Cloud Checkout: Documentation Package

Jeton Ajeti
Ronnie Arvanites
Tony Oprisek

Capstone Design II
May 8, 2019

Table of Contents

Work Package

Cloud Checkout

Prepared by T.Oprisek, R.Arvanites, J.Ajeti. May 8, 2019

Objectives

This work package outlines the different items and specifications produced in Cloud Checkout. Final deliverables for hardware, software, and project resources are included.

Period of Performance

January 16, 2019 to May 3, 2019

Specifications

Item	Specification	Units
Central Controller	See Task 1.1	1
NFC Controller	See Task 1.2	1
RFID Controller	See Task 1.3	1
Database	See Task 1.4	1
Mobile App	See Task 1.5	1
RFID Tags	See Task 1.6	50

Tasks

1.1 Central Controller

1. Connect Raspberry Pi to a monitor, keyboard, and mouse.
2. Plug Raspberry Pi into power outlet.
3. Download Arduino and the Spark_Fun_Simultaneous_RFID_Tag_Reader_Library from Github.
4. Download libnfc Library.
5. Connect RFID Controller via USB.
6. Free the UART on the Raspberry Pi
7. Connect NFC Controller via UART

1.2 NFC Controller

1. Solder pins to the PN532 NFC/RFID Controller Breakout Board.
2. Connect the T-Cobbler Plus to the Raspberry Pi.
3. Connect the NFC Controller's GND to GND on the T-Cobbler Plus.
4. Connect the NFC Controller's TXD to TXD on the T-Cobbler Plus.
5. Connect the NFC Controller's RXD to TXD on the T-Cobbler Plus.
6. Connect the NFC Controller's 5.0V to 5.0V on the T-Cobbler Plus.
7. Turn SEL0 and SEL1 off on the NFC Controller.

1.3 RFID Controller

1. Solder pins to the SparkFun Simultaneous RFID Reader.
2. Connect the SparkFun Simultaneous RFID Reader to the RedBoard.
3. Connect the UHF RFID Antenna to SparkFun Simultaneous RFID Reader via coaxial.
4. Connect the REDBoard to the Raspberry Pi via USB.

1.4 Database

1. Create a Google account.
2. Go to the firebase console in any browser.
3. Add a project.
4. Click database on the menu to the left.
5. Create your database schema.

1.5 Mobile App

1. Download Xcode.
2. Create a new application.
3. Obtain a paid developer license from Apple.
4. Obtain a NFC Tag Reading certificate from Apple.

1.6 RFID Tags

1. Remove one RFID tag from package.
2. Program the RFID tag with information using the the SparkFun Simultaneous RFID Reader and the Example5_Write_UserData program, in Spark_Fun_Simultaneous_RFID_Tag_Reader_Library.

Resources

Parts: RaspberryPi, Arduino Uno, PN532 NFC/RFID Controller Breakout Board, T-Cobbler Plus, SparkFun Simultaneous RFID Reader - M6E Nano, Sparkfun RedBoard, UHF RFID Tags, UHF RFID Antenna, UHF RFID Cable (TNC to RP-SMA), Interface Cable RP-SMA to U.FL, and 3 USB Cables.
Tools: Soldering iron and solder

Deliverables

Hardware

50 RFID Tags
1 Central Controller
1 NFC Controller
1 RFID Controller including antenna

The hardware modules will each work together. The Central Controller will allow each module to communicate to each other. Once the tags are scanned the RFID Controller will send the RFID data to the Central Controller via USB. The Central Controller will then save the data to a text file. After, the Central Controller will then send the data to the NFC Controller. Finally, the user can scan their phone with our mobile app open.

Software

The software is a mobile application written in Swift. In production, the app will be available to download in the Apple App Store. The mobile application allows users to see products available in the store, confirm their checkout, and see past purchases.

Design

Hardware

NFC Controller

For the NFC Controller, the PN532 NFC/RFID Controller Breakout Board was used, seen in figure 1. The PN532 NFC/RFID Controller Breakout Board works in all four NFC modes, read mode, write mode, peer-to-peer mode, and card emulation mode. It is also compatible with Arduino and Raspberry Pi. The PN532 NFC/RFID Controller Breakout Board can use UART, I2C, or SPI for communication.

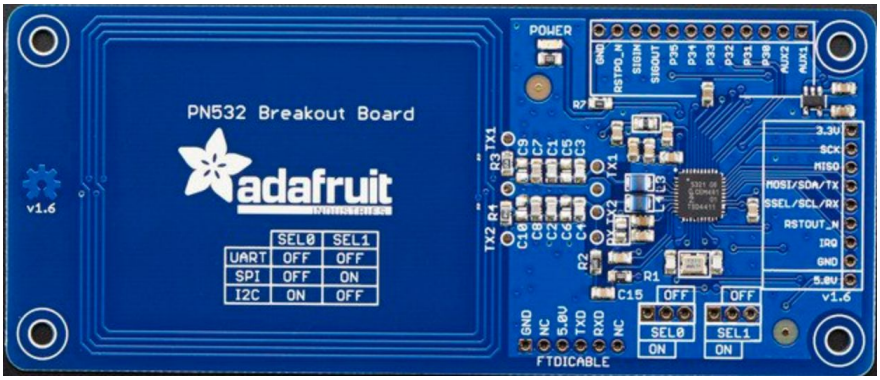

Fig. 1. PN532 NFC/RFID Controller Breakout Board

The PN532 NFC/RFID Controller Breakout Board is connected to the Raspberry Pi via UART connection, using a T-Cobbler Plus and jumper wires. The T-Cobbler Plus can be seen in figure 2, and the jumper wires can be seen in figure 3.

Fig. 2. T-Cobbler Plus

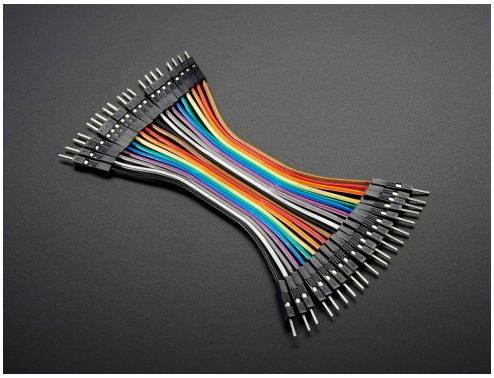

Fig. 3. Jumper Wires

One end of the T-Cobbler Plus was plugged into the Raspberry Pi and the other end was connected to a breadboard. On the PN532 NFC/RFID Controller Breakout Board, the GND, 5.0V, TXD, and RXD were connected to the GND, 5.0V, TXD, and RXD on the T-Cobbler Plus, respectively. The SEL0 and SEL1 bits were set to off by connecting the last two pins to each other for each one. The PN532 NFC/RFID Controller Breakout Board connected to the T-Cobbler can be seen in figure 3 and 4. The NFC Controller connected to the Raspberry Pi can be seen in figure 5.

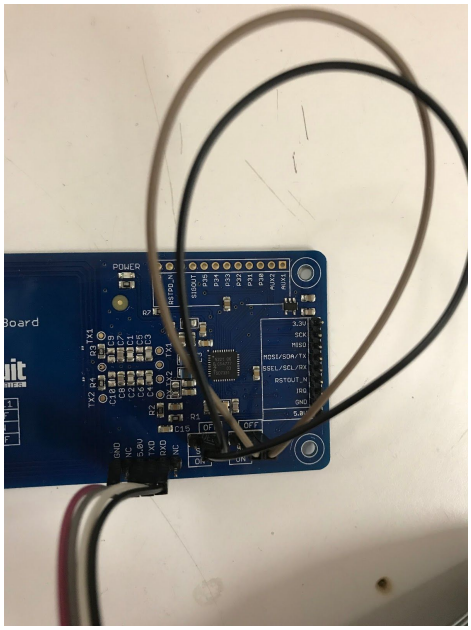

Fig. 3. PN532 NFC/RFID Controller Breakout Board Connection

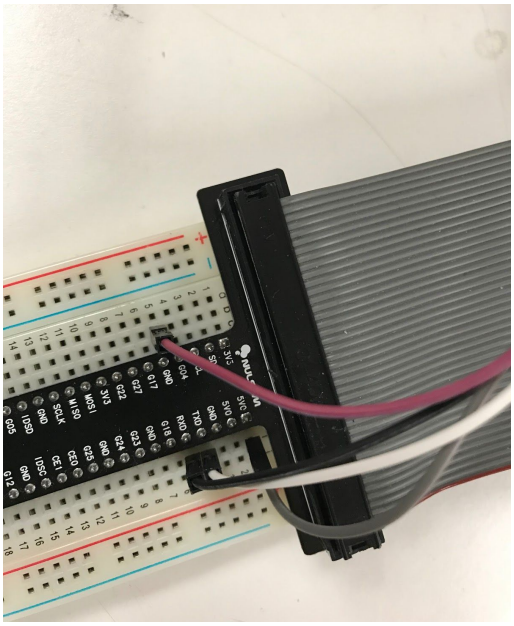

Fig. 4. T-Cobbler Connections

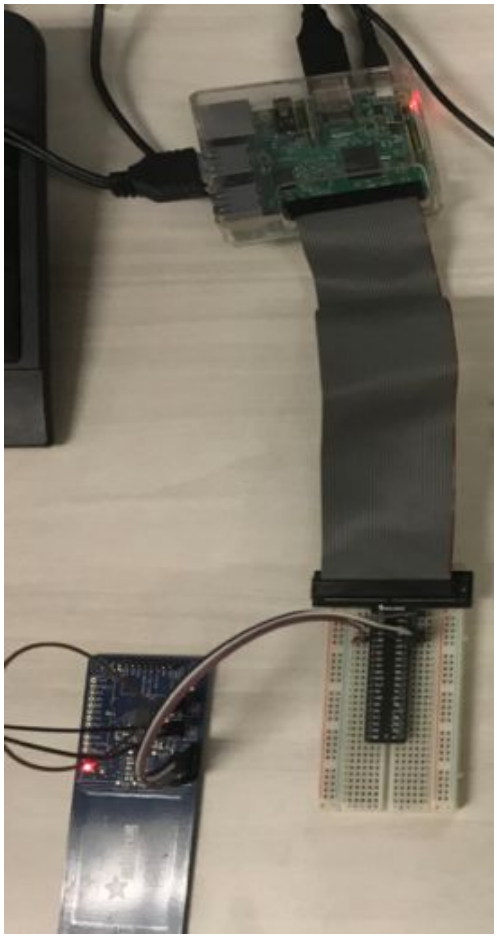

Fig. 5. NFC Controller connected to Central Controller

RFID Controller

For the RFID Controller, the Sparkfun RedBoard, SparkFun Simultaneous RFID Reader - M6E Nano, and UHF RFID Antenna were used. The SparkFun Simultaneous RFID Reader - M6E Nano reads EPCglobal Gen 2 tags at up to 150 tags per second. It also writes tags at 80 milliseconds. The SparkFun Simultaneous RFID Reader - M6E Nano has an adjustable power output from 0dBm to 27dBM, allowing users to increase the range of the antenna attached. The Sparkfun RedBoard, SparkFun Simultaneous RFID Reader - M6E Nano, and UHF RFID Antenna can be seen in figure 6, 7, and 8 respectively.

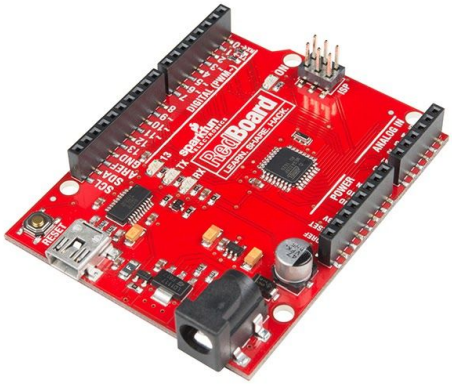

Fig. 6. Sparkfun RedBoard

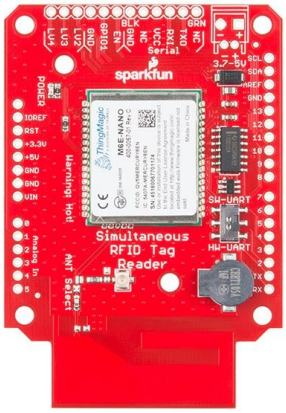

Fig. 7. SparkFun Simultaneous RFID Reader - M6E Nano

Fig. 8. UHF RFID Antenna

The SparkFun Simultaneous RFID Reader - M6E Nano was connected to the Sparkfun RedBoard via the GPIO pins. The UHF RFID Antenna was connected to the SparkFun Simultaneous RFID Reader - M6E Nano via an Interface Cable RP-SMA to U.FL, seen in figure 9. The UHF RFID Antenna connected to the SparkFun Simultaneous RFID Reader - M6E Nano can be seen in figure 10.

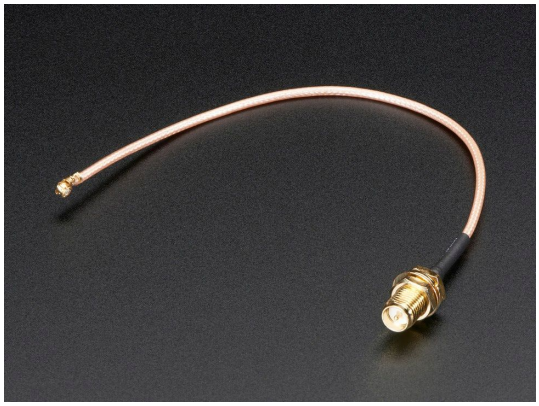

Fig. 9. Interface Cable RP-SMA to U.FL

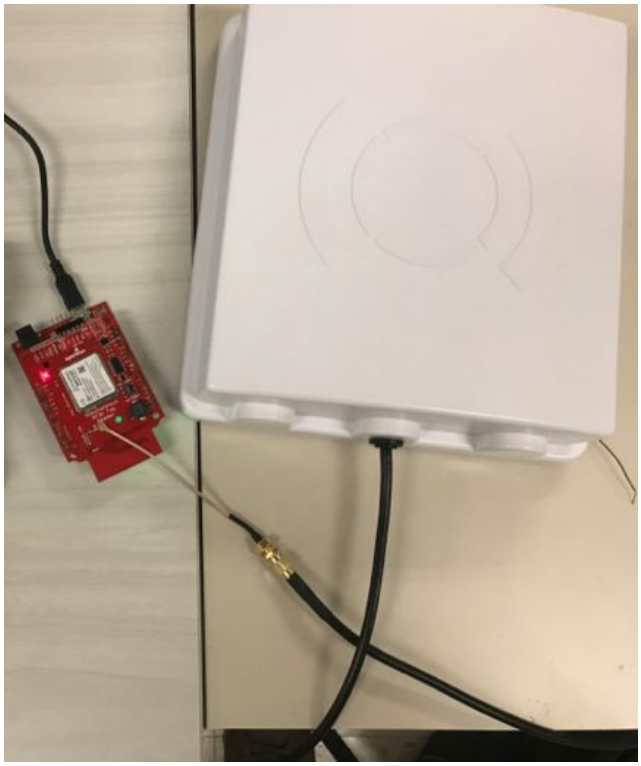

Fig. 10. UHF RFID Antenna connected to SparkFun Simultaneous RFID Reader - M6E Nano

The Sparkfun RedBoard was connected to the Raspberry Pi via a USB cable. This allowed data to be transferred between the RFID Controller and the Central Controller. The connection between the RFID Controller and the Central Controller can be seen in figure 11.

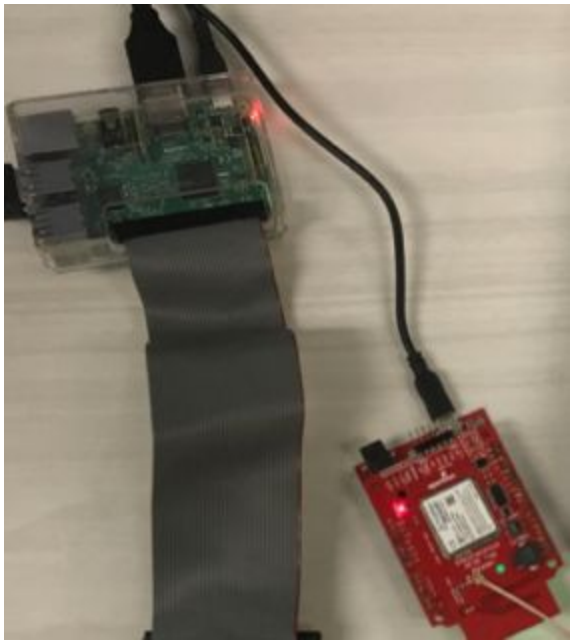

Fig. 10. RFID Controller connected to Central Controller

Software

Central Controller

For the Central Controller, a Raspberry Pi was used. The Raspberry Pi can be seen in figure 11.

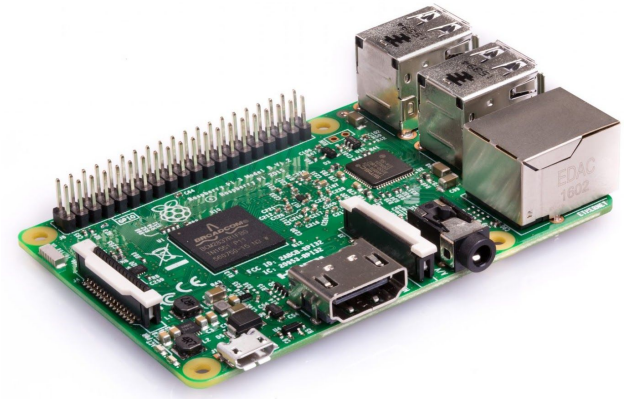

Fig. 11. Raspberry Pi

The Central Controller connects the hardware with the software in our system. On the Central Controller our main program, written in C, runs in an infinite loop calling other programs that interact with the RFID Controller or the NFC Controller. The main program, first runs the RFID Controller program, written in python, to get information from the RFID Controller and save it to a text file. Next, the main program calls the NFC Controller program, written in C, to emulate a tag on the NFC Controller. Finally, the main program restarts the loop. The flow chart for the main program can be seen in figure 12.

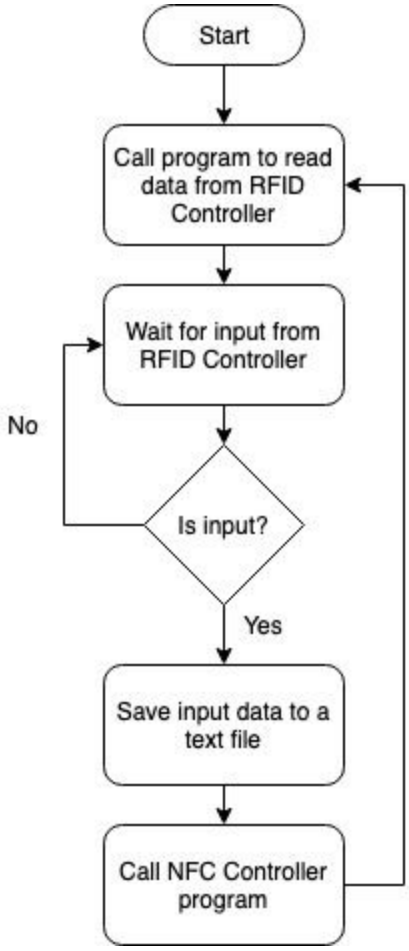

Fig. 12. Main Program Flow Chart

Database

For the database, Google Firestore database was used. Google Firestore database is a cloud database so no servers need to be set up. The database is a noSQL database so collections and documents are used. In our system, the product information and the user past purchases were stored in the database. Our database has a two main collections, Stores and Customers. Each

document in the Stores collection, stores the information about the store such as location, name, manager name, and a subcollection called Store Products are saved. Each document in the subcollection of Store Products, stores the SKU number, product name, quantity, price, and category. Each document in the Customers collection stores the customers UID, from google authentication, and a subcollection called Past Purchases. The UID is used to get all the users information such as name, birth date, and email. Each document in the subcollection Past Purchases contains the purchase data, store, and a subcollection of called Items. Each document in the Items subcollection stores the SKU number, product name, category, ImageURL, quantity and price. The database schema diagram can be seen in figure 13.

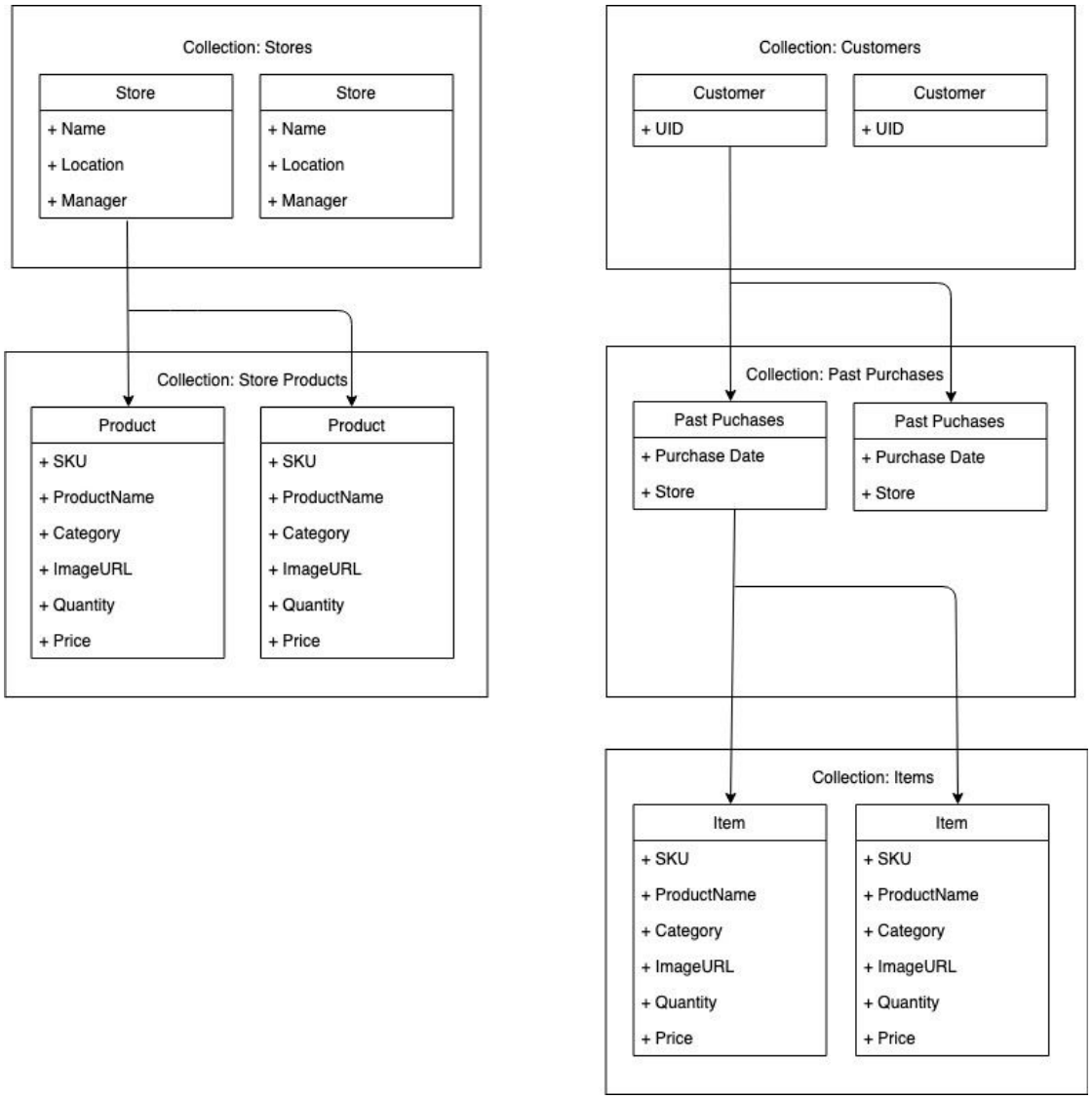

Fig. 13. Database Schema

Mobile Application

The mobile application is used to allow users to check out of the store. In production, the application will be available for anyone to download from the Apple App Store. The mobile app is a critical component in interacting with the checkout system. Once the customer has bagged the items and is about to leave the store, she will be required to tap the checkout counter with their phone. The app will read the items the system has detected and encoded into the NFC Breakout board once the phone is within the vicinity of the NFC board. The app receives the ID numbers, which identify each product in the database, and stores them in an array named **tokens**. To fetch the the information regarding the products the user is about to purchase, we make a call to the database based on the ID numbers stored in the array. The function below encapsulates the logic programmatically.

```
func FetchProductData(tokens: [String]) { //Fetch product data from the database.
    let myGroup = DispatchGroup()
    for token in tokens { //Tokens store SKU numbers of the products a customer has in the cart.
        myGroup.enter()
        let docRef = Firebase_Database.collection("Stores").document("Store One").collection("Store Products").document(token) // Fetch the product info for a specific
            product.
        docRef.getDocument { (document, error) in
            if let document = document, document.exists {
                let dataDescription = document.data()
                let product = Product(dictionary: dataDescription!, SKU: document.documentID)
                products.append(product) //Append product data to an array.
                myGroup.leave()
            } else {
                print("Document does not exist")
            }
        }
    }
    myGroup.notify(queue: .main) {
        self.performSegue(withIdentifier: "linkToCheckout", sender: self) // Once all the data has been retrieved form the FB, segue to the following VC.
    }
}
```

Fig. 14. Database Schema

The function above consists of a loop which gets the ID of each product an makes a call to the database asking to fetch the product name, price, URL image, etc. Once the data has been collected, we proceed to the next step. In the next step, the function continues to the subsequent VC (View Controller) which displays to the user the products they have in their cart and enables them to pay. The UI for confirming your purchase can be seen in figure 14.

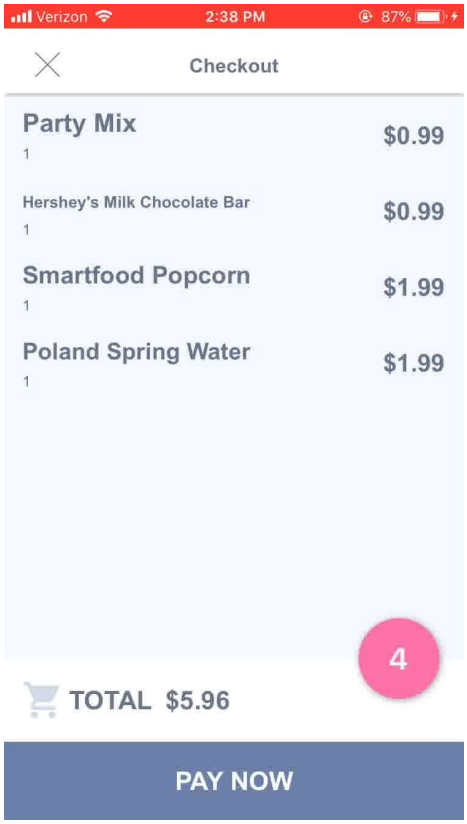

Fig. 15. Confirm Purchase UI

The mobile application also allows users to see products available in the store, including the quantity left. The products are displayed based on categories each product is under. The UI for the categories, seen in figure 15, allows users to tap a category. Once the user taps on a category, the products for that category are shown, seen in figure 16.

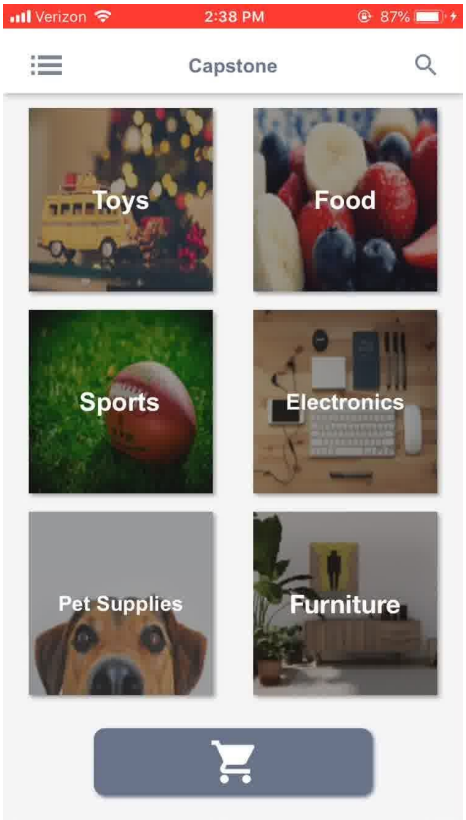

Fig. 16. Product Category UI

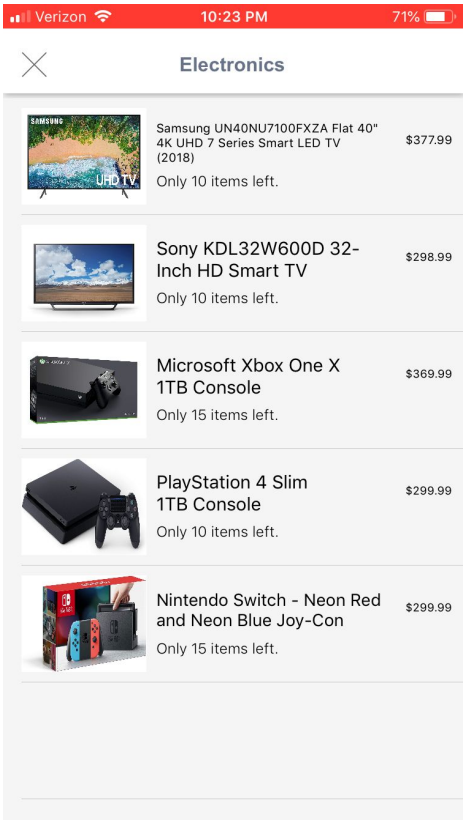

Fig. 17. Product List UI

System Architecture

System architecture consists of the critical components which power the express checkout system. The components can be separated into two categories: hardware and software. The hardware includes the RFID Reader, NFC Breakout Board, and a server connecting the two - Raspberry Pi. The software, on the other hand, includes the mobile app which allows the user to interact with the system and collect more information on the store they are purchasing and the products in stock. In addition, software includes the authentication of the users, payment processes, database, etc. The overall system operates in a for of a cycle. It starts with the customer approaching the checkout counter with at least one product (every product has a RFID tag, which contains the ID number, attached to it) in their cart. The RFID reader scans the products in the cart and send it to the server. The server encodes the data and to the NFC board in an NDEF format. Subsequently, the cycle completes when the user taps their phone to the counter to get a bill before leaving the store.

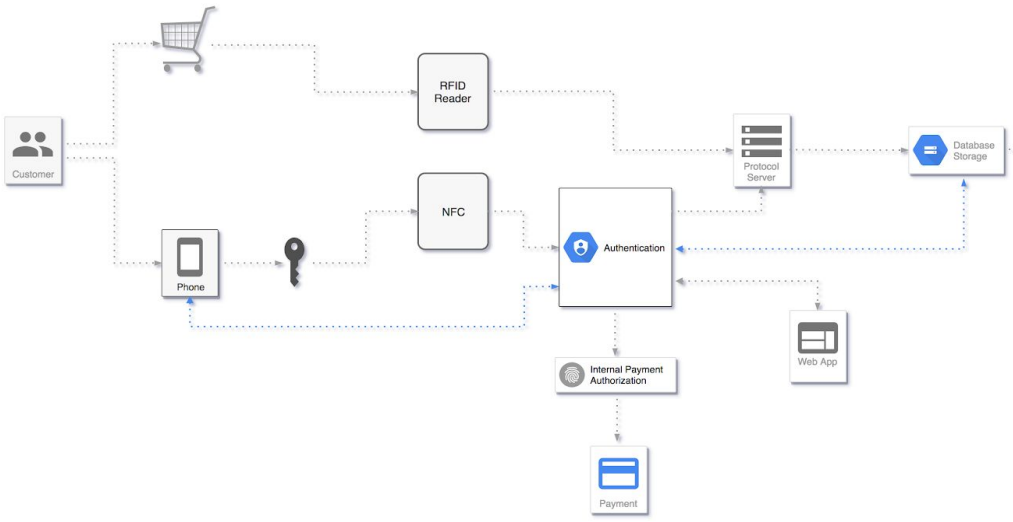

Fig. 18. System Architecture.

Testing

Hardware

For all the hardware testing done on the Cloud Checkout, white box testing was used. White box testing was used over black box testing because all the internal structure, design, and implementation were known. Each hardware component was tested individually and integrated in the system.

NFC Controller

The NFC Controller was tested both separately and integrated with the overall system. First, a unit test was performed on the NFC Controller to seeing if the SKU number from the text file can be displayed on the NFC reader app. To set up this test, a text file was saved with a six digit SKU number on the Raspberry Pi. The NFC Controller was also connected to the Raspberry Pi. First, the NFC card emulator program was run. Next, the NFC Breakout Board was scanned with a IPhone running an NFC reader app. After, the data read in the NFC reader app was compared to the SKU number located in the text file. On the IPhone the SKU number was the same SKU number located in the text file. Since the SKU numbers matched, the NFC Controller passed the unit test. The documentation for the NFC Controller unit test can be seen in figure 17. The NFC Controller was also tested integrated into our system and it passed all the tests.

Test Writer: Ronnie Arvanites						
Test Case Name: NFC Card Emulator				**Test ID:** NFCCE01		
Description: Testing to see if the NFC card emulator works and displays SKU numbers form a text file on the mobile app				**Type:** White Box		
Test Information						
Name of Tester: Ronnie Arvanites and Jeton Ajeti				**Date:** 1/30/2019		
Setup: NFC Breakout Board should be connected to the Raspberry Pi and a SKU number should be saved into a text file						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Start the NFC Card Emulator program	The program should run without errors	✓			The program ran without any errors.
2	Scan the NFC Breakout Board with an IPhone	The mobile app should say the tag was read	✓			The IPhone was held very close to the Breakout Board because the max read distance is only up to 4 cm.
3	Check the mobile app to see what data was read	The SKU number saved in the text file should be displayed in the mobile app	✓			The SKU number was displayed on the mobile app and it matched the SKU number in the text file.
Overall Test Results: Passed						

Fig.17. NFC Card Emulator Unit Test

RFID Controller

The RFID Controller was tested both separately and integrated with the overall system. Two unit tests were perform to test the RFID Controller. The first unit test, was conducted to see if the RFID Controller was able to read one tag. To set up this test, one of the RFID tags was programmed with a SKU number and the UHF RFID Antenna was connected to the SparkFun Simultaneous RFID Reader - M6E Nano. First, the RFID reader program was run. Next, the programmed RFID tag was held about two feet above the antenna. After, the console was checked to see if the SKU number was displayed. The SKU number was displayed on the console confirming that the test was passed. The read rate was under 5 seconds which was expected because only one tag was read. The test documents can be seen in figure 18.

Test Writer: Ronnie Arvanites						
Test Case Name: RFID Tag Reading				**Test ID:** RR01		
Description: Testing to see if the RFID reader reads one tag				**Type:** White Box		
Test Information						
Name of Tester: Tony Oprisek					**Date:** 12/5/2018	
Setup: RFID tag should be programed with a SKU number and the RFID reader should be connected to the antenna						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Start the RFID reader program	The program should run without errors	✓			The program ran without any errors.
2	Hold RFID tag above antenna	The RFID reader should read the tag	✓			The tag was held about two feet above the antenna. The reading distance for our antenna is up to 12 feet.
3	Check the console to see the output	The SKU number programed on the tag should be displayed in the console	✓			The SKU number was displayed on the console after a few seconds.
Overall Test Results: Passed						

Fig.18. RFID Tag Reading Unit Test.

The second unit test was performed on the RFID Controller to test to see if the RFID Controller can read multiple tags. To set up this unit test, two RFID tags were programmed with the same SKU numbers but different UIDs. The UHF RFID Antenna was also connected to the SparkFun Simultaneous RFID Reader - M6E Nano. First, the RFID reader program was run. Next, both the RFID tags were held about two feet above the antenna. After, the console was checked to see if the SKU numbers were displayed. Both SKU numbers were displayed, which showed that RFID program can read multiple tags. The documentation for this unit test can be seen in figure 19. The integration test was also performed on the RFID Controller and the RFID Controller passed the integration test.

Test Writer: Ronnie Arvanites						
Test Case Name: RFID Tag Reading				**Test ID:** RR03		
Description: Testing to see if RFID reader reads multiple tags for v2 of RFID reader program				**Type:** White Box		
Test Information						
Name of Tester: Tony Oprisek				**Date:** 4/6/2019		
Setup: Two RFID tags should be programed with the same SKU numbers but different UIDs. The antenna should be connected to the RFID reader						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Start the RFID reader program	The program should run without errors	✓			The program ran without any errors.
2	Hold both RDIF tags above antenna	The RFID reader should read both tags	✓			The tags were held about two feet above the antenna. The reading distance for our antenna is up to 12 feet.
3	Check the console to see the output	The SKU numbers programed on each tag should display in the console	✓			The SKU numbers were both displayed on the console after a few seconds.
Overall Test Results: Passed						

Fig.19. RFID Tag Reading Unit Test for RFID Reader Program V2

Software

For all the Software testing, we chose to use white box testing again. Each software component was tested individually and integrated into the overall system.

Mobile Application

The mobile application was test both individually and integrated into with the overall system. The first test unit test we performed was the user authentication test to see if the mobile app allows users to create an account with google authentication. To set up the test, the mobile app was running in a IPhone simulator in Xcode and the Firestore console was open in a web browser. First, information was entered in our app to represent a user. Next, the confirm button

was tapped to confirm the user's information was entered correctly. In the debugging window in Xcode, a success message was display confirming that the data was sent to Firestore successfully. After, the Firestore console was checked to see if the user was added. The user's data was displayed so the mobile app user authentication test was passed. The mobile app user authentication unit test documentation can be seen in figure 20.

Test Writer: Ronnie Arvanites						
Test Case Name: Mobile App User Authentication				**Test ID:** MA01		
Description: Testing to see if the mobile app allows user to create an account with google authentication				**Type:** White Box		
Test Information						
Name of Tester: Jeton Ajeti				**Date:** 2/6/2019		
Setup: IPhone simulation should be running with our app opened. Firestore console should be open to see if the user was created						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Enter user information into our mobile app	Our app should allow the user to add their information into our app	✓			The program ran without any errors.
2	Tap confirm to create the user's account	The information the user just entered should be sent to Firestore	✓			The information was sent successfully as seen in the debugging console in Xcode.
3	Check the Firestore console to see if the user's account was created	The user's account should be seen in the Firestore console	✓			The user's data was seen in the Firestore console.
Overall Test Results: Passed						

Fig.20. Mobile App User Authentication Unit Test

The second unit test was performed to see if data from the NFC Card Emulator can be displayed in the mobile application. To set up this test, the IPhone was connected to the computer running Xcode. The NFC Controller was also running. First, the app was built and loaded onto the IPhone. Next, the shopping cart button was tapped on the mobile application. After, the NFC Controller was scanned with IPhone. Finally, the console was checked to see if the SKU number on the NFC Controller was displayed. The SKU number was displayed in the console so the unit

test was passed. The documentation for this test can be seen in figure 21. The mobile application was also tested integrated into the overall system, were it passed all the tests.

Test Writer: Ronnie Arvanites						
Test Case Name: Mobile App NFC Read				**Test ID:** MA02		
Description: Testing to see if the mobile app reads data from the NFC Card Emulator				**Type:** White Box		
Test Information						
Name of Tester: Ronnie Arvanites and Jeton Ajeti				**Date:** 4/10/2019		
Setup: IPhone should be connected to computer running Xcode. The NFC Card Emulator should also be running						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Run the code in Xcode	The app should open on the IPhone	✓			The program was compiled without any errors
2	Tap shopping cart button on bottom of app	Our app should tell the user it is ready to scan	✓			The program ran without any errors.
3	Scan the NFC Card Emulator with the IPhone	The mobile app should have read some data	✓			The data read was displayed in the console.
4	Check the console in Xcode to see if the SKU number on the NFC Card Emulator was read	The data displayed in the console should be the SKU number that was on the NFC Card Emulator	✓			The SKU number was displayed in the console.
Overall Test Results: Passed						

Fig.21. Mobile App Read Data Unit Test

Database

The database was tested both individually and integrated into with the overall system. The unit test was performed to see if the data from Firestore could be retrieved in the mobile application. To set up this test, the IPhone should be connected to the computer running Xcode. The NFC Controller should also be running with three SKU numbers from the database. First, the app was built and loaded onto the IPhone. Next, the shopping cart button was tapped on the mobile

application. After, the NFC Controller was scanned with IPhone. Finally, the mobile application was checked to see if the items were displayed. Since the items were displayed, the database unit test was passed. The documentation for this test can be seen in figure 22. The database was also tested integrated into the overall system, were it passed all the tests.

Test Writer: Ronnie Arvanites						
Test Case Name: Database Data Retrieval				**Test ID:** DB01		
Description: Testing to see if the data stored in Firestore can be retrieved in the mobile app				**Type:** White Box		
Test Information						
Name of Tester: Jeton Ajeti				**Date:** 4/16/2019		
Setup: IPhone should be connected to computer running Xcode. The NFC Card Emulator should also be running with three SKU numbers.						
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**N/A**	**Comments**
1	Run the code in Xcode	The app should open on the IPhone	✓			The program was compiled without any errors
2	Tap shopping cart button on bottom of app	Our app should tell the user it is ready to scan	✓			The program ran without any errors.
3	Scan the NFC Card Emulator with the IPhone	The mobile app should have read some data	✓			The data read was displayed in the console.
4	Check the mobile app to see if the items are displayed	The mobile app should show the data of the SKU numbers scanned	✓			The data was displayed on the mobile app.
Overall Test Results: Passed						

Fig.22. Firebase Data Retrieval Unit Test

User Manual

An effective checkout system revolves around security, reliability, easy of use, and speed. We developed Cloud Checkout to meet this requirements in nearly any retail environment. We utilize UHF RFID technology to avoid lengthy lines and accelerate the checkout process faster than traditional barcode scanners. Each system would be implemented into a store for every checkout aisle/kiosk already set.

The system reads all the UHF tags on the items in the customers cart, process the data to the NFC reader, where the customer scans the reader with their mobile phone, and completes the checkout on the phone.

The antenna is able to read multiple tags many times a second to ensure every tag is scanned. The antenna then resumes its normal read operation when the customer has completed checkout. In all, it saves the customer time.

The system is expected to meet the following requirements:

- Design team must be able to create a system that provides customers in a retail environment with a smooth and quick checkout process by only using their phone and the items they wish to purchase.

- System must be able to use the UHF RFID tags to correctly identify the scanned items and the correct quantity of product in the bag/cart

- System should be easy to install in a retail environment and should easily integrate the inventory system of the store with the database of the SKUs/RFID tags.

- System must authenticate the customers to ensure the correct items are being charged to the correct accounts

- System should be accessed by workers and managers to handle previous purchases, returns, inventory checks, price checks, overrides, and other common retail uses.

- System must be able to complete the checkout process in a manner quicker than traditional methods.

- System must be user-friendly and easy to use in store

- System should be able to recognize all of the items in the customer's possession while leaving out other readings from their cart

Hardware

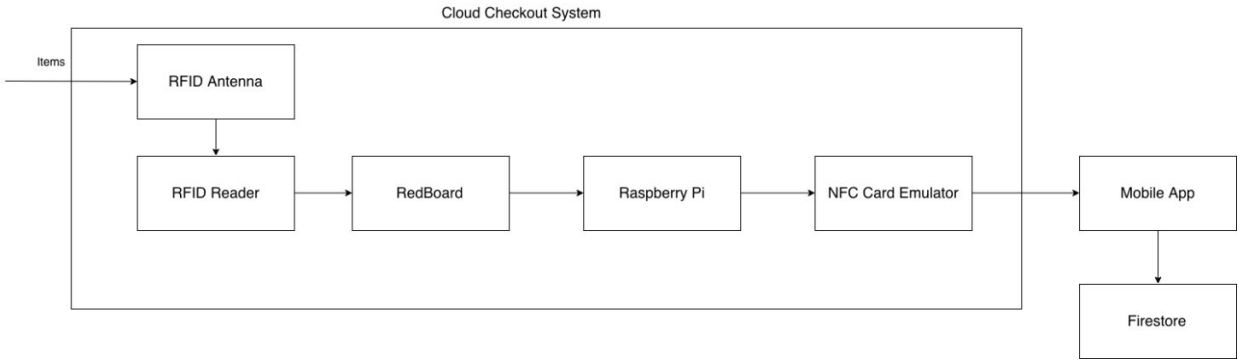

Fig.23. Overall Structure of the Design

Once assembled as outlined in the hardware designs earlier in this package, the antenna will read tags every 10 seconds. During that interval, it will attempt to read as many unique tags as possible, then save the results to the raspberry pi. The raspberry pi encodes the results into the NDEF format for NFC broadcast on the NFC Card Emulator. The customer then scans the emulator using the mobile app and proceeds to checkout.

Software

Once the customer has scanned their items, their items will appear on screen, where they are able to complete their transaction or scan again. The customer can also use the app to check the inventory of the store, view past purchases, and manage their account.

Source Code

See attached.

References

[1] Adafruit Industries, "PN532 NFC/RFID controller breakout board," *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/364. [Accessed: 07-May-2019].

[2]Theotherphp and Yennifs, “UHF RFID Antenna (TNC),” *WRL-14131 - SparkFun Electronics*. [Online]. Available: https://www.sparkfun.com/products/14131. [Accessed: 07-May-2019].

[3] “SparkFun RedBoard - Programmed with Arduino,” *DEV-13975 - SparkFun Electronics*. [Online]. Available: https://www.sparkfun.com/products/13975. [Accessed: 07-May-2019].

[4]“SparkFun Simultaneous RFID Reader - M6E Nano,” *SEN-14066 - SparkFun Electronics*. [Online]. Available: https://www.sparkfun.com/products/14066. [Accessed: 07-May-2019].