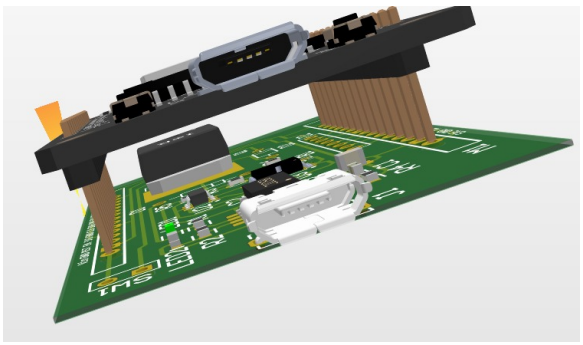

Fig. 4: 3D view of the charging module and Node MCU

Raspberry Pi

All of the custom electronic devices send data to a Raspberry Pi 3B+ Board which acts as the

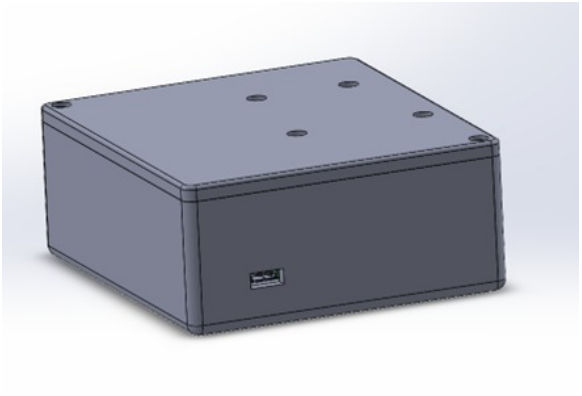

Fig. 5: 3D model of the device developed using Solidworks

central processor of the entire system. The onboard processor helps aggregate all the data sent from individual devices and communicates to the kitchen staff the dishes that need to be prepared while also enabling automated inventory management by tracking the dishes in stock/out of stock which can also be notified to the customers as required

IV. System Implementation

A. *Front-end and back-end*

We have used the ESP8266 web server to display the front-end. The web-app is accessed using the IP address of the SafEat device. The front-end is developed using Html, CSS, and JavaScript. The landing page of the web-app will indicate the customer to scan the RFID card to continue to the menu page to order their meal [Fig. 6]. When the RFID card is scanned the back end will load the menu page to the customer [Fig. 7]. The special characteristic of the front-end of the menu page is that we have not hard coded the menu items. Rather, we have used the back end to inject menu items to the menu page. The advantages of using this method are, we can update our menu without having to rewrite the ESP8266 repeatedly and it saves up the flash memory of the ESP8266. We just need to update the Firebase firestore to update the menu of the restaurant.

The back end consists of Google Firebase, and NodeRed. We use Google Firebase as the database for our application and NodeRed as the dashboard of the Kitchen, Cashier, and the Inventory. Using JavaScript scripts, we retrieve the menu items from the firebase to the front end and send the order details to the firebase. Using NodeRed we retrieve the order details for the Kitchen, the cashier, and

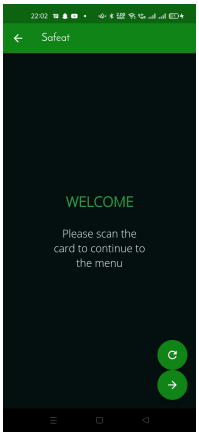

Fig. 6: Landing page of the web-app

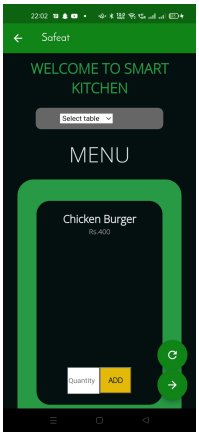

Fig. 7: Menu page of the web-app

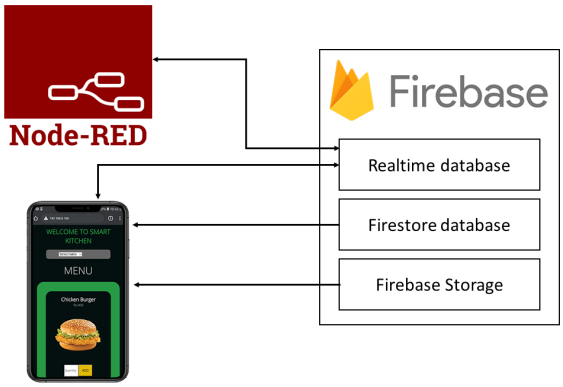

Fig. 8: The menu item details are read from the *Firestore* database to the web page. Data is read and written to the *Realtime* database from both the web page and Node-Red dashboard. *Firebase storage* is used to save all the multimedia files in a cloud storage.

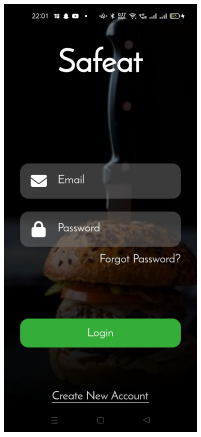

Fig. 9: Authentication page of the mobile application

send the daily sale to the owner and keep track of the inventory of the restaurant. Fig. 8 The following diagram indicates how the web-app, google firebase, and NodeRed dashboard is connected to each other

B. Mobile application

The mobile application consists of an authentication page where the user has to enter their email/username and password [Fig. 9].

The app also includes capabilities to scan QR codes [Fig. 10, Fig. 11](with flash on/off) which allows the user to navigate to the correct WebView page which is hosted on the ESP8266 web server. From there onward the user is free to choose his preferred menu items.

The application was tested using an Android phone but it can also be installed on a iOS platform.

V. System functionality achieved

A. Node-Red dashboard

The following figures show the Node-Red dashboard along with the required data by restaurant management. The application is currently configured for 3 tables, but the number can be adjusted as per user requirements. The Kitchen [Fig. 12], Cashier [Fig. 13], Inventory management [Fig. 14], and Billing [Fig. 15] functions are clearly displayed

B. User experience

The application offers the customer a comfortable user experience all the way from the authentication process to the final ordering process with relevant information as price and availability of dishes readily available to the user. Extensive technical knowledge is not required for the customer to have the

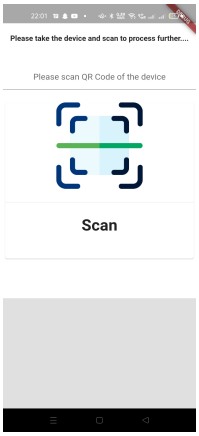

Fig. 10: QR scanning from the mobile application

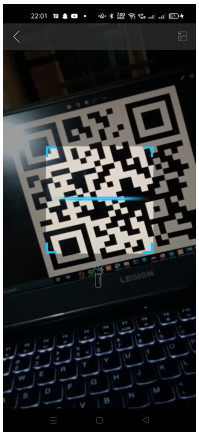

Fig. 11: Scanning the QR code

optimum experience as the interface is simple and instructions are straightforward with information clearly displayed [Fig. 16].

VI. Conclusion

In our implementation our main focus was on sending the data to cloud for quick access while keeping data security and integrity intact by protecting the system from power outages. The Firebase real time database along with the node-red palettes processes data in less than 1 second which makes the system extremely reliable and accurate. The RFID card reader adds an extra dimension of security as well as an added feature for potential users. As the implementation was successful with little to no human intervention we believe that our system offers a viable alternative to the current physical ordering systems in place, although further work will be required to refine the system.

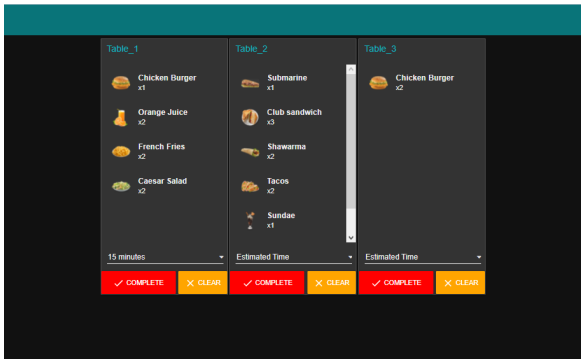

Fig. 12: Node-Red Kitchen dashboard

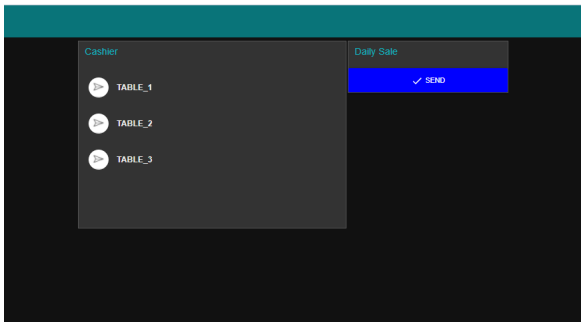

Fig. 13: Node-Red Cashier dashboard

References

1) Node-Red documentation
2) List of videos in a youtube playlist which we used for our implementation
3) Google Firebase documentation
4) Github Repository
5) Firebase arduino documentaion
6) Twilio documentation

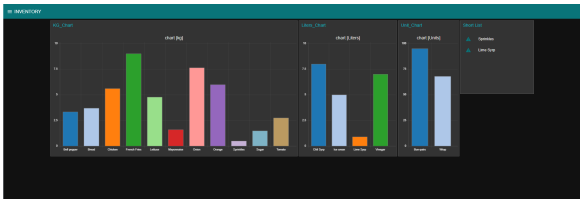

Fig. 14: Node-Red Inventory dashboard

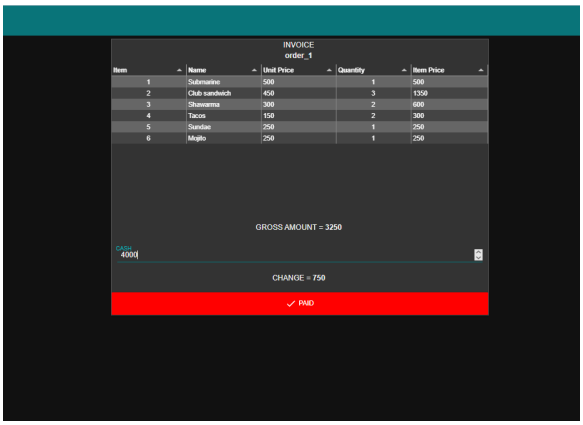

Fig. 15: Bill

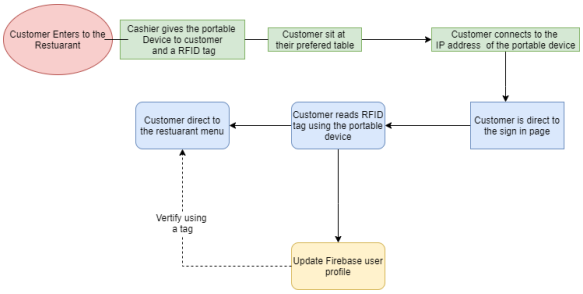

Fig. 16: The user experience of the application