SmartKart

Group G

University of Central Florida College of Engineering and Computer Science

Senior Design I

Final Documentation

Victoria Abreu - Computer Science

Christina Heagney - Computer Engineering

Lucas Ryan - Computer Science

Doran Senior - Computer Engineering

Table of Contents

1. Executive Summary

Shopping has become a regular routine and a necessity in our everyday lives, however, it may cause some issues. Everyday shoppers spend extra time collecting coupons and standing in line. The average duration of a shopping trip lasts approximately 45 minutes; this does not include the amount of time spent by shoppers who collect coupons. Our SmartKart shopping system remedies these issues, and was designed with convenience of the shopper in mind.

With our system, shoppers will be able to cut the time spent at the grocery store, by eliminating the need for standing in line. Our system will also allow users to save more time by eliminating the coupon-cutting process. This will not only be helpful to customers who routinely cut coupons, but it will allow other shoppers who traditionally do not cut coupons to save money as well.

Our system will employ an RFID system, which will keep track of all items a shopper places in their shopping cart. From there, this information will be sent to the user's smart phone, where they will be able to easily view the total cost of the items in their shopping cart, and any coupons relevant to their items that are currently offered at that grocery store. In addition to this, users will be able to view all previous purchases made from their account, and all receipts will be sent to their email account.

The SmartKart shopping system will also employ a GPS and locking mechanism, to help the grocery stores prevent theft of the carts. The locking mechanism will be attached to the wheels of the cart, so that when the GPS system detects that the cart has traveled outside of the store's perimeter, the lock will engage and the cart will lose mobility.

This report documents the research and design done by our team to help bring the SmartKart shopping system to fruition. It will indicate our team's motivations and goals for creating this project, and it will also describe the specifications and requirements for the system.

Following this will be information regarding the research done for the design of our system, including the software and hardware components considered, and other projects that have similar functions to our design. In this section, our team made important decisions in regards to which hardware components and software applications to implement in our system.

The document will then discuss related standards and constraints that helped to shape the design of the system. We will then discuss the details of the hardware and software design, and the results of our component prototyping and testing. Finally, the document will conclude with the administrative section, which details a milestone chart to help our team reach goals, as well as a layout of our system's budget.

2. Project description

This section discusses the motivation, goals, and objectives behind the SmartKart shopping system. It also goes in to detail about the requirements and specifications for the hardware and software components implemented in the system.

2.1. Motivation

Currently when a person goes shopping they might have a hassle that can be averted. The average customer spends time searching and collecting coupons, only to forget them, rendering them useless. Regardless of the store, a customer can always expect to wait in long lines before cashing out. Also, customers with a budget must continuously check that they have haven't exceeded their allotted funds. With all these factors, a simple task like shopping can become hectic and waste time that can better be used elsewhere for the common shopper. Motivated by the Amazon Go Supermarket, which improves shoppers' experiences and gets rid of checkout lines, our project is a more affordable alternative. The proposed project not only removes the issue of long lines, but also assists users by providing coupons available at the current store and displaying the cost of the products in their shopping cart.

2.2. Broader Impact

Currently, shoppers may spend valuable time reading store ads and collecting coupons, and can expect to wait in long lines as they checkout at the store. Although self-checkout machines offer an alternative to the customary cashier-checkout process, whenever a shopper wishes to remove an item from their purchase, they must wait for an employee to assist them, thus increasing wait and checkout times. The main goal of the SmartKart system is to automate the checkout process and improve the user's shopping experience. Users will be able to easily audit previous purchases, view products and coupons/promotions offered at a particular store, and keep track of the current cost of the products in their shopping cart, all from a mobile device. The SmartKart system will assist retail stores in decreasing overhead, and collecting customer and product data, which is critical for financial decisions.

2.3. Goals & Objectives

The goal of our project is to create a lightweight, portable device that can be integrated onto common shopping carts that will improve the shopping experience. Once shoppers create an account, they can login to any available cart and begin shopping. Using RFID and GPS, users will be able to add and remove items from their cart, and once they exit the store, their debit/credit card will be charged and the user will receive an electronic receipt sent to their email. There will also be a locking mechanism that will prevent theft of these cart. The wheels on the cart will lock once it surpasses a distance from a specified radius of the store. There will be a backend system that will store the current coupons available, account information, and handle payment transactions.

2.4. Requirements & Specifications

This section discusses the requirements and specifications of the SmartKart shopping system. It discusses in detail the general specifications of the cart, such as dimensions, weight, and cost, as well as the hardware and software requirements.

2.4.1. General

- The overall shopping cart shall be no bigger than 38 by 22 by 41 in.
- The shopping cart shall weigh no greater than 60 lbs.
- The overall system shall cost no more than $500 to build.
- The user shall be able to grasp the functionality of the GUI in approximately one minute.
- The device will allow users to purchase at least 25 items.

2.4.2. Hardware

- The store items shall have a passive RFID tag attached to them.
- The microcontroller shall use RFID sensors to read the RFIDs contained on each store item's RFID tags.
- The time it takes for an RFID sensor to recognize an RFID tag shall be no greater than 50 milliseconds.
- The distance required for an RFID sensor to recognize an RFID tag shall be no less than two feet.
- The microcontroller shall keep track of the shopping cart's location via a GPS module
- There shall be one electronic locking device, which locks the back two wheels.
- The wheels shall have an electronic locking device attached to them, and shall lock based on the shopping cart's location.
- The microcontroller shall send the data from the RFIDs to the android device via bluetooth communication.
- The power supply of the overall system shall supply a voltage no greater than 12 volts.
- The power supply shall power the system for at least one hour.

2.4.3. Software

- The system will return product information from the database back to application running on the android device.
- The application will be able to display between 5 and 10 coupons at a time on the coupon section of the user interface.
- The application will be able to display user purchases in the cost section of the user interface.
- The application will be able to display a user's previous purchases.

- The application will only allow users with a verified ‘active’ credit/debit card.
- The application will allow users to select which shopping cart to pair with.
- Once the application receives the product data from the database it will update and apply the coupons under 1 second.
- Communication between the system and application must be encrypted(SSL/TLS/HTTPS) when user is cashing out.
- The system will use a credit/debit card processing application to store credit/debit card information and charge users.
- The system will send an electronic receipt to user’s specified email.
- The application must validate user input before submitting information.

2.5. House of Quality

The house of quality, shown in Table 1, is a diagram whose structure depicts the relationship between the engineering requirements and marketing requirements as applicable to our SmartKart shopping system.

		Battery Life	Time to Learn	Dimensions	Cost	Weight	Startup Time
		+	+	-	-	-	-
RF Sensor Scanning Distance	+	↑	↑	↑	↓↓		↑
Cart Usability Radius	+		↑	↑		↑	↑
Input Voltage	-	↓↓	↑	↑	↓	↓	↑
Sensor Scanner Accuracy (ms)	+	↑			↓↓	↓	↑
Cost	-	↓		↓		↓	↓
		> 1 hour	< 2 min	6 x 6 x 3 in.	< $300	< 10 lbs	< 1 min

Table 1. House of Quality Diagram

Marketing requirements are a set of needs and wants from the perspective of the potential consumer. This includes things such as cost, which consumers want as low as possible. The cost value under the marketing requirements means the cost of the product that the end user will pay.

The time to learn is also a crucial element to the system's marketing requirements; the SmartKart should be user-friendly and intuitive enough for all shoppers to understand, regardless of age and prior technology experience. As this system will be utilized by customers daily, it needs to be simple and straightforward enough so that shoppers may not only learn it quickly, but also be able to retain this information for ease of their future shopping trips.

The marketing requirements also include battery life for the system, which should be as high as possible. The average shopping trip lasts approximately 41 minutes, and thus, our specification is that the battery should be capable of powering the SmartKart system for at least an hour.

As for the remainder of the marketing requirements, the only requirement for the dimensions of the overall system is that it may easily fit underneath the shopping cart and away from user interferences. The weight of the system should also be light enough such that it does not hinder the shopping cart's mobility. It is also desired that the system have a quick start up time, so that customers may quickly begin their shopping trips without being hindered by the system's initialization.

In regards to the engineering requirements, these specifications were created with the engineers and designers in mind. The RFID system should have a large scanning distance and accurate sensors in order to pick up all items being put into the shopping cart at any position.

The GPS portion of the system should also have a radius large enough to cover the entire perimeter of the grocery store. It is desired that the cost be minimal, and the input voltage be low, to help preserve battery life.

3. Research

This section will be used to discuss the research our group performed in preparation of building our SmartKart shopping system. This includes information on similar projects, hardware components, such as micro-controllers and wireless communication, as well as software applications, such as frameworks and APIs.

3.1. Similar Projects

This section will discuss other services and applications that are already on the market, which have similar aspects to our proposed SmartKart shopping system. These include Amazon Go, InMarket, and the Publix and Retale applications.

3.1.1. Amazon Go

Announced in December 2016, Amazon Go has revolutionized the idea of supermarket shopping. Using advanced machine learning, computer vision, and sensor fusion concepts to retrieve data from many sensors, Amazon is improving shopping experiences with their checkout-free technology. The system will keep a virtual shopping cart of products the customer has placed in their physical shopping cart, will the ability to detect whenever a user adds or removes an item from their cart. This removes the requirement for checkout lanes to check what a user has in their cart. The only requirement is having the Amazon Go app and login with a working Amazon account.

3.1.2. InMarket

One company that is implementing Bluetooth technology to improve customer experience is inMarket. InMarket's Mobile to Mortor iBeacon platform integrates Apple's iBeacon Bluetooth with GPS technology on mobile devices to tracker user's location and send them notifications of coupons and items from nearby stores that support inMarket. InMarket uses iBeacons as location markers inside retailer stores to anonymously track location of customers in a store, allowing retailers to notify customers of discounts on items near their location. Note, by implementing a mesh network of beacon technology, retailers can enhance shoppers' experience by providing navigation tools on their retail apps to assist shoppers in finding products in their store.

3.1.3. Publix App

Available for Android phones, the Publix app allows users with accounts to keep track of their shopping list and find closest store and store information to them. Users can create, save, edit, and review grocery lists, using the built-in barcode scanner feature, they can scan and add item to their shopping list. Once a user selects a store, they can browse products and promotions available at the store. Users can save a virtual copy of store coupons allowing the user to redeem them at the store. The app allows users to browse products on sale by category or

department. The Publix App saves users time with searching for store coupons and notify users what isle the items on their grocery list are located in the store.

3.1.4. Retale App

Retale App is a native Android applicate that allows users to view electronic version of coupons currently available at stores. With this app, users can select their favorite stores and receive notifications of new deals related to their favorite stores. Once users create a shopping and grocery list, Retale can do a lookup to see what items on their list have discounts in nearby area. Instead of requiring users to bring printouts or cutouts of coupons to apply additional sales to purchase, Retale allows users to display a digital copy of coupons to the clerk to receive the discount. This app saves its users money by notifying them of discounts that can be applied to groceries they plan on purchasing and time spent on collecting and making physical copies of coupons.

3.2. Hardware

This section introduces the hardware components that will be employed in our SmartKart shopping system. This includes micro-controllers, wireless communication, the power supply, GPS modules, RFID sensors, locking mechanisms, and our overall PCB design.

3.2.1. Microcontrollers

For this project, a microcontroller is needed to perform essential functions. Some of the components that the microcontroller will operate are an RFID sensor, GPS module, WIFI module, and locking mechanism on the wheels. When the RFID sensor receives an RFID, the microcontroller will need to pull the RFID from the sensor and send the RFID to the application's database via the WIFI module. Additionally, the GPS module will constantly need to update the microcontroller on the location of the shopping cart. If the condition for valid locations has been broken, then the microcontroller will need to use the locking mechanism to lock the wheels.

This section will look into the technical specifications of three different microcontrollers, and see which microcontroller best fits this projects needs and requirements. These are the MSP430G2553, ATmega328P, and ATmega2560

3.2.1.1. ATmega328P (Arduino MEGA)

The ATmega328P has an 8-bit architecture, 32KB memory, three timers, a maximum clock frequency of 20MHz, and can operate between 1.8V to 5.5V. The Arduino UNO series generally uses this microcontroller. Arduino is fairly popular within the embedded programming community, due to its user-friendliness, large community, and many examples and tutorials that can be found online. This MCU can potentially be a major advantage to this project. Even though our embedded

programmers are more familiar with programming with MSP430s, the vast amount of online resources for programming the ATmega328P would aid in resolving this matter.

3.2.1.2. Atmega2560 (Arduino UNO)

The ATmega2560 has an 8-bit architecture, 256KB memory, six timers, a maximum frequency of 16MHz, and can operate between 4.5V to 5.5V. This microcontroller is generally found within the Arduino Mega. The ATmega2560 essentially has all the features as the ATmega328P and more. It is ideal for embedded applications that require a large amount of component and larger code size. The process of performing the embedded programming with the ATmega2560 would not be so difficult with the available resources online.

3.2.1.3. MSP430G2553 (Texas Instruments)

This microcontroller is the same one that is used in the laboratory for Embedded Systems. Since our embedded programmers have experience programming with this microcontroller, there would be a minimal learning curve when writing the software to make all the components communicate effectively. Some of the features it has is that it can operate between 1.8V to 3.6V, has a maximum clock frequency of 16MHz, 16KB memory, and two 16-bit timers. Additionally, it offers five low-power modes, and the microcontroller is inexpensive (costs around $10.00).

3.2.1.4. Comparison

The following table is a comparison of all micro-controller modules considered, including the MSP430G2553, ATmega328P, and ATmega2560.

Table 2: Microcontroller Comparison

Features	**MSP430G2553**	**ATmega328P**	**ATmega2560**
Operational Voltage	1.8V – 3.6V	1.8V – 5.5V	4.5V – 5.5V
Power Consumption	Active Mode: 230µA w/ 1 MHz Off Mode: 0.1µA	Active Mode: 0.2mA w/ 1 MHz Off Mode: 0.1µA	Active Mode: 500µA w/ 1 MHz Off Mode: 0.1µA
CPU Architecture	16-bit RISC	8-bit RISC	8-bit RISC
Memory	16 KB FLASH 0.5 SRAM	32 KB FLASH 1 KB EEPROM 2 KB SRAM	256 KB FLASH 4 KB EEPROM 8 KB SRAM
Analog I/O	Both	Input only	Input only
Digital I/O	Both	Both	Both
Maximum Clock Frequency	16 MHz	20 MHz	16 MHz

Features	MSP430G2553	ATmega328P	ATmega2560
Number of Clocks	2	3	6
Low Power Mode	Yes	Yes	Yes
I/O Pins	24	23	86
Temperature Range	-40°C to 85°C	-40°C to 85°C	-40°C to 85°C
Price	$9.99	$24.95	$45.95

3.2.1.5. Choice

After careful analyzation of Table 2, it was decided that the ATmega328P MCU would best serve this project for several reasons. The ATmega328P provides a clock frequency of 20MHz, which means overall better processing speed and performance. The ATmega328P also has access to three clocks, while the MSP430G2553 only has two. Having only two clocks available might hinder the implementation process when writing the program that will monitor the GPS module, locking mechanism, RF sensor, and WIFI module. Three clocks are ideal, since one clock can be used to handle interrupts for the RF sensor and WIFI module, one clock can be used to handle the interrupts for the GPS Module and locking mechanism, and the other clock can be used to monitor the overall system. Additionally, analog output does not need to be provided to any of the components for this project, so this feature that the MSP430G2553 has would be left unused. The operational voltage for the ATmega328P has a maximum voltage of 5.5V, which would be more ideal for our battery. The battery would supply a voltage of either 9V or 12V. Using the battery in combination with a buck converter and the ATmega328P would produce a minimal dropout voltage and less power dissipation. While the ATmega2560 has all features of the ATmega328P and more, many of its features would most likely not be needed. For example, all 6 clocks and 86 general-purpose I/O pins would not be used, and would lead to wastefulness. The ATmega328P has everything that is desired and does not leave behind too many features that will go unused.

3.2.2. Communication

The SmartKart system requires a method of communication between the microcontroller and the user's mobile device to allow for easy and practical use. The following sections will discuss the different types of wireless communication technology components that our team considered, the advantages and disadvantages of each component, and a thorough analysis of how teach device operates. Analyzing each component from a low level perspective allows our team to fully optimize the design of our SmartKart system.

3.2.2.1. Bluetooth

Bluetooth is a universal wireless standard used for connecting digital devices. It

eliminates the need for wires or cables by alternatively using radio waves, ranging from 2.40GHz-2.485GHz, to allow the devices to communicate. Each Bluetooth device is embedded with a transceiver microchip, which allows it to connect and communicate, or “pair”, with other in-range devices that also possess Bluetooth capability.

Since Bluetooth uses radio frequencies to communicate, it does not need line-of-sight to communicate. Any device may connect with several other devices at one time, so long as they are all Bluetooth enabled, and within transmission range. Bluetooth data transmission will not interfere with other types of data transmissions, such as RF devices and WLAN.

Bluetooth is commonly used in computers, smart phones, music players, and a multitude of other smart devices. Bluetooth technology allows for easy installation and connection, and it is also inexpensive. Disadvantages include security risks, as Bluetooth devices may be easily hacked, and short-range communication, as the transmission range is approximately 30 feet

a. Adafruit Bluefruit LE SPI Friend

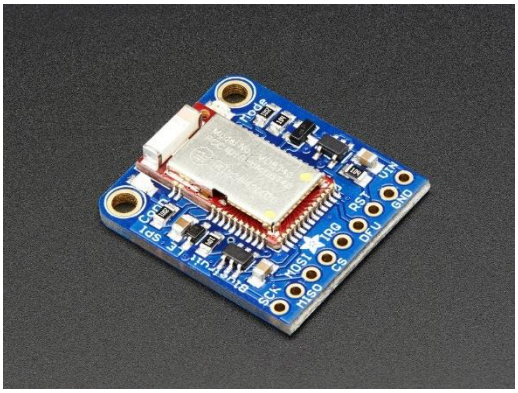


Figure 1. Bluefruit LE SPI Friend

The Adafruit Bluefruit LE SPI Friend is a module that allows for seamless incorporation of Bluetooth Low Energy into electronic projects. Bluetooth Low Energy is intended to require significantly less power consumption than Classic Bluetooth, while still maintaining a comparable communication range. This is achieved as the Bluetooth Low Energy protocol only sends data as needed, instead of maintaining a constant connection running, making it an excellent option for devices that are fueled by power sources, and need to be power consumption-friendly.

This module allows you to easily add Bluetooth Low Energy wireless connection to any devices supporting SPI. The following includes the technical specifications for the Adafruit Bluefruit LE SPI Friend module, as provided by the official Adafruit website, www.adafruit.com:

- ARM Cortex M0 core running at 16MHz
- 256KB flash memory
- 32KB SRAM
- Transport: SPI at up to 4MHz clock speed
- 5V-safe inputs (Arduino Uno friendly, etc.)
- On-board 3.3V voltage regulation
- Bootloader with support for safe OTA firmware updates
- Easy AT command set tunneled over SPI protocol to get up and running quickly
- 23mm x 26mm x 5mm / 0.9" x 1" x 0.2"
- Weight: 3g

As sold by the official Adafruit website, this module may be purchased for $17.50.

b. HC-05 Bluetooth Module

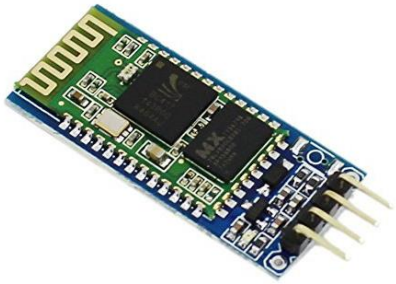


Figure 2. HC-05 Bluetooth Module

The HC-05 is a Bluetooth module that behaves as either a master or slave device. This module can be used to build a wireless connection to other devices. For our system, the user's phone would connect to it; after the RFID scans a tag, its value would be sent to the phone, and the phone would send it to the server.

This module contains a variety of pins. The KEY pin is used to trigger the AT command setup mode. The VCC pin is used for powering the device. The GND pin acts as a ground. TXD is the serial output pin, used to transmit data between the module and the microcontroller. RXD is the serial input pin, used to receive data from the microcontroller's serial transmit. The STATE pin is used to verify if the device is actively connected.

The following includes the technical specifications for the HC-05 Bluetooth module**:**

- Operating Voltage: 3.3V
- Baud rate(s): 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Dimensions: 28mm x 15 mm x 2.35mm
- Operating Current: 40 mA

3.2.2.2. Wi-Fi

Wi-Fi is a type of electromagnetic radiation that is used to provide internet connectivity to electronic devices. This technology allows for wireless connectivity for many devices, such as computers, mobile phones, and tablets[c1]. Wi-Fi's wavelength is approximately 12cm, which gives it a range of approximately 50-100 meters.

WiFi utilizes radio waves to transmit data and information across the network. A wireless transmitter receives data from the network, then converts this data into radio waves and emits them. This creates a Wireless Local Area Network, or WLAN, allowing all devices within range to wirelessly connect to the network. On the other hand, if the user attempts to send data through the internet, this procedure acts in reverse; the wireless transmitter will receive the data as a radio signal from the user's device, and will convert this signal to be communicated back.

Some advantages of Wi-Fi include convenience, mobility, and expandability. Wireless networks provide users with the capability of accessing the internet from nearly any location that is convenient to them within range of their network. This is particularly useful for mobile devices, such as laptops, smartphones, and tablets. Mobility also plays a large factor into the benefits of Wi-Fi. This allows users to access the network outside of their typical home environment, in places such as shopping stores and coffee shops. So long as there is a public network connection, the user will still have full connection capabilities.

Expandability is another advantage of Wi-Fi communication. A wireless network may easily accommodate an increased number of users, so long as all clients are within the network range, whereas wired networks would entail additional wiring in order to expand the internet access capabilities to support a larger amount of users.

The disadvantage of Wi-Fi is that it may be easily interfered with. There are many physical barriers, such as wood, concrete, and metal, that prevent a WiFi signal from reaching its full range. There may also be frequency interference, stemming from microwaves, other wireless devices, or even other WiFi Networks, that may cause interference with a WiFi signal. WiFi may not always be reliable, as it is subject to the aforementioned every day interferences.

Other disadvantages include security, as public networks are inherently unsecure and may be easily compromised by hackers, and range, as the typical network may only span about 10 meters, and expanding this range will require additional network access points to be purchased, which may quickly increase costs.

a. ESP8266 Wifi Transceiver Module Esp-01

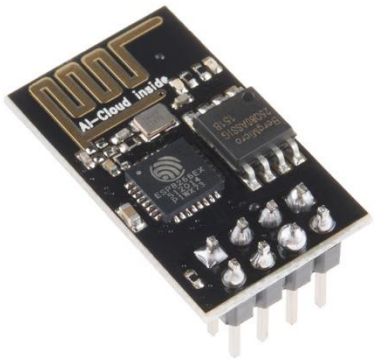


Figure 3. ESP8266 Wifi Transceiver

The ESP8266 Transceiver module is a system-on-a-chip, or SOC, that integrates the TCP/IP protocol stack; it allows for Wi-Fi network access to any microcontroller. The ESP8266 module is pre-programmed so that the user may simply connect the module to a microcontroller and obtain a comparable amount of Wi-Fi capability as a Wi-Fi shield would offer.

This module has powerful on-board processing and storage, which allows for easy integration of sensors and “other application specific devices” through its general-purpose input/output pins with minimal up-front development and runtime loading.

Serial-to-WiFi modules utilize serial Transceivers/Receivers to send and obtain Ethernet buffers, and may use serial commands to modify the settings of the module. In order to communicate amid the Wi-Fi [c1] and a microcontroller, the module requires two wires, the Transceiver and Receiver.

An advantage of the ESP8266 module is that it is specifically designed to be an energy-efficient, low-power consumption, module. This module operates in three separate modes: active mode, sleep mode, and deep sleep mode. These different options are beneficial, as you can modify the module to fit the needs of your project. For the SmartKart, it would be most beneficial to use the sleep mode, or deep sleep mode, as our system will be running off of a battery, and thus, will need to use an option that consumes less power.

One disadvantage is that this module is not breadboard-friendly, meaning that the user cannot simply plug it in to a breadboard for quick prototyping. This issue requires the user to either employ female-to-male jumper wires or a breadboard adapter in order to properly set up the module for testing.

The following includes the technical specifications for the ESP8266 Serial Wireless

Wifi Transceiver module, as provided by the official SparkFun Electronics website, www.sparkfun.com:

- 802.11 b / g / n
- Wi-Fi Direct (P2P), soft-AP
- Built-in TCP / IP protocol stack
- Built-in TR switch, balun, LNA, power amplifier and matching network
- Built-in PLL, voltage regulator and power management components
- 802.11b mode + 19.5dBm output power
- Built-in temperature sensor
- Support antenna diversity
- off leakage current is less than 10uA
- Built-in low-power 32-bit CPU: can double as an application processor
- SDIO 2.0, SPI, UART
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU, A-MSDU aggregation and the 0.4 Within wake
- 2ms, connect and transfer data packets
- standby power consumption of less than 1.0mW (DTIM3)

As sold by Amazon.com, this module may be purchased for $6.99.

b. Adafruit ATWINC1500 Wi-Fi Breakout

Figure 4. ATWINC1500

The Adafruit ATWINC1500 Wi-Fi Breakout module is used to connect your Arduino to the internet. This module includes Secure Sockets Layer (SSL) support, which is a standard in security which establishes an encrypted link between a client and its server. It also utilizes the SPI protocol to communicate, which means it may quickly communicate with several devices over short distances.

One disadvantage to this module is that it is less cost efficient than other wireless communication alternatives. As sold by the official Adafruit website, this module may be purchased for $24.95, making it the priciest module to be considered for our wireless communication requirements.

The following includes the technical specifications for the Adafruit ATWINC1500 WiFi Breakout module:

- 33mm x 28mm x 4mm
- Weight: 4.4g

3.2.2.3. Comparison

After extensive research, our team decided that the Adafruit Bluefruit LE SPI Friend and the HC-05 Bluetooth modules did not meet the needs and requirements of our project, and thus will not be used.

As observed, one of the benefits of the ESP8266 Transceiver Module is that it is very cost efficient, and may be purchased on Amazon for only $6.99. In contrast, the Adafruit ATWINC1500 WiFi Breakout module is sold at $24.95 per device.

Another benefit of the Esp8266 module is that it has many options for energy-efficiency modes, whereas the ATWINC1500 Breakout module does not offer this. This is vital for our SmartKart shopping system, as the system will be powered by a battery, and thus, our device needs to be as energy-efficient as possible.

Our team will ultimately use the ESP8266 Serial Wireless Wifi Transceiver Module as our choice of wireless communication for our SmartKart system.

3.2.3. Components

This section will discuss the different components needed for our system's design, including the locking mechanism, GPS system, and sensors. It will detail an overview of each of the sections, and specify which components were considered for each.

3.2.3.1. Locking Mechanism

Our SmartKart system will also employ a locking mechanism to help prevent theft. The locking mechanism will be set up in correlation to our GPS module. When the GPS module detects that the SmartKart shopping cart has gone outside of a predetermined range, the wheels on the cart will lock.

There are two distinct concepts regarding locking devices. The first is a "fail-safe" lock. Fail-safe locks will lose connection and unlock when the power fails. These types of locks are typically used in situations where safety is at risk; it is ideal for locks such as on fire exit doors. The second, "fail-secure" locks, remain locked when the power is cut. These locks are more ideal when security is of utmost importance, in places like banks or government facilities.

One of the locking mechanism options to be considered by our team is the Electromagnetic lock. The other options are locking systems that are already employed by stores to ensure protection of their shopping carts; these include the GS2 and CAPS systems.

a. Electromagnetic

The first option to consider is an electromagnetic lock. Electromagnetic locks consist of both an electromagnet and an armature plate. When the electromagnet is energized, it becomes strongly attracted to the armature plate. This is due to the current that passes through the electromagnet when power is applied; this current creates a magnetic flux, causing the two components to attract each other. When these two components are in contact with each other, they create a securely locked connection.

The majority of electromagnetic locks are inherently fail-safe; when disconnected from their power source, they will automatically unlock. However, some may be constructed to be fail-secure devices. These locks use magnets to clasp on to a spring-loaded locking mechanism, so that the door may only unlock when the power is on.

There are many advantages to electromagnetic locks. One is that the lock may be easily turned on or off by simply adjusting the status of the power source. They are also easy to install, in comparison to other types of locks, as they consist of two main parts (the electromagnet and armature plate), and require no interconnecting components. They are also quick to release when the power source is cut, which is vital in emergency situations. Electromagnetic locks are also quite sturdy; if the lock is forced open, the physical components will generally suffer from little to no damage, as the lock does not consist of any movable, breakable components.

There are also a few disadvantages to using electromagnetic locks. One downfall of these locks is that they require the power source to be constantly active in order for the lock to remain secure. This means that they also pose security risks, as the locks may be compromised by simply cutting the power source. These locks are also more expensive than traditional mechanical locks, and in order to implement a “fail-secure” system, additional hardware will be required, such as a magnetic bond sensor, which will further increase cost.

Electromagnetic locks are built with metal plates, which are encompassed by a coil of wire in order to establish a secure attraction between the electromagnet and the armature plate. Depending on the number of coils, the lock will have different holding forces. The following table demonstrates the range of holding forces that electromagnetic locks may employ:

Table 3. Electromagnetic lock holding forces

Lock Size	Holding Force
Micro Size	275 lbf
Mini Size	650 lbf
Midi Size	800 lbf
Standard Size	1,200 lbf
Shear Lock	2,000 lbf

b. GS2 System

The GS2 wheel is one of the most popular shopping cart theft-prevention mechanisms used today. The system is comprised of an RF receiver antenna and a Perimeter Antenna. The RF receiver antenna within the GS2 wheel receives a signal from the Perimeter Antenna; when the shopping cart enters the range of which the Perimeter Antenna, and crosses the transmitter boundary, a locking signal is sent out to the GS2 wheel, and tells it to lock the shopping cart. The amount of time it takes for the signal to be transmitted from the Perimeter Antenna, and received by the GS2 wheel, is approximately 2.2 second.

There are two methods to unlocking the shopping cart. The first is that a signal must be sent from the CartKey remote control, which will be controlled by a store employee. The signal from this remote control unit is approximately 4 feet. The second involves the Perimeter Antenna; this must be set to Master Unlock mode, and from there it may send a signal to the GS2 wheel to unlock the cart.

In the event of a system failure, the GS2 wheel takes a “fail-secure” approach. The only chance of a system failure would occur when the GS2 wheel is already in the locked position, and as such, would remain locked. This could be due to a physical component of the wheel failing while in use.

As for the power supply, the GS2 system runs off of a CR123A, 3-Volt lithium battery. This battery is durable and assumes a long life expectancy, with a shelf life of approximately ten years when uninstalled, and five years when installed, but inactive. The GS2 system is expected to last approximately three to four years, under the assumption that the wheel will go through a daily process of being in the locked and unlocked state, four times each day. This is beneficial for shopping stores, as it cuts back on labor and expenses in regards to replacing the power system for each shopping cart.

c. CAPS

Carttronics' CAPS is an alternative cart retention system to GS2. This system is comprised of a gauge antenna wire, a signal transmitter, and a Carttronics' locking CAPS caster. The antenna wire is used to draw boundaries around the store's perimeter to set a range for how far the shopping carts are allowed to migrate. The transmitter is located inside the store; when a shopping cart passes over the perimeter, the transmitter sends a signal to the CAPS caster, which deploys a braking shell on the wheel, preventing any further movement. The system's braking shell may be removed by the stores employee with the use of a reset controller, which will allow the cart to return to normal operation.

The following lists key features regarding Carttronics' CAPS shopping cart locking system, as provided by the official Carttronics website, www.carttronics.com:

- Highly effective, durable and visible braking shell that serves to stop the cart and protect the wheel from flat spots
- Visible braking shell provide "notice" to customers and store personnel to minimize the risk of injury and liability and expedite recovery and return to service
- Modular, maintainable hardware enables repair and reuse and the industry's lowest lifetime cost of ownership
- With internet connectivity, provides around-the-clock remote monitoring
- Availability of parts and components to facilitate self-maintenance or 3rd party managed preventative maintenance programs
- Includes web-based maintenance and operations training programs and supporting management reports
- Option to include automated battery voltage monitoring and other software services

3.2.3.2. GPS Overview

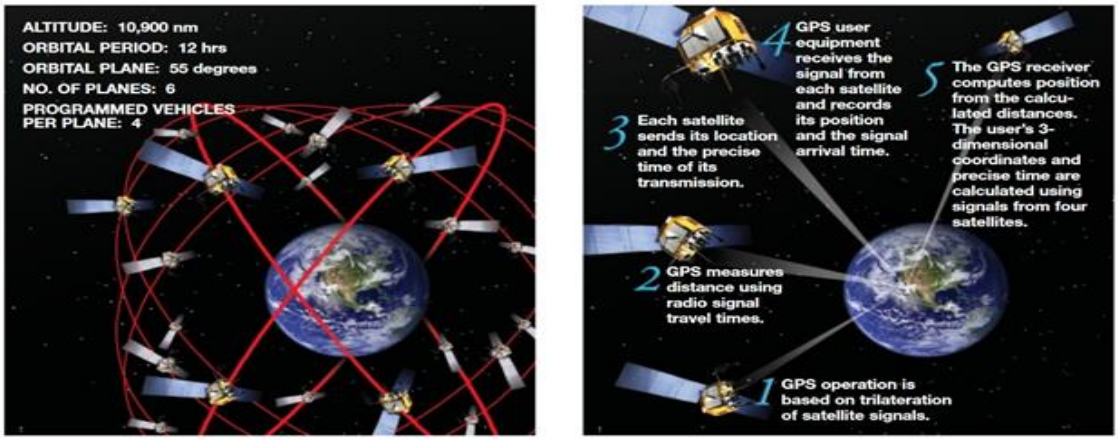


Figure 5. GPS Constellation and Trilateration (Northrop Grumman)

The Global Positioning System (GPS) was created by the United States Department of Defense for military use, and was later changed for civilian use. GPS consists of a constellation of 27 Earth-orbiting satellites. Each satellite weighs about 3,000 to 4,000 pounds, and can be positioned as much as 12,000 miles (19,300 km) from the earth's surface. A GPS receiver is used in many devices to calculate the exact position the device is on the earth's surface, relative to the satellites. To accomplish this, a GPS receiver needs to locate at least 4 or more satellites and find its distance between each satellite. This information can then be used to calculate positioning information, such as latitude, longitude, and altitude. This process is known as three-dimensional trilateration.

a. Trilateration

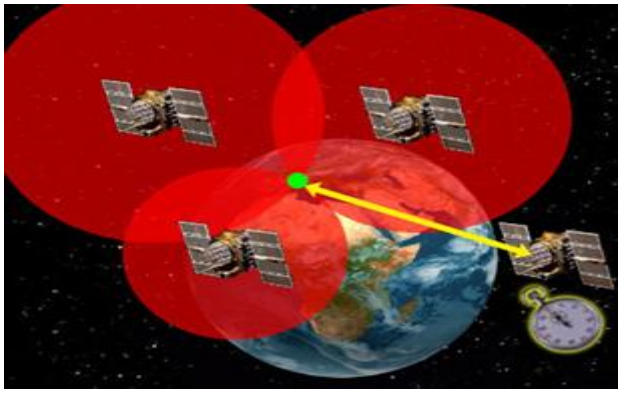


Figure 6. GPS Receiver Location (GPS Reflections Research Group)

The GPS receiver distinguishes the satellites through a unique binary code associated with each one. The receiver also has these codes stored in its memory, so that it can identify which satellite it is receiving a radio signal from. When a satellite sends out its code to a GPS receiver via radio signal, the receiver compares the received code with the satellite codes in its memory, to identify the satellite. The code is shifted by a given number of bits to represent the travel time. For example, if the code is shifted by 25 bits, then we could say it took 25 milliseconds for the satellite's signal to reach the receiver.

The travel time can then be multiplied by the speed of light to calculate the distance the satellite is from the receiver. This distance is used as the radius of a sphere around a satellite, with each point on the sphere indicating a possible given point the receiver is located. After 4 or more satellites have sent their codes to the receiver and their spheres have been created, the receiver's position can be found from the point at which all the spheres intersect adjacent to the earth's surface. From this point, the receiver's longitude, latitude, and altitude can be found.

To make an accurate positioning, the clocks operating on the satellite and receiver need to be synchronized. Atomic clocks are used in satellites to provide exact time tracking, but are not used for receivers, due to atomic clocks being extremely expensive. To solve this, the clock on the GPS receiver is constantly reset and then set to the same time value the satellites have.

b. Error Correction

Errors can occur when calculating the distance a satellite is from a GPS receiver. The earth's electromagnetic field can slow down the travel speed of a satellite's radio signal, and radio signals can also bounce off sizable objects. When situations like these occur, the distance calculated can sometimes be misrepresented. To solve this, differential GPS (DGPS) stations are used. DGPS stations are fixed at one location, so their position is always known. With this new information, any distance inaccuracy is found and corrected. The error correction can be broadcasted to DGPS-compatible receivers in the surrounding area, and these receivers will make the necessary adjustments.

c. GPS Module

For this project, a GPS module will be needed to track the shopping cart's position relative to the grocery store. The software within the microcontroller will use the GPS module to constantly know its location. The location of the grocery is fixed, and its coordinates will be known. If the GPS module's latitude/longitude coordinates have an 80 meter difference than the store's latitude/longitude coordinates, the microcontroller will lock the wheels of the shopping cart. This procedure would a customer prevent stealing a shopping cart. In the inevitable chance a customer places the shopping cart in their car and drives off, the GPS

module will still be tracking the cart's location, which could be used to locate and apprehend the thief.

There are two GPS modules that are sought after for this project. They are the Adafruit Ultimate GPS Breakout and the U-Blox NEO-6M GPS Module. Listed below are some of the technical specifications associated with the modules.

Table 4: GPS Technical Specifications

Feature	Adafruit Ultimate GPS Breakout	U-Blox NEO-6M GPS Module
Satellites	22 tracking w/ 66 channels	50 channels
Patch Antenna Size	15 mm x 15 mm x 4 mm	N/A
Update Rate	1-10 Hz	5 Hz (max)
Position Accuracy	3 m	N/A
Velocity Accuracy	0.1 m/s	0.1m/s
Warm/Cold Start	34 s	26s
Acquisition Sensitivity	-145 dBm	-160 dBm
Tracking sensitivity	-165 dBm	-161 dBm
Maximum Velocity	515 m/s	500 m/s
Input Voltage	3.0-5.5 V DC	2.7-3.6 V DC
Load Current	25 mA tracking 20 mA navigation	67 mA

After some time analyzing and determining which GPS module would be most effective for this project, it was decided. The GPS module that will be used for this project is the Adafruit Ultimate GPS Breakout. The Adafruit Ultimate GPS Breakout can operate at a maximum input voltage of 5.5V, while the NEO-6M can only operate at 3.6V max. This allows more flexibility for the voltage regulator when supplying an input voltage, and would ultimately dissipate less power. Additionally, the module is NMEA 0183 compliant, has a 9600 baud rate, and is DGPS/WAAS/EGNOS supported.

3.2.3.3. Sensors

The SmartKart shopping system will employ the use of sensors in order to keep track of the items in the user's cart. All items will have an RFID tag embedded in to them. The RFID sensors will line the perimeter of the shopping cart, so that when the user adds a new item into their cart, our system will be able to detect what the item is, its cost, and eventually scan the store's database to identify any potentially available coupons associated with that product. The system will also require the use of ultrasonic sensors in order to detect when an item has been removed from a user's cart.

This section will provide an in depth discussion on how RFID systems work, and compare and contrast two available options that may be implemented in to our SmartKart system.

a. RFID

RFID systems are used to keep track of specific items. A simple RFID system is comprised of four components:

- Tag chips/integrated circuits
- Tag antennas
- Reader control/application software
- Reader antenna

In an RFID system, tags are embedded into all items that need to be tracked. The tags are created from tag chips/integrated circuits that are connected to a tag antenna. The integrated circuit contains all necessary information about an item, including its electronic product code, in order to be traced and identified by the RFID readers.

The electronic product code, which exists inside the memory of the integrated circuit, is written and stored in the tag chip's memory via an RFID printer. The electronic product code used is embedded into the integrated circuit as a 96-bit data string. The initial eight bits of the string are reserved for the header; this header is used to identify the protocol version. The following twenty-eight bits of

the string are used to identify the business that administers the tag's data. The following twenty-four bits in the string are designated as an object class; this is used to identify what type of product it is. The final thirty-six bits of the string are encoded as an exclusive serial number, used for a specific tag.

The RFID readers are devices connected to the network that utilize antennas to provide a link between the system's software and the tags in order to send necessary data about the items to the system. The RFID reader is capable of wirelessly communicating with any tags within the antenna's range, in order to gather data such as inventory, filter through products, or write to specific tags

PN532 NFC/RFID controller breakout board-v1.6

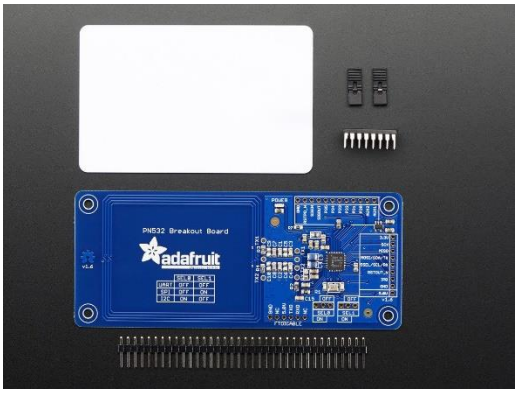


Figure 7. PN532 NFC/RFID breakout board

The PN532 is a Near Field Communication (NFC) chip, and may be used for a multitude of tasks, such as read and write to RFID tags, communicate with mobile devices, and behave like a NFC tag.

Near Field Communication is a set of communication protocols that behaves as short range Bluetooth would, without any need for authentication. Near Field Communication allows two devices to establish a wireless communication, so long as they are within a 4cm distance of one another. NFC can be used for a multitude of tasks, such as mobile payment systems, keycards, and even social networking.

As this chip can read and write to RFID tags, it may also be utilized for tasks and projects that require an RFID system.

The technical details of this module, as specified by www.Adafruit.com are as follows:

- 2" x 4.7"
- 0.425" thickness
- Utilizes I2C 7-bit address 0x48

As sold by the official Adafruit website, this module may be purchased for $39.95

RFID Reader ID-3LA (125 kHz)

Figure 8. ID-3LA Texas Instruments RFID Reader

The ID-3LA is a small reader module that may be used in RFID systems to communicate with RFID tags. This module utilizes an external antenna, providing a receiving range of up to 30cm, in order to communicate with and read tags. Data formats that are supported by the reader include Wiegand26, Magnetic ABA Track2, and ASCII.

Using the module is straightforward; simply power the module, expose it to a 125kHz card, and the output will be a serial string which contains the unique ID of the given RFID tag.

The features of this module, as provided by www.sparkfun.com are as follows**:**

- 5V supply
- 125kHz read frequency
- EM4001 64-bit RFID tag compatible
- 9600bps TTL and RS232 output
- Magnetic stripe emulation output
- 100mm read range

As sold by the official Sparkfun website, this module may be purchased for $25.95.

b. Ultrasonic Sensor

Ultrasonic sensors work by transmitting high-frequency sound pulses, and calculating how much time it takes for the echo of these pulses to reflect back. That time is then used to calculate the distance of an object.

Ultrasonic sensors are capable of measuring the following types of data:

- Distance
- Position
- Level
- Diameter

Ultrasonic sensors may be used for a variety of tasks, such as security systems, robotic navigation, and winder roll applications.

These sensors typically have two openings; one is used for transmitting the high-frequency sound pulses, whereas the other opening is used for receiving them. The following mathematical equation is used for calculating distance:

$$Distance = \frac{Time\ x\ \text{Speed of Sound}}{2}$$

Where the speed of sound is approximately $340.29\ \frac{m}{s}$. This is used, alongside the total time it takes for the sound pulses to be transmitted and received by the sensor, to determine the sensor's distance from an object.

Ultrasonic Sensor HC-SR04

The HC-SR04 ultrasonic sensor can detect within a distance of 2 cm-400 cm. The module also has a ranging accuracy of up to 0.3 cm. Each HC-SR04 comes complete with an ultrasonic transmitter, a receiver module, and a control circuit.

The HC-SR04 module contains four pins: VCC, Trig, Echo, and GND. This pins are used for power, input, output, and ground, respectively.

Technical features, as provided by www.sparkfun.com, include:

- 5V DC operating voltage
- 15mA operating current
- 15° measure angle
- 2cm - 4m distance range

As sold by the official Sparkfun website, this module may be purchased for $3.95.

PING))) Ultrasonic Distance Sensor

The PING))) Ultrasonic Distance Sensor is another option to consider for distance measurement. This ultrasonic distance sensor module contains three pins: VCC, SIG, and GND. These pins are used for power, input/output, and ground, respectively.

Key features of this module, as provided by the official Parallax website, www.parallax.com, include:

- 2cm-3m distance range
- Ultrasonic measurements are unaffected by lighting conditions
  - May be used as a supplement for infrared object detectors
- Input and output requires a single I/O pin
- Burst indicator LED indicates sensor activity
- No soldering required to connect to a development board

Details of this module, as described by www.parallax.com, include:

- Narrow acceptance angle
- Range: approximately 1 inch to 10 feet (2 cm to 3 m)
- 3-pin male header with 0.1" spacing
- Power requirements: +5 VDC; 35 mA active
- Communication: positive TTL pulse
- Dimensions: 0.81 x 1.8 x 0.6 in (22 x 46 x 16 mm)
- Operating temperature range: +32 to +158 °F (0 to +70 °C)

This module can be purchased at www.parallax.com for $29.99.

c. Choice

Upon further research, our team has decided that ultrasonic sensors will not be necessary for the construction of our project. Our team has chosen to implement the PN532 NFC/RFID controller breakout board in our system, as it can be used to easily keep track of what items are being put in to the user's cart, as well as any items that are removed by the shopper.

3.2.4. Voltage Regulators

This section will be used to discuss the research our team did while considering power supply solutions to employ in our SmartKart shopping system. This includes different types voltage regulators, including Linear Voltage Regulators and Switching Voltage Regulators.

3.2.4.1. Linear Voltage Regulator

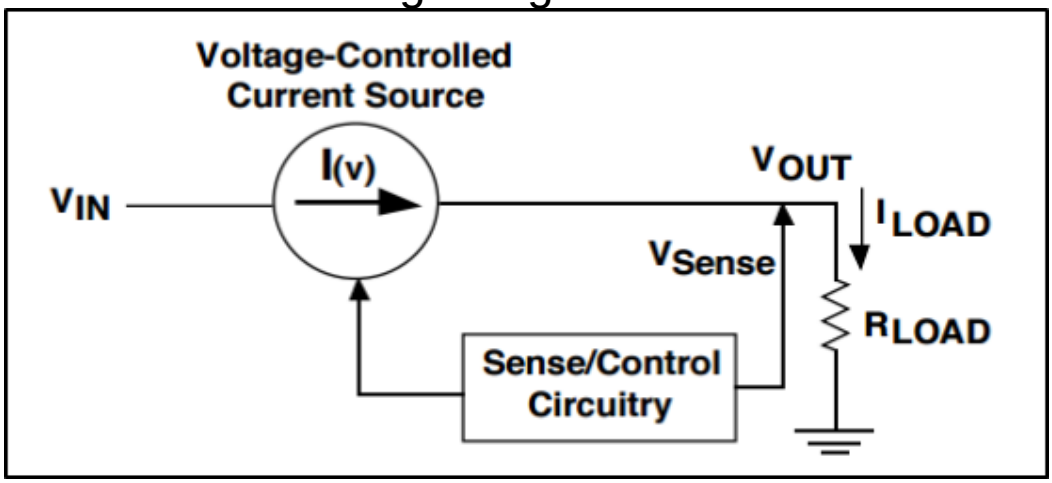


Figure 9. Linear Voltage Regulator General Configuration (Texas Instruments)

Linear voltage regulators generally function by using a voltage-controlled current source to push a constant voltage at the output terminal. The output voltage must be monitored to ensure it is constant. To achieve this, control circuitry is used to adjust the current source. The design constraints on the current source can limit the maximum current the regulator can output to the load. A feedback loop is used to control the output voltage but requires compensation. An important trait to consider is the transient response time of a linear voltage regulator after a change in the current load has been detected.

3.2.4.2. Feedback Control Loop

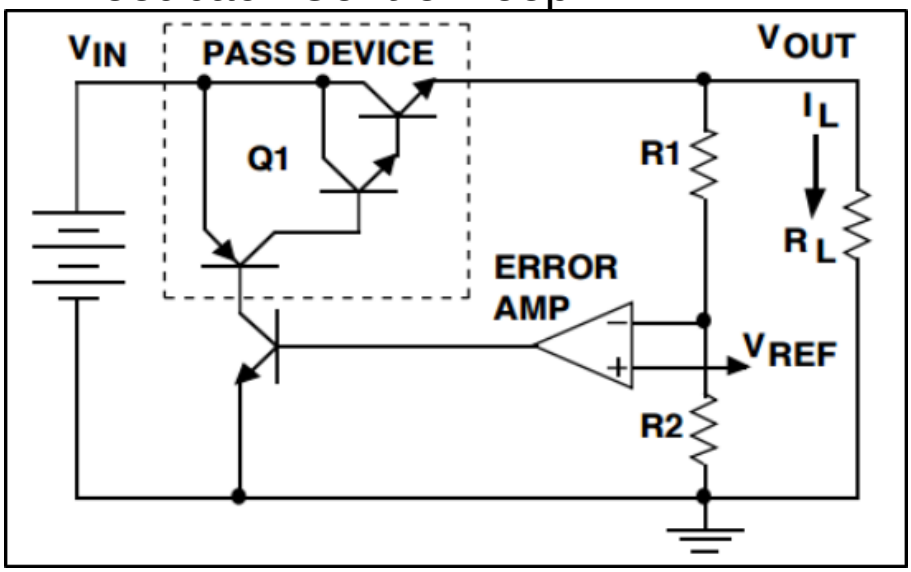


Figure 10. Linear Voltage Regulator Control Loop (Texas Instruments)

Linear voltage regulators typically use a passing device to provide a load current. In Figure 10, the passing device consists of a NPN Darlington pair that is driven by a PNP transistor. The current coming from the passing transistor's emitter junction provides the current load and flows into the negative end of the error amplifier. R1 and R2 resistors make up a voltage divider to monitor the output voltage. The positive end of the error amplifier is connected to a reference voltage. The purpose of this is for the error amplifier to constantly adjust the voltage it is outputting to the pass device. This will force the voltage on the error amplifier's negative end to be equal to the reference voltage.

The control loop provides a constant output voltage, even when the load current varies. When the load current does vary, the output voltage varies as well. When this occurs, the control loop begins to adjust to the transient response. The voltage that appears through R1 and R2 is no longer the same as the reference voltage. This voltage difference causes the error amplifier to correct the voltage it outputs, as well as the current that flows into the passing device.

3.2.4.3. Linear Voltage Regulator Types

We will look three types of linear voltage regulator designs. These include the Standard Regulator, Low Dropout (LDO) Regulator, and the Quasi LDO Regulator.

Among all three of these regulators, an important aspect to consider is the dropout voltage, which is defined as the minimum voltage drop that can be produced across the regulator while still being able to maintain output voltage regulation. Linear voltage regulators with small dropout voltages will dissipate small quantities of internal power and produce high efficiency. The Standard Regulator typically produces high dropout voltage, while the LDO Regulator produces low dropout voltage.

a. Standard Regulator

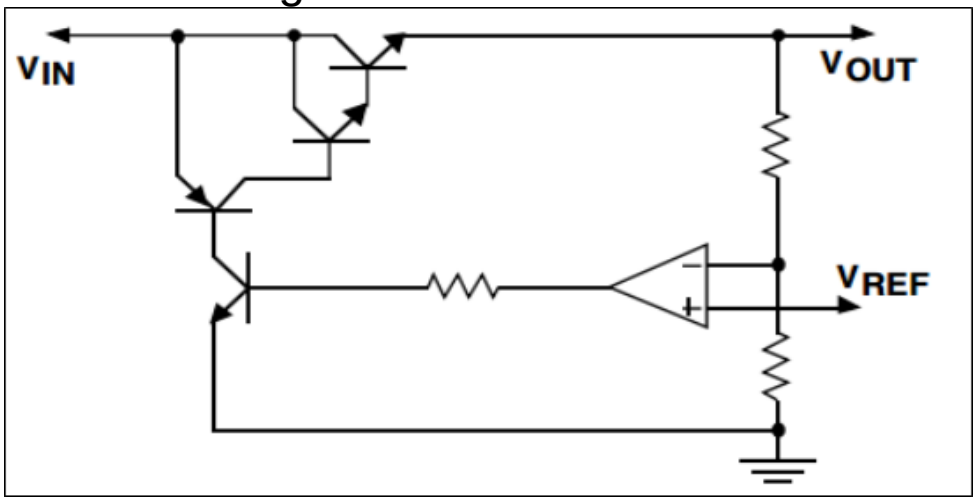


Figure 11. Standard Linear Voltage Regulator (Texas Instruments)

The standard linear voltage regulator in Figure 11 uses a NPN Darlington transistor pair to provide a constant output voltage. To provide consistent output voltage regulation, the pass transistor needs to have a minimum voltage across it, which is given by this equation:

$$V_{D(MIN)} = 2\,V_{BE} + V_{CE} \qquad \text{(Standard Regulator)}$$

This voltage is typically set around 2.5V to 3V by the manufacturer to clearly detail performance limitations. The dropout voltage is typically around 1.5V to 2.2V for a Standard Regulator.

b. Low Dropout (LDO) Regulator

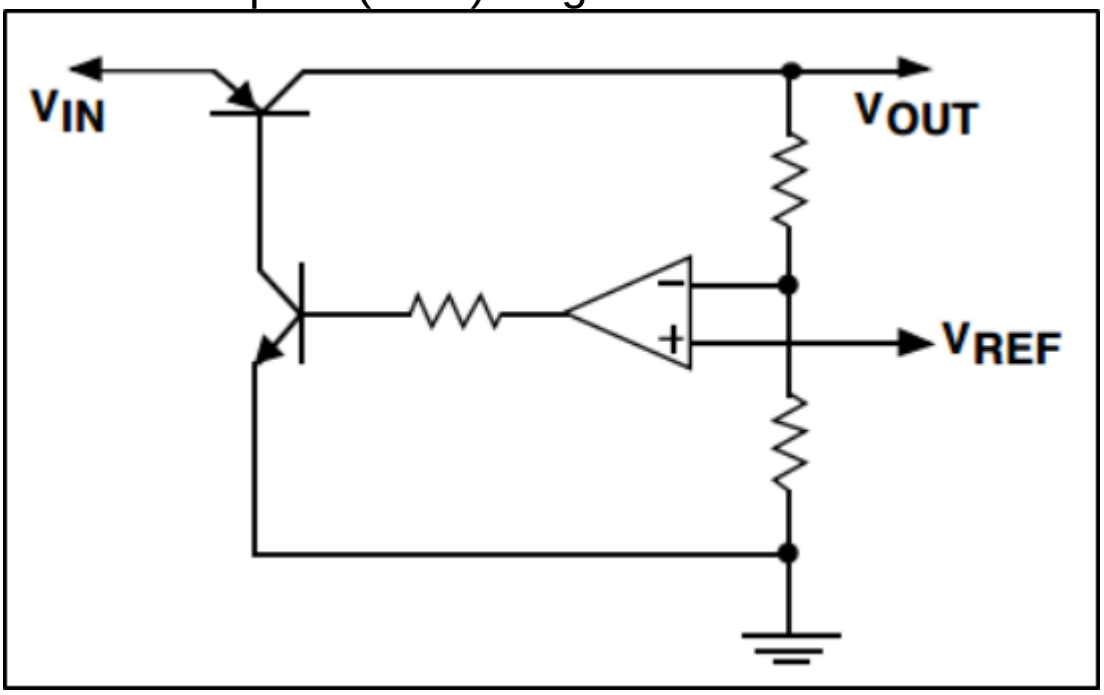


Figure 12. LDO Linear Voltage Regulator (Texas Instruments)

The pass transistor of the LDO Regulator is a PNP transistor, as seen in Figure 12. To maintain consistent output voltage regulation, the LDO Regulator needs to have a minimum voltage drop, which is given by this equation:

$$V_{D(MIN)} = V_{CE} \qquad \text{(LDO Regulator)}$$

The dropout voltage of the LDO Regulators is considerably lower than the other linear voltage regulators. This provides low power dissipation and high efficiency. The maximum dropout voltage is typically around 0.7V to 0.8V at full current. There is a direct relation between the dropout voltage and the load current.

c. Quasi LDO Regulator

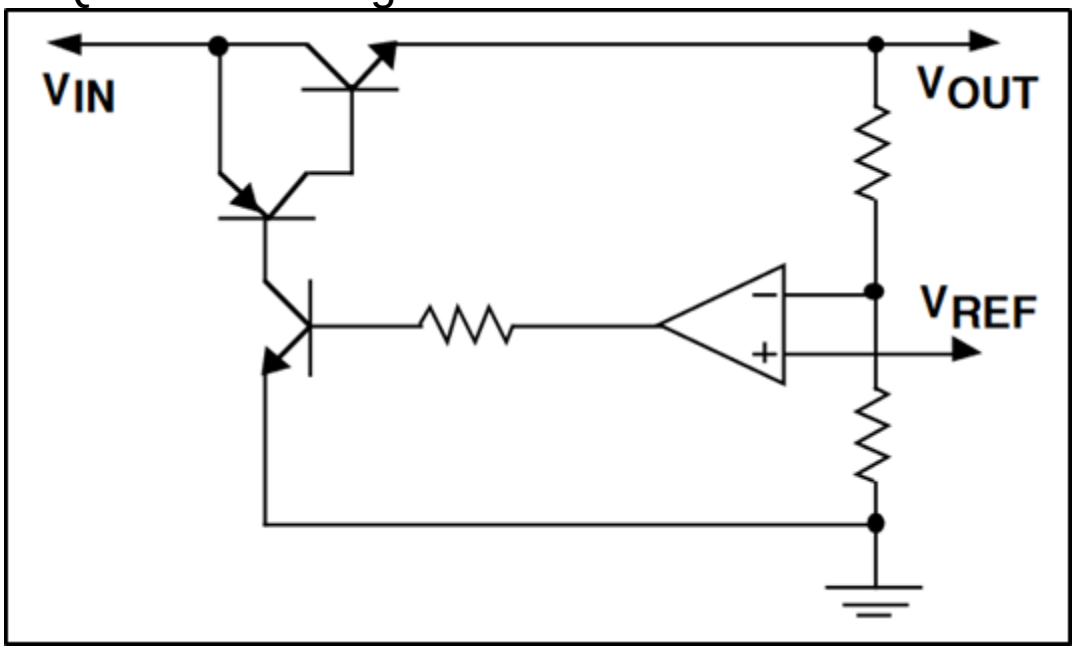


Figure 13. Quasi LDO Linear Voltage Regulator (Texas Instruments)

The Quasi LDO Regulator is a variation of the Standard Regulator, and uses a NPN and PNP transistor, as seen in Figure 13. To provide consistent output voltage regulation, the pass device needs to have a minimum voltage across it, which is given by this equation:

$$V_{D(MIN)} = V_{BE} + V_{CE} \qquad \text{(QUASI-LDO Regulator)}$$

The maximum dropout voltage for the Quasi LDO Regulator is typically around 1.5V. The dropout voltage for the Quasi LDO Regulator is exists between the dropout voltages for the Standard Regulator and the LDO Regulator.

3.2.4.4. Switching Voltage Regulators

Switching voltage regulators are generally more preferred than linear voltage regulators. Linear voltage regulators are mainly used in small-scale applications where minimal cost is an important factor. Additionally, linear voltage regulators are typically only able to operate with a relatively small voltage difference between the input and output. Linear regulators dissipate a lot of power, which creates lower efficiency, and heat sink is sometimes needed to prevent the linear regulator from overheating.

Switching voltage regulators dissipate less power, have higher efficiency, and use additional techniques such as pulse width modulation and frequency filters to regulate output voltage.

The types of switching voltage regulators that will be discussed are the Buck (step-down) converter and the Boost (step-up) converter.

a. Buck (Step-down) Converter

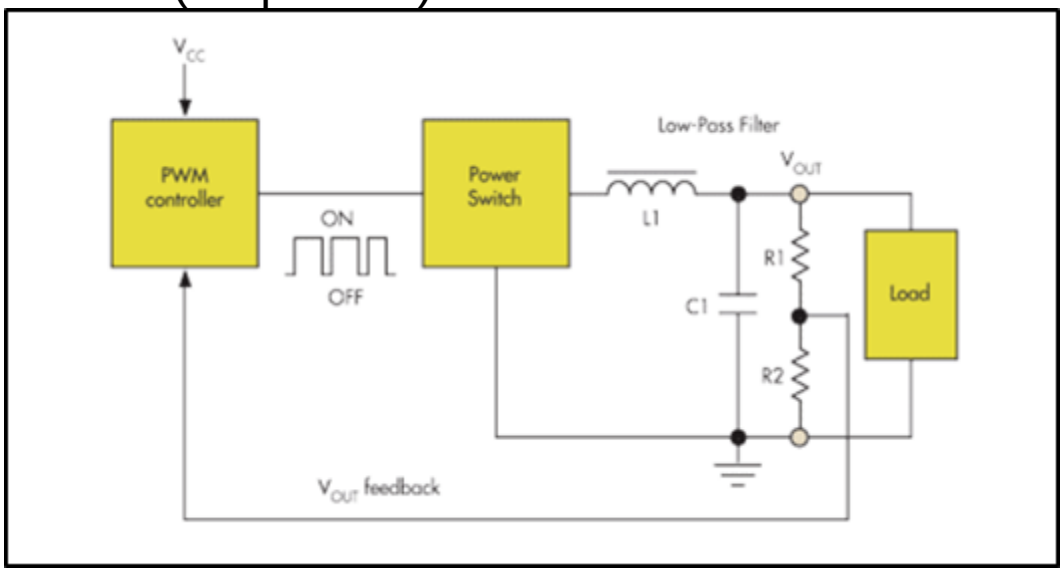


Figure 14. Buck Converter General Configuration (Powerelectronics.com)

The buck converter takes in an input voltage and produces an output voltage that is lower than the input voltage. For example, an input voltage of 8V to 25V can be converted to an output voltage of 0.5V to 5V. The output voltage is controlled through a switching technique, which regulates how voltage is received at the output for a given period. The most common technique to achieve this is pulse width modulation (PWM). PWM produces a duty cycle, which is the ratio of the amount of time the switch is on versus the total period time.

Using the input voltage and duty cycle, buck converters regulate the output voltage. The inductor stores energy as the regulator is switched on, and keeps the output voltage constant when the regulator is switched off. The capacitor is useful for minimizing the ripple voltage present. The passing device in the buck converter is made up of a transistor, typically a MOSFET.

Buck converters can be advantageous because they do not overheat (lower power dissipation) and can have very high power efficiency. For any electrical application that requires producing a lower voltage from a high voltage source, typically a buck converter of some sort is used.

b. Boost (Step-up) Converter

The boost converter functions the same way as the buck converter. Instead of producing an output voltage that is lower than the input voltage, the boost

converter produces a higher output voltage. While the input current is continuous, the load current is discontinuous due to an output diode. During the switching cycles in which low voltage (close to 0V) is supplied, an output capacitor provides the load current. The switching cycles are driven through the PWM switch model.

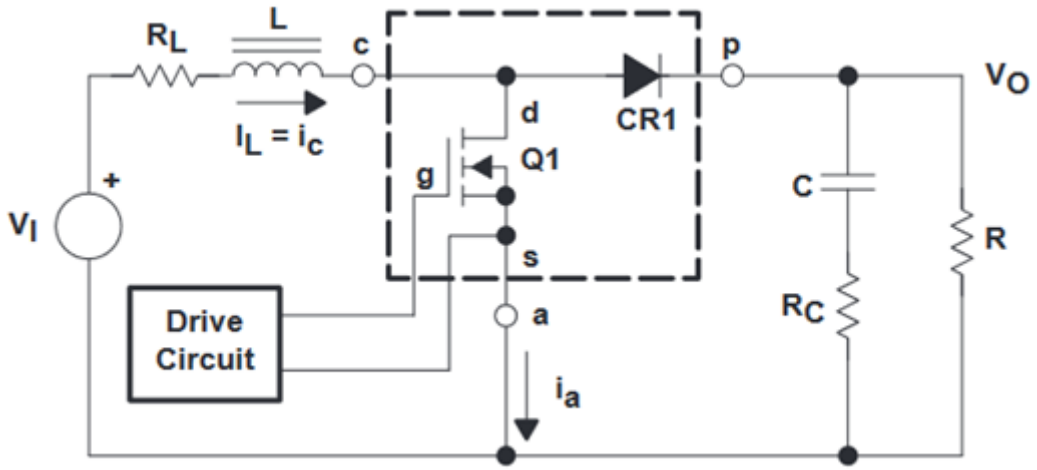


Figure 15. Boost Converter General Configuration (Texas Instruments)

The switching device is an n-channel MOSFET, as shown in Figure 15. The boost converter normally operates through the transistor constantly being switched on and off. The amount of on and off cycles is determined by the drive circuit. When the output diode is switched on, the inductor and capacitor form a filter which produces a constant DC voltage from the queue of voltages seen during the switch cycles.

3.3. Software

This section covers programming languages, frameworks, and web stack choices that could be used to develop the system. We also review different web hosting platforms and APIs that would promote the development of a robust modular system.

3.3.1. Web Stack

A Web Stack is a set of different applications software that are bundled together to host web applications. Due to the PHP Laravel and Node.js frameworks, we would either implement a LAMP stack if we use the PHP Laravel framework or a MEAN stack if we chose Node.js. While the LAMP Stack is platform dependent, the MEAN Stack is platform independent.

3.3.1.1. LAMP

The LAMP Stack is one of the more popular open source web stacks that performs basic installation and configuration of the components listed below:

- Linux: Unix-based operating system
- Apache: Web server. Will host the API and application.
- PHP: Scripting language designed for web development that can be embedded in HTML. Code is executed on the server side and outputted to the client, can return images, files, and multimedia files.
- MySQL: Relational SQL database management system.

3.3.1.2. MEAN

The MEAN Stack is also an open source, and primarily used to build dynamic web sites and web applications that supports programs written in JavaScript for both client and server side executable code. The components are:

- MongoDB: a NoSQL database
- Express.js: a web application framework that runs on Node.js.
- Angular.js: a JavaScript MVC framework that run in browser JavaScript engines
- Node.js: execution environment for event-driven server-side and networking applications

3.3.2. Framework

Due to the project requirements, the team must build a web application that allows shoppers and retail stores to interact with the system, via an application and an API for the shopping carts to communicate with the system. For easy scalability and debugging, we wish to build the API and web application using either PHP Laravel Framework or Angular.js using Node.js.

3.3.2.1. PHP Laravel

Initially released in June 2011, Laravel's creator describes it as a framework for people who value "elegance, simplicity, and readability." Laravel allows for developers to build robust systems without wasting time on routing and modularizing code, and focus more on debugging and building functional application features. Laravel offers an alternate called Lumen, for when developers are looking to build APIs. Though it is possible to build a SOAP API with Laravel, it may be cumbersome, require additional libraries/configuration, and many PHP developers disagree with creating a SOAP API and would rather create a REST API.

For our application, we require multiple libraries and APIs to implement certain features. Laravel does provide a few libraries/packages that we would need:

- Laravel Cashier: Interface to allow us to communicate with Stripe to validate, process payments, and return funds to accounts' credit cards.

- Laravel Passport: Handles login authentication, via OAuth2, so users can navigate the website only signing in once. Also allows for easy set up of API authentication, since APIs typically use tokens for authentication.
- Laravel Echo: Allows for the system to update a client's webpage without having to reload the page. Using WebSockets, the system can post updates to the user page instead of having the client continuously poll the server for updates.
- Laravel Mail: Interface to SwiftMailer to easily send and receive emails, however, we only require this for sending emails to verify accounts and provide a pdf copy of their receipt.

Additional open source libraries that we require, not provided by Laravel, are:

- Snappy: PHP5 wrapper library to interface with wkhtmltopdf that allows thumbnail, snapshot and PDF generation. Can generate pdfs from url(s), webpages, or build the data to store in the pdf. To guarantee all text from the webpage is rendered in the pdf, the following system-level libraries will be needed: xfonts-base, xfints-75dpi, and urw-fonts installed.
- Twilio: Voice, Video, Messaging, and Authentication REST API. Would be used to text user a confirmation of any purchase recently made on their account.
- Zbar-qrdecoder: PHP wrapper for zbar-tools to decode barcodes and QRcodes scanned. Supported formats are: Code 39 (encode 43 ASCII characters); Code 128 (encode 128 ASCII characters); EAN_13 / ISBN (13 digits).
- DomPDF: This is a backup pdf generator that will be utilized if our team is unable to setup Snappy. It can only render what is on the screen and output it to the browser. The only limitation is it can take a while to render larger files with lots of data.
- Webcamjs: HTML5 Webcam image capture. Will be used to allow users to see where they are aiming their camera when aiming at barcodes.

Laravel implements the Model-View-Controller design pattern and maps URL routes to controller actions (unique functions). Although, Laravel provides a directory for storing Controllers and Views, it is up to the developers to decide where they would like to store their models. Furthermore, Laravel uses a template engine, called "blade", to bind data returned from the database or model to objects stored in the view, and insert multiple partial-views into a parent-view, allowing developers to delete redundant code.

For security, Laravel provides additional functionality to assist developers in writing code that validates and sanitizes user input. Developers can also configure their project to encrypt any sensitive data, only permit HTTPS connections, and securely hash passwords/tokens.

3.3.2.2. Node.js

Node.js is not a framework, but rather a JavaScript runtime environment built on Google Chrome's JavaScript V8 Engine. Developers use Node.JS for applications that must be fast, event driven, and handle many requests. Unlike other frameworks, Node.js promotes non-blocking programming to remove buffering and cutback on CPU and memory resources.

On a high level, Node.js runs the same way as a traditional web server, however, Node.js runs different under the hood. Normally, when a web server receives a request to perform an operation it will create a new thread, perform all necessary operations, return the data, and end the thread. While Node.js runs on a single event thread that runs any non-blocking operations and passes any I/O blocking to a thread pool, it is the developer's responsibility to write asynchronous code by implementing callback functions to be called once a function finishes running. Though developers can write blocking functions, all Node APIs support callbacks and all developers promote implementing callbacks.

Although Node.js is a moderately new framework, there is a vast amount of documentation and APIs available to effectively implement any system. Using the NPM package manager, we can install any necessary modules to assist in the development of the desired system.

For this project, we need to the following modules/libraries:

- Express.js: Light weight Node.js web application framework with features to support web and mobile applications. Will be used for the API portion of the system and hosting the web application for customers to use.
- Express-jwt: Middleware module that validates JsonWebTokens. Used to protect API endpoints (client & server) instead of just over the 'wire', since it is not possible to implement sessions in APIs. Allows developers to prevent malicious users from adding products to other shopping carts without permission.
- Bcrypt.js: module for securely hashing passwords and comparing login passwords. Allows us to store hashed version of passwords.
- MongoDB driver: Used for creating Schemas/Models component in MVC and communicate with MongoDB database.
- Passport.js: Flexible middleware module to authenticate login attempts via username & password, Facebook, Twitter, etc.
- Stripe API: REST API used for validating, storing, and charging account credit cards. Used to stay compliant with US laws for keeping users' credit card information secure. It cost 2.9% of transaction, plus 30 cents, to charge the card, but with GitHub's student pack, the transaction fees are waived for the first $1000 processed.

- Qrcode-reader: Module to capture QR-code on shopping carts and decode information, which will request login credentials, if necessary, and bind the shopping cart to the user.
- Webcamjs: HTML5 Webcam image capture. Will be used to allow users to see where they are aiming their camera when aiming at barcodes.
- Web-push: module to implement push notifications so app owners can be alerted without having the app open. Will be used to notify users their payment was successful, and any other information that seems important to notify users immediately.
- SendGrid API: API that can send/receive emails for a system with attachments and real-time event notification WebHooks. Will be used to send users a copy of their receipt of their recent purchase to their email. With the GitHub student pack, we can send 15,000 emails a month for free, however, normal pricing is a minimum of $10 per month, and allows users to send 40,000 emails.

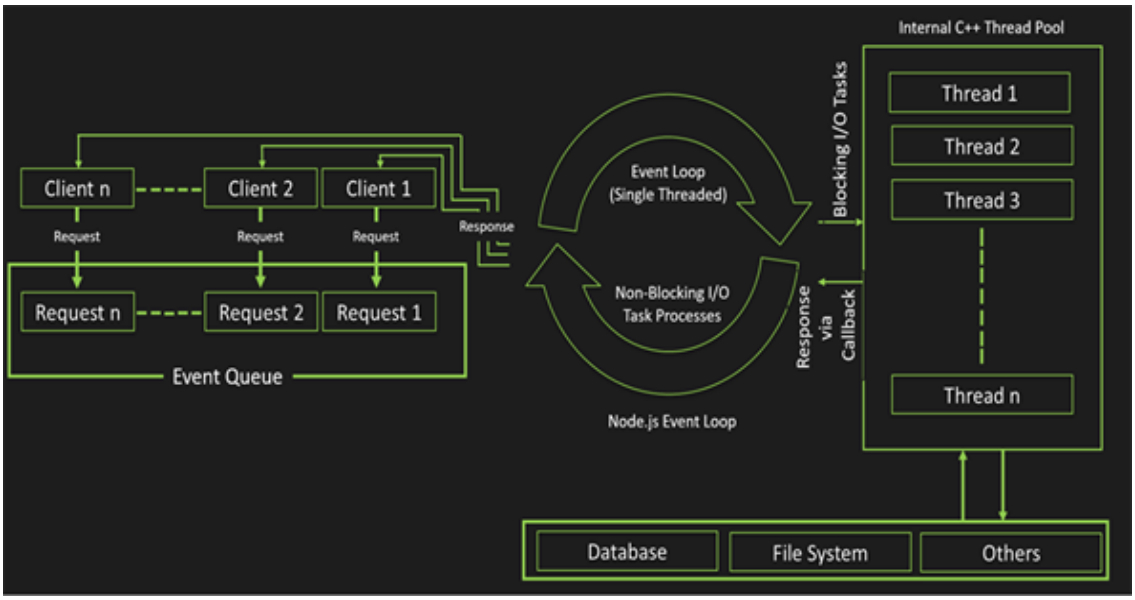


Figure 16. Node.js Architecture

3.3.2.3. Angular.js

Angular.js is a JavaScript based frontend web application framework maintained by Google, which is used to develop single-page applications. The goal is to develop applications that have the similar UI/UX as desktop and native mobile applications. Angular.js solves the issue of HTML not being designed for dynamic views, saving processing power for manipulating a page's Document Object Model (DOM). The Angular application is built with HTML templates with angular tags, component classes to manage data being presented in HTML templates, and logic to handle actions. When the Angular application is launched, the bootloader will

inject the necessary Boot Strap classes and tags. The main building blocks in an Angular application are:

- Modules: Classes that store information of the data that will be presented on the web page. Initially, the application has only the root module, but you can always create additional modules for your application. Modules are defined with the @NgModule.
- Components: Controls a specific view on the screen. In the component class, you define any component application logic that handles user interaction, i.e. load the next set of elements in table when next button is pressed. Throughout an application's lifecycle it is constantly creating, updating, and destroying component instances.
- Templates: Defines a component's view. Although much of the code is basic HTML, there are tags you can use to bind data or functions to call in the component to show to the user.
- Metadata: Tells Angular how to process a class, and information of the page, like dependencies, what template the component binds to, etc. With properly adding metadata information, you can see improved performance.
- Data binding: Binds the data from the components to the view, saving developers lots of time and frustration writing jQuery code to handle user actions. You can bind elements from the component to multiple elements in the view with one displaying the data, and another part allowing users to edit the data. As the users modify the data, its changes to also being changed throughout the rest of the page.
- Directives: Tags that manipulates the DOM when angular renders the selected template.
- Services: Broad term to describe any additional features an application requires. Instead of having components filled with code for fetching data from cache, databases, and APIs, we store all this logic in services and call designated functions from the component class. There are many open-sources services for handling caching and network I/O operations.
- Dependency injection: Solves any dependency issues a newly created component may have. Most of the time components are dependent on services. This is done by reviewing constructor parameter types and looking at the services listed in the provider array. Providers can be included in modules and components.

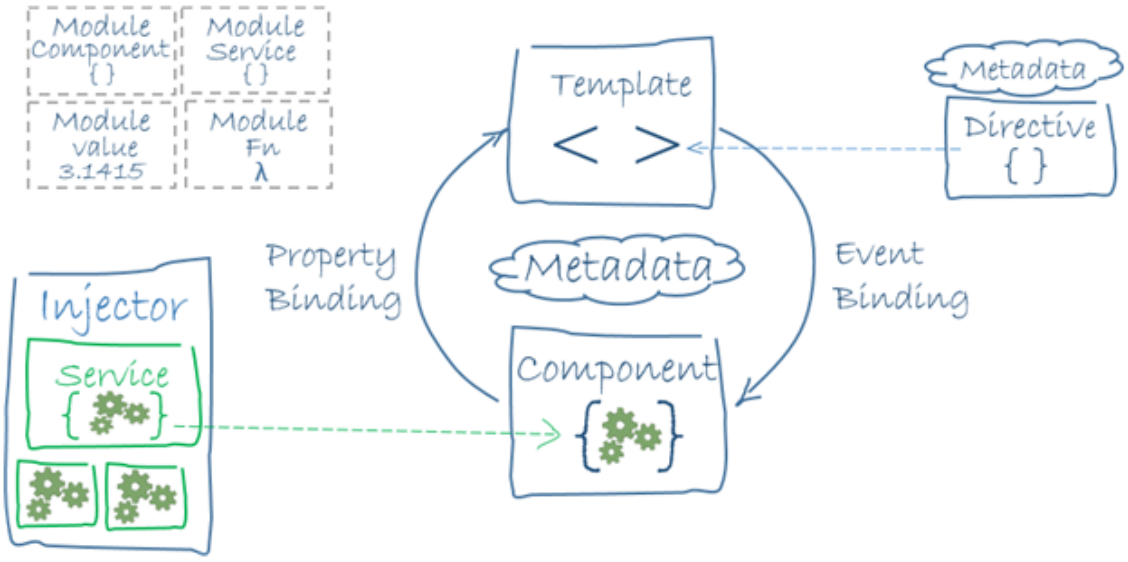


Figure 17. Angular Application Components

3.3.2.4. Choice

Although some team members have more experience with developing websites in PHP, we believe it is better if we proceed to build the system with Anguler.js, using Node.js as the development environment. Although there is a substantial amount of documentation for PHP and Angular.js, there are many security flaws in PHP that are nonexistent if we proceeded with Angular.js. We can implement asynchronous calls in Angular.js which can improve performance, and more APIs are available for Node.js applications.

3.3.3. API

An Application Programming Interface (API) is a program that other programs can use to solve smaller problems. For this system, we need a web API that allows the devices on the shopping cart to communicate with the system to update a customer's shopping cart. Two possible solutions are a SOAP and REST API.

3.3.3.1. SOAP

Simple Object Access Protocol (SOAP) is an XML-based messaging protocol to communicate with another computer via HTTP protocol. A SOAP message includes the following elements:

- Envelope: A mandatory element that signals the start and end of the message.
- Header: An optional element that stores data used in processing the message.
- Body: A mandatory element that contains XML data being sent.

- Fault: An optional fault element that provides information about errors that arose during code execution. Found in the body element of the SOAP message envelope.

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
 SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >

   <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotations" >

      <m:GetQuotation>
         <m:QuotationsName>MiscroSoft</m:QuotationsName>
      </m:GetQuotation>

   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>


HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
 SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >

   <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotation" >

      <m:GetQuotationResponse>
         <m:Quotation>Here is the quotation</m:Quotation>
      </m:GetQuotationResponse>

   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Figure 18. SOAP Example: Request and Response

When implemented with HTTP, developers don't have to worry about existing firewall policy being modified to allow post/response exchanges to be dropped. SOAP also has built in error handling and is language, platform, and transport protocol independent (SMTP, etc.). However, due to slow parsing speed of XML, which will require additional resources that are already limited on the device on the shopping cart, which could affect its functionality.

3.3.3.2. REST

Representational state transfer (REST) is a stateless web service that allows easy messaging where each HTTP verb maps to a unique action. Review the table below for information for the HTTP verbs and an example.

Table 5. REST API

Uniform Resource Locator (URL)	HTTP VERBS			
	GET	PUT	POST	DELETE
http://api.site.com/books/	List of all books recorded	Replace entire book collection with new set of books	Add a new book into records. Data for the new book is stored in the body	Delete entire book collection
http://api.site.com/books/1232	Retrieve data about book that has the id of 1232	Update specified information for book with id 1232. Information to update book is stored in the body	Standardly not used when addressing an object	Delete the book with the id 1232 from the collection

With a REST API, its common to send data in the body in JSON format, using key value pairs to represent data. Although REST APIs can easy scale up and not require time for parsing data, developers must write additional code and require more data to be sent to authenticate clients when they necessary since REST APIs don't implement sessions.

3.3.3.3. Choice

Due to Node.js default implementation of JSON, our API can be stateless, and resources are important to shopping cart devices and the system we believe that

a REST API is the best choice. It's less resource intensive if data being fetched from a MongoDB collection since the data is JSON formatted, so don't need execute extra code to parse JSON formatted data to XML format. Although, developers won't spend time checking XML messages are properly formatted, they will need to write error handling and message code.

3.3.4. Web Sockets

Traditionally clients were required to send requests to the server to receive new data. Introduced in 2001 the WebSocket protocol allows developers to implement server-side events, where the server can push new data to client computers at any moment. Allowing developers to writing AJAX request code to have clients continuously poll the server, thus greatly diminishing server overhead. The WebSocket connection begins as the HTTP(S) handshake occurring. Currently, the WebSocket Protocol is supported my Google Chrome, Firefox, Safari, and Microsoft Edge. With WebSocket Protocol we can easy design the system to send new data to client devices whenever the device on the shopping cart detects items being added/removed from the shopping cart, or push messages to their device to show confirmation upon successfully connecting to the shopping cart and completion of checkout.

Due to project requirements, it seems that socket.io is the best WebSocket API for our project. It works with Express.js and allows implementation over HTTPS. Above all, socket.io allows the system to send data to single, groups, or all users currently on connected to the system.

3.3.5. Database

The system being developed will require a database that manages account, store, product, and coupon information. In this section covers SQL and NoSQL database systems.

3.3.5.1. SQL (RDMS)

SQL was made in the 1970s and is used by database administrators and developers that what to setup and run analytical queries. Some popular examples of SQL databases include MySQL, Postgres, Microsoft SQL, and Oracle. SQL is used for modifying database tables and index structures. This includes adding and updating rows of data and getting information from within a database. SQL operations such as queries normally take the form of commands written as statements, such as select, insert, create, and add.

SQL databases are relational systems that are made up of tables comprising of data formatted in rows and columns. Each column in the tables relates to a category of data such as manager or address. While each row contains data about an individual or thing that the table is keeping track of. So for example who is John's manager or what is John's address. This format is very similar to a spreadsheet in Excel and is easy for first time users to understand.

The scaling for SQL databases is done vertically. That means that in order the deal with increased traffic in the database a single server needs to become increasingly more powerful. Although it is not impossible to have a SQL database spread out over many servers there is usually a substantial amount of extra engineering necessary in order to do so. Furthermore, many core features such as JOINs, referential integrity and transactions tend to become lost in the process.

In the schema for a SQL database the structure and data types are setup beforehand. In addition to store data about a new data item, such as a phone number, the complete database needs to be changed. This requires the database to be taken offline which means that the applications will not be able to access during that time period. This can be very irritating for the users who are trying to use the application and it might discourage them from using it in the future. Therefore, it is important that the people who are developing the database to be fully aware of its function and thoroughly analyses it.

The development model for SQL is a mixture of open source, such as Postgres or MySQL, and closed source like the Oracle database. SQL has also been developed for strong consistency which means that although ta person might read old data from a table when the replication system lags they will never read partial or inconsistent data. That happens when only part of the data has been changed that means the user will either get the old data or the new data but never a mixture of the two.

3.3.5.2. NoSQL

NoSQL was created in the late 2000s because of the limitations of SQL databases. These limitations include scalability, geo-distribution, and agile development sprints. Some examples of NoSQL databases include MongoDB, Cassandra, and HBase.

There are also a variety of different NoSQL databases that overcome this limitation such as document databases, graph stores key-value stores, and wide-column stores. Document databases work by pairing a unique key with a complex data structure called a document. These documents can comprise of various diverse key-value pairs, key-array pairs, or nested documents. Graph stores are used to hold information about networks of data. An example of this is a graph of people and their friendships. An example of Graph stores are Neo4J and Giraph. In Key-value stores every item in the database is stored as a key with its value. This is the simplest of all the NoSQL database types and some examples include Riak and Berkeley DB. Wide-column stores are improved for queries over vast datasets and store columns of data together. Some examples include Cassandra and HBase.

Compared to SQL databases NoSQL databases has improved scalability and better performance. Furthermore, NoSQL databases' data model addresses several issues that the SQL model fails to address. These issues are as follows:

- Large amounts of swiftly varying structured, semi-structured, and unstructured data

- Agile sprints, rapid schema iteration, and recurrent code pushes
- Flexible and simple to use object-oriented programming
- Geographically distributed scale-out architecture

As stated earlier for SQL databases the schemas must be defined before any data is added. This works poorly for agile development because each time a new feature to the product is complete the schema for the database has to be changed. That means that adding a new column to a SQL database requires the developer to migrate the whole database to the new schema and if the database is a sizable one this could take an unreasonably long amount of time. NoSQL databases, however are made to allow for data to be inserted without a predefined schema. That means making substantial application changes in real-time without service interruptions is an easy process. This tends to make development faster and code integration more reliable.

NoSQL databases tend to support auto-sharing. The databases automatically spread data across an arbitrary amount of servers without demanding the application's awareness of the configuration of servers. The data and query load are evenly distributed among the servers and if a server goes down it can be swiftly replaced without application disruption.

Table 6. Database comparisons

	SQL Databases	**NoSQL Databases**
Examples	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Data Storage Model	Records stored as rows in tables, with each column storing a specific piece of data about that record.	Document in JSON, can store records in nested records.
Schemas	Fixed	Dynamic
Scaling	Vertically, spread SQL databases over many servers.	Horizontally, database automatically spreads data across servers as necessary.

	SQL Databases	NoSQL Databases
Supports Transactions	Yes, updates can be configured to complete entirely or not at all	In certain circumstances and at certain levels (e.g., document level vs. database level)
Data Manipulation	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs

3.3.6. MVC

Model-View-Controller (MVC) is a software architecture pattern that promotes efficient code reuse. MVC has 3 components:

- Model: Logic that stores data retrieved from database or other resources. (What to display)
- View: This is the frontend portion of the system that includes CSS, HTML, and JavaScript code. This is the component that allows users to interact with the system to fetch or edit account information. (How it's displayed)
- Control: Handles actions for retrieving data, modifying fetched data, and saving data modified by user. (format model for display)

Figure 19. MVC Diagram

As seen in the diagram, the view should only trigger events, also known as call functions, in the controller. The controller then performs the desired user actions after making sure the user is authorized to perform specified actions. In cases where users are adding or modifying data, and the model isn't returning any data, it is good coding standard to return feedback if the changes made were successful.

A controller can return a view when redirecting to a page, model when reloading a page, or additional data on an object or a list. In practice, it is best to load a View with an empty model, then fetch the data requested to decrease load time and improve memory efficiency.

3.3.7. Applications

For a project we require an interface that allows customers to easy interact with the system to view previous purchases, available coupons, and a running total of their shopping cart. We are focusing on the Android Platform since a majority of the group has an android phone, and a team member has previous experience developing Android apps. We are looking at developing either a native Android application or a Progressive Web App for users to interact with the system.

3.3.7.1. Android App

Since Android's initial release September 23, 2008, Google has developed a mobile operating system that incorporates security and easy usability, adding additional features every release. From the diagram, we can see that many people are still using previous releases of Android. Thus, we would have to design the app to be able to run on Lollipop and Marshmallow.

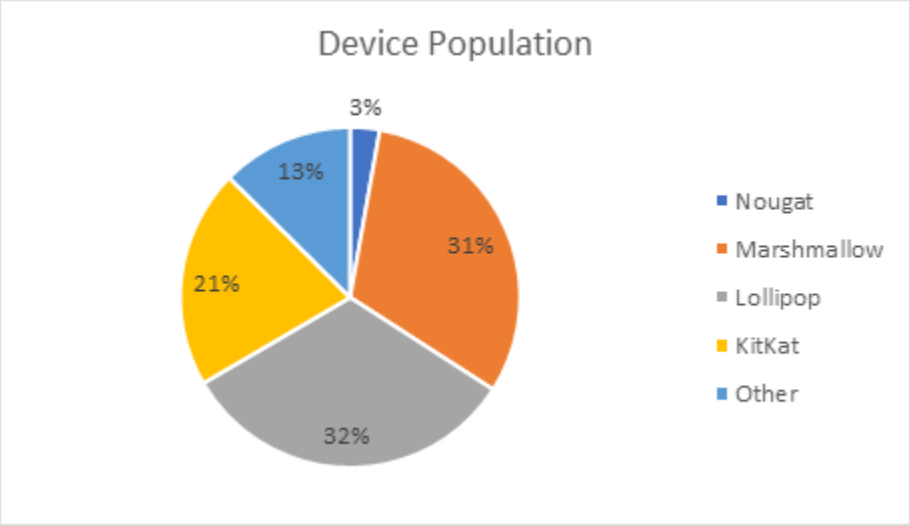


Figure 20. Device population

a. Native Android App

In an Android application, there are four essential types of components, each with a different entry point which the system or user can enter the app, serves a distinct purpose, and has a unique lifecycle. The four app components are:

- Activities: The entry point for interacting with the user that holds the logic for a single screen view and lifecycle. Activities can communicate with other activities in an application to present cohesive user experience, they can also be designed to allow other apps on the phone to start a given activity.
- Services: A general purpose entry point to keep the app running in the background or perform certain operations in the background instead of the main thread. Services are used when apps must communicate over

the network to fetch/upload data to a server, or listen to a song from music app while playing a game on another app. There are Started Services that tell the system to continue running the service in the background until its done. A Bound Service allows for other applications to use it, so the system knows that there is a dependency between applications if App-A called a service in App-B, so it knows it must keep App-B running for App-A.

- Content providers: Manages a shared set of app data that can be stored on the web or persistent local storage that the app can access. They allow apps to share data securely, by allowing them to initiate intents to a specific app or any app that meets the requirements. However, applications don't have the permissions to directly active a component from another app, so they must message the system with the intent saying to start a component.
- Broadcast receivers: Enables the app to respond to system-wide broadcast announcements. Although most broadcast originate from the system notifying apps about device information, other apps can also broadcast to other apps notifying them data is available for other apps to use. Any request made is received in an Intent object.

Using the Manifest.xml file, developers must define the components, permissions, and dependencies required to run the application.

b. Cordova

Apache Cordova is an open-source mobile development framework that allows web developers to write applications that can be ran on various platforms using (HTML), Cascading Style Sheet (CSS), and JavaScript. The application will execute in a designated wrapper and rely on the standards-compliant API bindings to access the device's sensors, data, network, etc. Cordova is for projects where developers want to extend their application across multiple platforms without spending countless hours rewriting code in each platform's language.

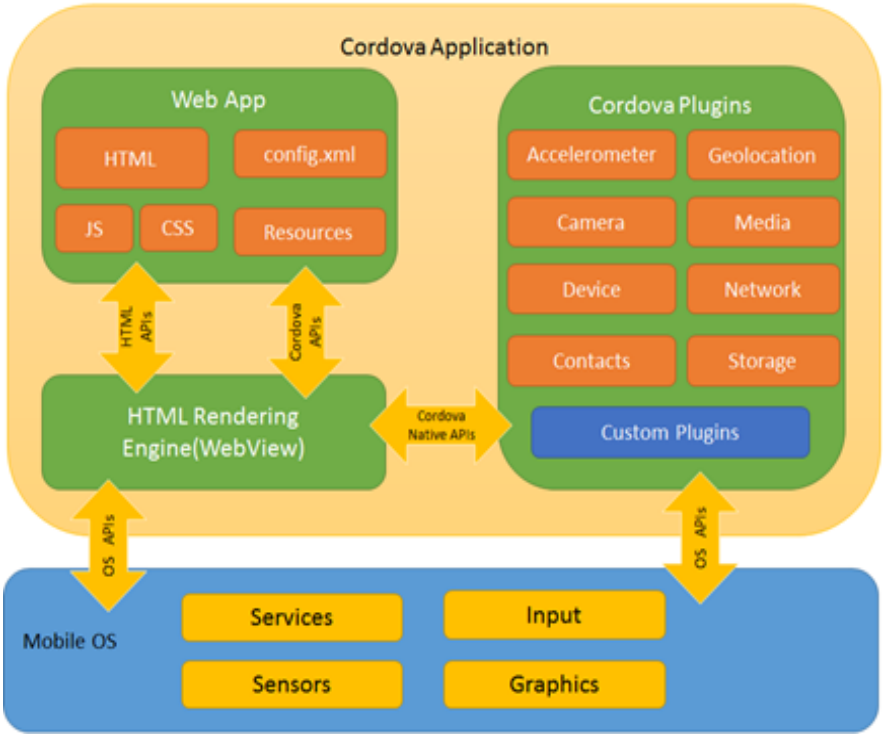


Figure 21. Cordova Application Architecture

Seen in the above diagram, there are three components to a Cordova Application:

- WebView: Handles how application will be presented, developers can also embed Cordova WebViews in native hybrid application components on some platforms.
- Web App: Where all the code for the application resides, there is a default index.html file that references CSS and JavaScript and other resources necessary to run. The config.xml is a crucial file that provides information about the app and parameters that affect how the app runs.

Plugins: Provide an interface for Cordova and native components to communicate with one another allowing the app to invoke native code from JavaScript. Cordova provides a core set of plugins to interact with the devices, battery, camera, contacts, and more, but we can also include other plugins to abstract any additional native code required to communicate with the device's components.

c. Progressive Web Application

In late 2015 Alex Russell a Google Chrome engineer and designer Frances Berriman defined the concept of a Progressive Web App as an application that takes advantage of features offered modern browsers that function like a native application. A successful Progressive Web App must be:

- Reliable: Regardless of network conditions the application must load instantly or perceive to look like its loading. Written in JavaScript,

service workers solve this issue and put the developer in control of the cache and how to respond to resource requests. This allows users to access the application if they are offline. Because service workers can hijack connections and fabricate responses, you can only use service workers on pages served over HTTPS so we know the service worker the client device receives hasn't been tampered.

- Fast: The app must quickly respond to user interaction and smoothly transition to other pages within 2 seconds, users should never see a white screen. We can optimize how code is delivered to clients to improve delivery time and rendering performance. Using service workers and App Shells, we can give the illusion of performance by loading the template with data from the cache while we wait for a network response with the data we queued instead of a white page. Every time data is fetched from a server we should always save that data in the cache.
- Engaging: Users can install the app, which will live on the home screen, without the need for accessing the app store. The Manifest file defines the app icon on the home screen and how the app will render full screen without showing the address bar.

Figure 22. PWA Shell App

3.3.7.2. Choice

While some group members have experience developing native Android applications with Java, we are deciding to build the application as a Progressive Web App. Unlike an application built with Cordova, a Progressive Web App doesn't require additional libraries to communicate with the device since we are running in the Chrome browser and using their APIs to communicate with the device, allowing our application to be platform independent without having to include additional libraries which could affect the application's performance.

3.3.8. Web Service Provider

This section will be used to discuss the different options for Web Service Providers that our team considered to employ in our SmartKart shopping system. This

includes an in-depth discussion on Amazon Web Services and Google Cloud Platform.

3.3.8.1. Amazon

Amazon Web Services is a cloud computing platform created by Amazon.com. It originally offered online services for websites and client-side application. Now, Amazon Web Services offers over three dozen services, including CloudDrive, Dynamo Database, Elastic Compute Cloud, and more. All of their services are billed according to how much it is used, although the rates vary from service to service.

A cloud service platform such as Amazon Web Services provides rapid access to IT resources. Furthermore, with cloud computing there is no expensive upfront investment in hardware, and there is no time needed to manage the hardware. Instead, there is only a need to pay for what computing resources are being utilized. Amazon Web Services, as well as other providers, may have higher economies of scale, which means there can be lower prices for their users.

Amazon Web Services allows the user to pick whatever platform they want. It also provides a software development kit for the more popular platforms, such as Java and Node.js. Furthermore, Amazon Web Services has no restrictions on which content management system is used for hosting the website; it may be anything from WordPress to Drupal.

Amazon Web Services provides an elastic cloud computing infrastructure, which allows for the website to grow or shrink based on the amount of traffic that a website is receiving. There are also many datacenters around the world which allow Amazon Web Services' customers to host their website globally within minutes.

Amazon also provides Amazon CloudFront, which is a content delivery network. It can be used to deliver the entire website using a global network of edge locations. The requests for website content are routed to the closest edge location so the content is delivered as quickly as possible. Amazon Web Services also provides a "Trusted Advisor" to give the user automated analysis and recommendations for the website.

Amazon Web Services offers a "Free Tier", which allows its users to obtain free hands-on experience with the Amazon Web Services platform and services. These free services include, but are not limited to:

- One million requests per month using Amazon Web Services Lambda: This is a compute service that will run the code in response to events. It will also automatically manage the compute resources

- 25 GB of storage in DynamoDB: A NoSQL database[c1]
- 20,000 free requests per month of Amazon Web Services' Key Management Service: This service provides encryption with administrative controls 250 device minutes from Amazon Web Services' Device Farm so the users can test their iOS, Android, and FireOS apps on real devices in the Amazon Web Services cloud.

DynomoDB is Amazon's NoSQL database service. DynamoDB contains features such as document data model support. So the user can store, query, and update documents and by using AWS SDK the user can write applications that store JSON documents directly into the DynamoDB tables they have setup. This reduces the amount of new code needed to be written to insert or update a JSON document.

DynamoDB also provides its users with key-value data model support. With this each row is a key value pair where the primary key is the only requisite attribute for the rows in a table. Since DynamoDB is schema-less each item or row can have a limitless amount of attributes. Using Global Secondary Indexes and Local Secondary Indexes the user can query non-primary key attributes.

DynamoDB has continuous throughput and storage scaling with API and the AWS Management Console. So the amount of throughput or storage that the user can use is limitless. Furthermore DynamoDB is equipped with automatic and synchronous data replication in a total of three facilities in a Region. Therefore the user's data is protected against any individual or facility level machine failures.

DynamoDB has a downloadable version so that the user can develop and test their code on their machine beforehand. Later DynamoDB is able to scale the application on the cloud for global use.

With DynamoDB the user can use secondary indexes that they create to efficiently search any attribute. The user can of course delete and add these indexed for their table at any time. Moreover DynamoDB Streams allows the user to keep track of the latest item level change. In addition the user can receive all the item level updates that has taken place in the last 24 hours.

DynamoDB supports cross-region replication across the AWS regions. This allows users to build globally distributed applications with faster data access, improved traffic management, simpler disaster recovery, and easy data migration. DynamoDB has been integrated with AWS Lambda to deliver Triggers to the user. With Triggers the user can automatically execute a custom function when item level changes are detected in a table.

DynamoDB has also been combined with Titan to give the user the ability to proficiently store and traverse graphs regardless of their size. The graph

databases are optimized for quick travels of a verity of relationships. Unlike many NoSQL databases DynamoDB development process is simpler by allowing the user use strong consistency on reads so that they get the latest values from the database.

DynamoDB uses established cryptographic methods to authenticate users and prevent unauthorized access to data. DynamoDB is also incorporates AWS Identity and Access Management for access control for users within the user's organization.

Amazon's Relational Database Service, or RDS, is Amazon's SQL database service. Amazon's RDS provides the user with several familiar database engines to choose from. Some of these include Amazon Aurora, MySQL, Oracle, and PostgreSQL. Meaning that the code, applications and other items that the users already uses with their current database can be used with Amazon's RDS. Amazon's RDS can handle everyday database tasks like provisioning, patching, backup, recovery, failure detection, and repair.

Amazon's RDS is simple to use because it allows the user to use the AWS Management Console, the Amazon RDS Command Line Interface, or API calls to access the abilities of a production-ready relational database. Amazon's RDS instances are also pre-configured with parameters and settings suitable for the engine and class that the user has selected.

At no additional charge Amazon's RDS provides its users with Amazon CloudWatch metrics for their database instances. They can use it to view metrics such as I/O activity and compute/memory/storage capacity utilization. Furthermore Amazon's RDS provides the user with Enhanced Monitoring which gives them access to over 50 CPU, memory, file system, and disk I/O metrics. Another feature that Amazon's RDS provides its users is automatic software patching. This ensures that the user's relational database software stays up-to-date. The user can employ optional control over when and if their database instance is patched.

Amazon's RDS has easy storage scaling. The user will be able to setting up additional storage for when their storage requirements grow. The Amazon Aurora automatically grows the size of the user's database volume as needed. The engine will continue to do this until it reaches 64TB or a maximum the user sets. MySQL, MariaDB, Oracle, and PostgreSQL allows the user to provision up to 6TB of storage on-the-fly with zero time lost.

Amazon's RDS has an automated backup feature that is turned on by default. This allows for a poit-in-time recovery of a user's database, should anything happen to it. Amazon's RDS backs up the user's database and transaction logs. These are

then stored for a holding period specified by the user. The automatic backup retention period can be configured to up to 35 days.

Amazon's RDS also has a database snapshots feature. The database snapshots are user-initiated backups of the user's instance that are stored in Amazon S3. These snapshots are then kept in safekeeping until the user explicitly deletes them. Users can create new instances whenever they want using a database snapshot. Database snapshots function operationally as full backups however the users will only be billed for the storage used.

Amazon Web Services' IoT is a platform that allows the users to connect devices to other AWS Services and other devices. It also provides the user with secure data and interactions, process and act upon device data, and permits applications to interact with devices even while they are offline.

AWS IoT has a software development kit, known as SDK, to help the user rapidly connect their hardware device or mobile application. Using MQTT, HTTP, or Web Sockets protocols, the SDK also allows the users device to authenticate and exchange messages with IoT. The SDK supports the use of C, JavaScript, and Arduino. It also includes the client libraries, developer guide, and porting guide for manufactures. However if the user prefers they can use an open source SDK of their choosing as well as write their own SDK.

Device Gateway is also a feature with AWS IoT. This allows for devices to securely and efficiently communicate with the IoT Using a publication/subscription model, the Device Gateway can exchange messages this allows for both one-to-one and one-to-many communications. AWS IoT one-to-many communication sequence allows for a connected device to transmit data to multiple subscribers.The Device Gateway supports protocols from MQTT, WebSockets, and HTTP 1.1 and it is also simplistic for the user to implement support for proprietary/legacy protocols.

In order to insure that data is never swapped between devices and AWS IoT without proven identity IoT has mutual authentication and encryption at all points of connection. The AWS method of authentication, known as SigV4, as well as X.509 certificate based authentication is supported by AWS IoT. Both of these methods can be used if the connection is using HTTP however if the connection is using MQTT only use certificate based authentication and connections using WebSockets uses SigV4. Users can utilize AWS IoT generated certificates and also certificates signed by their preferred Certification Authority. The users are also able to map their choice of role and policies to each certificate. Therefore users can authorize devices or applications to have access. Furthermore users can also revoke access without ever touching the device.

Amazon Web Services' Lambda is a serverless compute service. It runs the user's code in reaction to events and manages the core compute resources automatically. AWS Lambda can extend other AWS services such as Amazon DynamoDB or Amazon S3. AWS Lambda can run code to respond to multiple events like buckets or table updates in Amazon DynamoDB automatically. The code is run on a high-availability compute infrastructure and performs all the administration concerning the compute resources, code patch deployment, security patch deployment, code monitoring, code logging, as well as capacity provisioning and automatic scaling for the user.

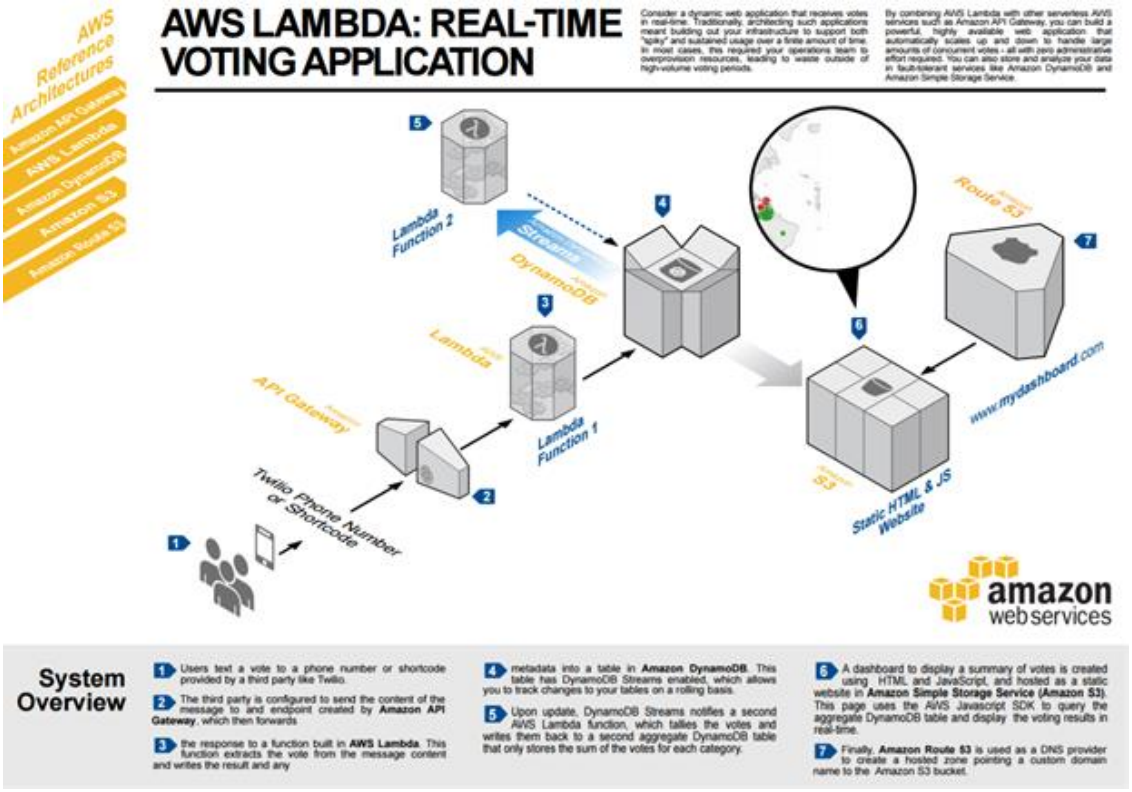


Figure 23: Example of System running on AWS

As previously mentioned AWS Lambda can extend other AWS Services. AWS Lambda does this by allowing the user to add custom logic to AWS resources like Amazon DynamoDB tables. This makes it simple to apply compute to data as it enters and moves around the cloud.

To start implementing Lambda the user first needs to either upload their code or write it directly into the Lambda console. Then the user just needs to select the memory, timeout period, and AWS Identity and Access Management role. After that the user only needs to specify the AWS resource to trigger the function, for example the resource could be a DynamoDB table. Once the resource, in this case the table, changes Lambda will run the user's function as well as launch and

manage the compute resources as needed in order to stay on top of incoming requests.

Amazon Web Services also provides Cloud Security to its users. Amazon Web Services' data center and network architecture has been built to meet the requirements of some of the most security-sensitive organizations. AWS cloud also allows users to scale and innovate, while keeping a secure environment. Users only pay for the security services that they use so that mean that users will tend to have a lower cost and no big upfront expenses as they would have if they used an on premise environment.

In infrastructure security AWS provides its users with numerous security capabilities and services to increase privacy and to control network access. One of these is the network firewalls that are built into Amazon VPC. There are also web application firewall abilities in AWS WAF. This allows the user to create private networks and control access to their products. There is also encryption in transit with TLS throughout all of AWS's services. The user also has several connectivity selections that permit private or even dedicated connections from their office or on premise environment.

AWS has built all their services and technologies from the ground up for resilience against DDoS attacks. The grouping of AWS services can be used to implement defense strategies and hinder any DDoS attacks. The AWS services designed with automatic response to DDoS attacks help reduce the time to mitigate and diminishes the impact.

AWS gives the user the capability to add a layer of security to their data by providing them with scalable and effective encryption features. Some of these encryption features are as follows:

- Data encryption abilities accessible in AWS storage and database services like Oracle RDS.
- Flexible key management options that allow the user to choose to have AWS manage the encryption keys or to control the keys themself.
- Hardware-based cryptographic key storage using AWS CloudHSM

AWS also has APIs that allow the user to incorporate encryption and data protection with any of the services they create or use in an AWS environment.

AWS also has tools that allow their users to see what is happening in their AWS environment. There is visibility into APL calls through an AWS service called CloudTrail that contains details such as who, what, and from where calls were

made. Furthermore, there is log accumulation options that simplify investigations and compliance reporting. Amazon CloudWatch alerts the user when particular events transpire or thresholds are surpassed. All these tools and features provide the user the visibility needed to foresee issues before they arise and impact the business. It also gives the user the chance to improve their security standing and reduce their risk profile.

Amazon Web Services also has services that offer security measures for identity and access control. One of these services is creatively named AWS Identity and Access Management also known as IAM. With IAM the user can define individual user accounts with permissions throughout AWS resources. AWS also has AWS Multi-Factor Authentication for restricted accounts, this even includes selections for hardware-based authenticators. There is also AWS Directory Service that allows the user to incorporate and merge with corporate directories to minimize administrative overhead which in the end improves the user-end experience.

3.3.8.2. Google

Google also offers a cloud computing platform called "Google Cloud Platform." Like Amazon Web Services, Google also allows the user to use their preferred programming language. The platform has a wide assortment of services for compute, storage, and app development. The Google Cloud Platform's main services include the following:

- Google Compute Engine: A service for users with virtual machines for workload hosting
- Google App Engine: This service offers software developers access to Google's scalable hosting. Furthermore, the developers can use an SDK to make products that run on App Engine.
- Google Cloud Storage: Designed for storing large, unstructured datasets. There are also database storage options.
- Google Container Engine: A management system for Docker containers. Google Cloud offers a free tier platform so the users can access Google Cloud for free on a trial basis. The trial has two parts: the first is a 12-month free trial with a $300 credit, and the second is Always Free. In the first part of the trial, the user has a $300 credit that they may spend on Google Cloud for 12-months.

During the first part of the trial, the user has access to the tools from the Always Free trial, as well as all the Google Cloud Platform products, with some limitations. There are limitations on the Google Compute Engine where the user can not have more than eight cores or eight Virtual CPUs running simultaneously. There are also some restricted actions on the free trial, such as not being able to use Google Cloud Platform services to engage in mining cryptocurrency. The trial ends when

the user has spent the $300 credit or if the 12-month period has ended, whichever is met sooner.

Once the trial is finished, the account will be put on hold, and the data and resources that were made during the trial will remain available for an additional 30 days. After the 30-day period, the data and resources that were made will be removed, even if the user upgrades later on. The user has a billing account that is similar to a payment profile, which includes the payment method. Users can create more than one billing account, and the billing accounts are attached to projects; this allows the user to pick which accounts the projects get charged to.

The Google Cloud Platform has several tools to help the user estimate the cost, including:

- Pricing calculator: Used to estimate the user's monthly charges
- Product Pricing: Contains details about the cost of using Google Cloud Platform Products
- Cost-comparison calculator: allows the user to calculate the potential savings between other web services, such as Amazon Web Services and Google Cloud

For the Always Free trial, the user must not have a custom contract or rate card with Google, have an upgraded billing account, and the user must be in good standing. An upgraded billing account is a billing account that receives automatic charges for any usage beyond any credit to the user's account. Always Free allows the user to try participating products for free up to their non-expiring usage limits. These participating products are Google Compute Engine, Google Cloud Storage, and Google Cloud Pub/Sub.

Google App Engine allows users to build scalable web and mobile backends in any of their language of their choosing on the Google infrastructure. App Engine uses an open cloud platform to help the user build state of the art web and mobile applications. This setup allows the user to bring their own frameworks and libraries. Because Google App Engine abstracts away the infrastructure the user can focus primarily on coding.

Some of the primary languages that Google App Engine supports is Node.js, Java, and C#. Programmers that are accustomed to these languages will be instantly productive in the acclimate environment. An alternative to proprietary tools Google App Engine allows users to use their own Docker image or custom software stack. Users can also deploy their application to a container-based system such as Kubernetes or in the cloud, whether it be public or private.

As stated earlier Google App Engine will manage all infrastructure concerns for the user such as scaling their app, handling up or down traffic, load balancing, and applying needed updates to the underlying OS. This is Google’s attempt to help the user focus on their code and creating their web and mobile app. However, if

users can still control their infrastructure as needed. Users can SSH straight into their instances to deploy their custom code, or manage containers. Users can also integrate the Google App Engine into their DevOps process. If the user identifies their app's CPU and memory requirements Google App Engine will routinely provision the infrastructure on the user's behalf.

Google Cloud Storage is a scalable object storage service that is able to deal with all different types of unstructured data. The user is provided with robust, accessible, safe and cost-efficient object storage that is able to deal with several types of workloads. Users can use three different storage choices within one API. Each of these options has been optimized for different use cases for example batch data analytics or backup and disaster recovery.

Cloud SQL is a fully-managed database service hosted on Google Cloud Platform. It is simple for the user to setup, maintain, manage, and administer their MySQL and PostgreSQL databases in the cloud. Cloud SQL gives the user high performance, scalability, and convenience. If the user doesn't wish to use a SQL database, they are able to use Cloud Bigtable. Cloud Bigtable is an immensely scalable NoSQL database that is ideal for low-latency and high-throughput workloads. It integrates with common Big Data tools such as Hadoop and Spark with ease and it also supports HBase API. Cloud Bigtable is an excellent choice for operational and analytical application.

Google Cloud Platform also offers Persistent Disk. It is a high-performance block storage service that is perfect for both Virtual Machines and container storage. The user only pays for the storage that they use and never pay for provisioned IOPS. Persistent Disk also has multi-reader mounts and on-demand volume resizing. This is meant to simplify operations for the user.

Google designs and builds their data centers ensuring the incorporation of multiple layers of physical security protections. Only a small number of Google employees have access to these data centers and they use security measures such as biometric identification, cameras, and laser-based intrusion detection systems. Google also hosts servers at third-party data centers however they make certain that there are Google-controlled physical security measures in place on top of the layers already in place at the data center. Both the server boards and the networking equipment ant the Google data centers are custom-designed by Google and they include a special hardware security chip. With the chip Google can identify and validate genuine Google devices at the hardware level. At the hardware level Google also has secure boot stack and machine identity.

Services are deployed securely through Google's infrastructure because their infrastructure doesn't assume any trust among services running on the infrastructure. The first way it does this is by using cryptographic authentication and authorization at the application layer when dealing with inter-service communication. To prevent IP spoofing Google doesn't rely on internal network segmentation or firewalling as main security methods instead they use ingress and egress filtering at several locations in their network.

All services are provided with unique cryptographic credentials that it is able to use to prove its identity when sending or receiving remote procedure calls to other services. Furthermore, the infrastructure provides users with access management features that allow them to state which other services their service can communicate with. For access management of end user data the infrastructure provides a central user identity service. This service issues the user's client device a user credential or "end user permission ticket". After the user credential has been issued every request from the device into Google requires the device to present that user credential.

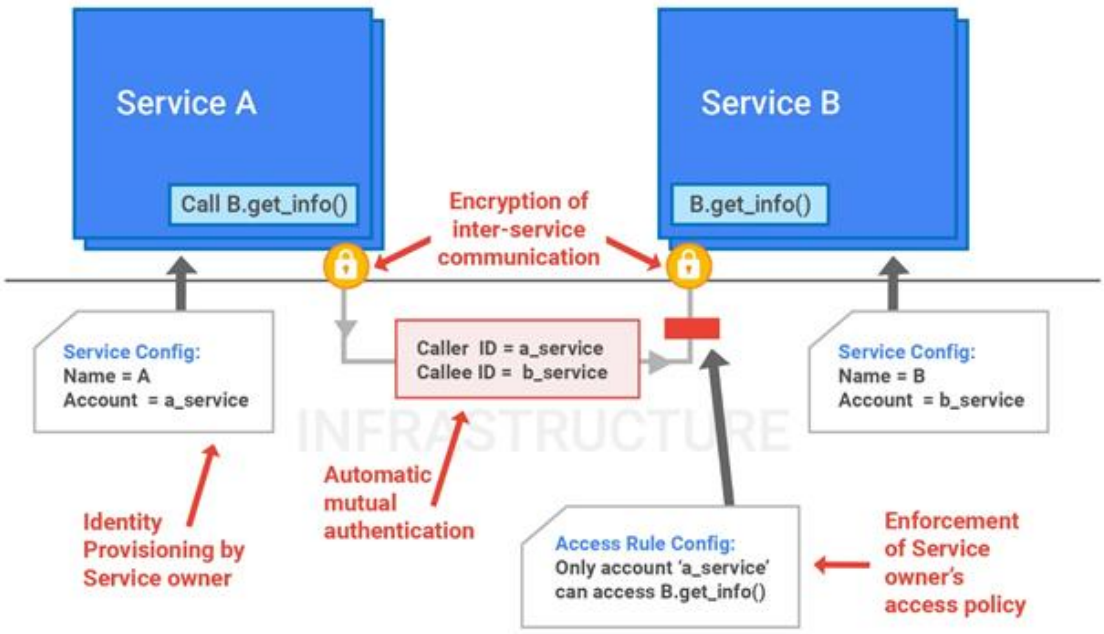


Figure 24: Example of Google Security Services

Google's infrastructure provides different storage services to the user such as BigTable and it is important that these services remain secure. Storage services such as BigTable can be configured by the user to use keys from the central key management service to encrypt data before it is written into physical storage. The key management service does the following:

- Supports automatic key rotation
- Provides extensive audit logs
- Integrates with the end user permission tickets

Google also has hardware encryption support in their hard drives and SSDs enabled. When deleting data Google first marks the item as "scheduled for deletion" in order to recover anything that was deleted by mistake. Then if it was

not a mistake Google will delete the data in accordance to the service-specific policies.

3.3.8.3. Choice

Although Google and Amazon offer similar features like NoSQL databases, file storage, extensive data logging, security configuration and dynamic resource allocation. Upon reviewing the documentation sites, it seems the API references and documentation for AWS is easier to comprehend and follow with their vast amount of tutorials. Furthermore, AWS provides lambda functions that allow developers to write programs that can be run on separate computers instead of using resources on hosting server. AWS also provides a student account that will cover a majority of the expenses. As a result, we believe Amazon's AWS is the better option for our group for hosting our application.

4. Related Standards and Design Constraints

Section 4 discusses the related standards and design constraints applicable to our SmartKart shopping system.

4.1. Standards

This section includes a discussion on different standards our team considered while designing our system. These include PCI DSS, Wi-Fi, battery and programming standards, as well as version control.

4.1.1. PCI (Process Credit Card Info)

Any software application involved in processing, or storing credit card data it must comply the Payment Card Industry Data Security Standards (PCI DSS). It is crucial that any cardholder data is secure or else if a hacker can dump accounts credit card information users can lose trust in the company and possible cause the company to close. With the Primary Account Number (PAN) and any other authentication data a hacker can steal a cardholder's identity. If a system must store cardholder it should never store the Customer Identification Number (CIN), a unique number for each credit card, or anything on the back of the credit card.

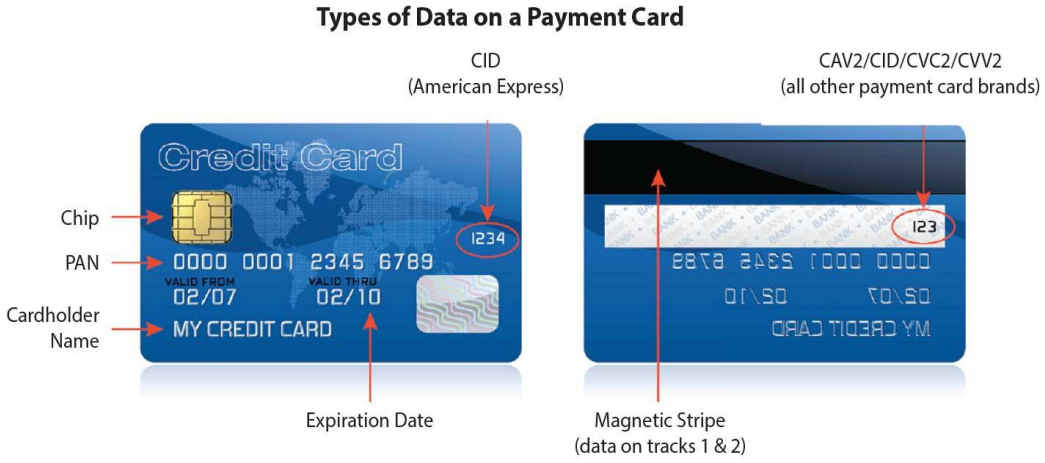


Figure 25. Payment Card Data

The standard is to not store any cardholder data on your system, protect private networks, protect data storage and transmission. Using Stripe, a PCI Service Provider Level 1 certified organization, allows us to build an application that is PCI-compliant by serving any page that handles payments with the HTTPS protocol, using Transport Layer Security (TLS) for encryption. TLS reduces the risk of customers being the victim of man-in-the-middle attacks by performing peer-2-peer encryption and verification between the client and server. To setup TLS one must acquire a digital certificate from a certification authority which says that the site is he actual site.

4.1.2. Wi-Fi Standards

IEEE 802 is a wireless standard, comprised of several subparts, including IEEE 802.3 Ethernet, IEEE 802.11 Wi-Fi, and IEEE 802.15 Bluetooth. These standards cover specifications for the physical layer and data-link features of networking for different technology devices. This section will focus on the 802.11 family of standards.

4.1.2.1. IEEE 802.11

The IEEE 802.11 standard was formed in 1997; this marked the beginning of the first WLAN standard. This standard supports a maximum bandwidth of 2 Mbps; this is too slow for a majority of applications, and thus, 802.11 products have ceased manufacturing. Since this standard, several amendments have been made to 802.11, including 802.11a, b, g, n, and ac. Any product or device that implements a standard from the 802.11 family is required to pass a set of tests in order to be considered “Wi-Fi certified.”

4.1.2.2. IEEE 802.11b

The original IEEE 802.11 standard was expanded in 1999, with the 802.11b edition. This specification supports a maximum bandwidth of 11 Mbps. It utilizes the same 2.4 GHz radio signaling frequency that the original standard does. This specification may run into interference with other devices that utilize the unregulated 2.4 GHz range, such as cordless phones and microwave ovens. However, this issue may be avoided by keeping these appliances a suitable distance away from one another.

The advantages of standard 802.11b is that it is low cost, and the signal range is not easily congested. The disadvantages of IEEE 802.11b is that it is slow, and its unregulated 2.4 GHz range may be interfered with.

4.1.2.3. IEEE 802.11a

Another extension of the 802.11 standard was created; IEEE 802.11a. This specification is typically used in business networks, as opposed to 802.11b which is traditionally used in home networks. IEEE 802.11a supports a maximum bandwidth of 54 Mbps, and it utilizes a 5 GHz radio signaling frequency. This frequency causes signals of the 802.11a speciation to have difficulty penetrating obstructions, such as walls.

The advantages of IEEE 802.11a include that it is fast, and it has regulated frequencies; this prevents interference issues from other applications. The disadvantages of this specification are that it is high cost, and it has a short range signal that is easily congested.

4.1.2.4. IEEE 802.11g

Another standard implemented was IEEE 802.11g, which was released in 2003. IEEE 802.11g was created with the idea of combining the advantages of 802.11a and 802.11b into a single standard.

IEEE 802.11g supports a maximum bandwidth of 54 Mbps, and it utilizes a 2.4 GHz radio signaling frequency. This specification is compatible with IEEE 802.11b, and as such, the access points of both standards are compatible with the wireless network adapters of each other.

The advantages of IEEE 802.11g is that it is fast, and has a signal rage that is not easily congested. The disadvantages of 802.11g is that it is more expensive than 802.11b, and that its unregulated 2.4 GHz signal frequency range may be interfered with.

4.1.2.5. IEEE 802.11n

The IEEE 802.11n specification was created in 2009 to improve the 802.11g standard. It did this by increasing the amount of bandwidth support by using MIMO technology.

Multiple Input, Multiple Output, or MIMO, is a type of technology that utilizes antennas for wireless communication. The antennas are found at both the source and destination, or the transmitter and receiver, and are used to optimize data transmission rates and minimize errors.

The 802.11n standard supports a network bandwidth of up to 300 Mbps, and offers an improved range in comparison to earlier standards. This standard is also compatible with 802.11b and 802.11g devices.

The advantages of IEEE 802.11n is that it has the quickest speed and the most optimizes signal range. It is also better suited for protecting against signal interferences from other devices. The disadvantages of this specification is that it has not yet been finalized, and it is expensive, more so than IEEE 802.11g. This specification may also produce signal interference with other 802.11b and 802.11g networks nearby.

4.1.2.6. IEEE 802.11ac

The IEEE 802.11ac specification is the most recent edition of the original 802.11 standard. This standard can support connections for both the 2.4 GHz and 5 GHz signal frequency ranges through use of dual-band wireless routers. IEEE 802.11ac supports a maximum bandwidth of 450 Mbps for the 2.4 GHz range, and up to

1300 Mbps for the 5 Ghz range. The IEEE 802.11ac standard is also compatible with the 802.11b, 802.11g, and 802.11n editions.

4.1.2.7. Comparison

The following table lists the comparisons between the 802.11 family, including release date, band, bandwidth, and maximum data rates.

Table 7: IEEE 802.11 Standards Comparison Chart

Standard	Release Date	Band	Bandwidth	Maximum data rate
802.11	1997	2.4 GHz	20 MHz	2 Mbps
802.11b	1999	2.4 GHz	20 MHz	11 Mbps
802.11a	1999	5 GHz	20 MHz	54 Mbps
802.11g	2003	2.4 GHz	20 MHz	54 Mbps
802.11n	2009	2.4 GHz, 5 GHz	20,40 MHz	600 Mbps
802.11ac	2013	2.4 GHz, 5 GHz	40,80,160 MHz	6.93 Gbps

4.1.3. Security Standards

As seen in the news, security breaches in a company's system can wreak havoc on their business, in some cases forcing them to close. The Open Web Application Security Project (OWASP) organization releases a document covering the top 10 web application security risks, the most common attack vectors are Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), SQL/NoSQL Injections, security misconfiguration, and expose sensitive data.

A Cross Site Scripting occurs when malicious users inject malicious code into trusted sites via user input and the system doesn't validate or escape the data received. Whenever a user visits the website their browser will execute any script it receives from the server, allowing the malicious script to have access to cookies, session tokens, and other sensitive information the browser has access to, and manipulate pages being rendered. OWASP suggests web sites should never insert untrusted data to the DOM, if we must escape the untrusted data before

inserting it to the DOM, and escape URL before inserting untrusted data into URL parameter values.

Cross Site Request Forgery is an attack that tries to get the victim user to execute unwanted actions on a web application that they are currently logged into so it can inherit the identity of the victim. Although this take isn't designed to steel user information it can cause victims with normal accounts to perform undesired requests like transfer funds change account setting, but if the victim has an administrative account the hacker can compromise the system. General CSRF defense involves verifying source origin and target origin in headers and use CSRF tokens that's value change whenever a state changing operation is made.

SQL Injections occur when data from the user isn't not escaped or washed of special characters that can modify the query statements. The goal is to perform CRUD operations on sensitive information in the database that they don't have authorization to. For instance, if we had a proc that returned all account information for a given account when we prove the `account_id` a hacker could send a string for `account_id` that returns more than their account information.

```
SELECT * FROM accounts WHERE id = account_id;
```

If the hacker sets account_id to "1' or '1'='1" then the query becomes

```
SELECT * FROM accounts WHERE id = '1' or '1'='1';
```

Then regardless if there isn't any account id equal to 1 since the where clause will always return true due to the '1'='1', allowing a hacker to get all information in the accounts table. With simple tweaks to the string that allowed us to dump everything in the account table hackers can append additional SQL statements to execute.

Another one of OWASP's top 10 security risks is security misconfiguration. Although a system securely validates and transmits data, it means nothing if developers didn't properly implement standard security policies on their servers. Common scenarios include:

- Remove guest accounts and not changing default admin passwords
- Not updating firmware and software
- Not disabling directory listing
- Displaying stack traces of error that occurs to the end users
- Disabling unnecessary ports
- Setup default drop rule on firewalls

- Properly log all actions on network and whenever throw an exception on servers

4.1.4. Battery Standards

IEEE Std 1187-2002 is a recommended standard for valve-regulated lead-acid (VRLA) batteries. It provides some insight into proper design practices, and procedures for storing, mounting, venting, and assembling VLRA batteries. It also discusses some safety practices, and has useful information for float-service stationary applications.

The standard first covers general safety. It states that VLRA batteries can pose a hazard to users, as any battery can. In order to properly handle and install VRLA batteries, certain precautions must be made known. Any work associated with VRLA batteries shall only be handled by those with that are knowledgeable of VRLA batteries, have proper training, tools, and protective gear. Before any task involving VRLA batteries is initiated, hazard analysis shall be conducted. It is recommended that the potential hazards associated with the task be made known and discussed with everyone, and should include the importance of protective gear. Significant hazards listed are an overheated battery, electric shock, electrical arc, and toxic chemical leaks, and improper carriage and handling of a VRLA battery. The associated VLRA battery's datasheet shall be perused by those working with the battery. The datasheet should have relevant information on hazards and first aid. Analysis for arc flash prevention shall be done during the beginning phases of design and installation. Any information that is relevant to any potential risks needs to be documented and made available with the battery's installation.

The document goes on to discuss electrical hazards. It is recommended that proper practices be performed when connecting electrical components to batteries, to prevent shorting of the battery. Shorting a battery can be dangerous because it generates a spike in the current and arc flash. Systems that can be potential shock hazards produce an overall voltage greater than 50V. Any system below 50V is not considered to be a substantial shock hazard. IEC/TR 60479-5 and NFPA 70E define the touch voltage threshold at 100V. Shock-resistant gloves should be made available to anyone working with grounded batteries that produce 50V to 100V. For batteries greater than 100V, insulated gloves need to be worn that are able to withstand over 100V.

Arc flash hazards are defined as having a DC voltage greater than 100V. For batteries under 600V, arcs that exceed a couple inched cannot be sustained. For batteries over 100V, the risk of arc flash is greatly reduced if potentials are spread out at least one foot. The chemicals inside a VRLA battery can pose a hazard, due to the electrolyte containing sulfuric acid. Damaged VRLA batteries can sometimes leak the electrolytes, which poses a threat. Although spill containment is not

required, it is recommended. A neutralizing agent for the electrolyte can generally be provided by the manufacturer. Eye protection is required when handling batteries to avoid electrolyte contact with eyes. If an arc flash is likely to occur with a battery, gloves should be provided that aid against thermal hazards.

The location in which a battery is installed should be considered. There should be enough space for the battery and other components to reside safely. The area in which the battery resides should be well-kept, dry, and properly ventilated. Additionally, the battery should avoid direct contact with sunlight to prevent overheating and case material degradation. Battery installations should be done so that they can be maintained in the future. Ventilation is another important consideration to take note of for a VRLA battery. VRLA batteries typically display optimal performance when operating between temperatures of 20°C to 25 °C. VLRA batteries operating in steady-state conditions produce a temperature that is higher than the surrounding temperature. When a VRLA battery is properly ventilated, this temperature rise is no longer an issue. Also, placing a VRLA battery in a temperature lower than 20°C decreases performance and even sizing. Overall, lower temperatures cause decreased battery capacity, while higher temperatures cause a short life cycle and overheating.

4.1.4.1. Design Impact of Battery Standards

IEEE Standard 1187-2002 has many great recommendations that our team will be sure to follow when we are developing, prototyping, and testing our system. To power the components of the shopping cart, a 12V rechargeable valve-regulated lead-acid battery will be used. Since the battery that powers the shopping cart is a VRLA battery, the recommendations in this standard can be taken and applied to the project.

A significant recommendation that was mentioned was considering the operating temperature of the battery. Shopping carts spend a good amount of time outside in parking lots where rising temperatures can occur, especially during the day. Additionally, the sunlight shines on objects and heats up their surfaces. These two factors are should be sought to be eliminated, as they damage the battery and cause it to exhibit less than optimal performance. To prevent this, it will be a priority to place the have the battery reside in an enclosed area where sunlight cannot reach the battery. Additionally, the enclosed area will be well-ventilated so that the steady-state temperature of the battery will not produce negative effects. The enclosure for the battery will also prevent the any leaked electrolyte solution from the battery to escape outside of the enclosure. Although this is not a requirement in the standard, it is a safe practice that should be followed in this scenario, since it could potentially protect a user of the shopping cart of toxic chemicals.

4.1.5. Programming Languages

This section provides an in-depth discussion on standards and common practices regarding programming languages implemented in our system. The C language will be used for embedded programming, while Node.js will be used for the backend development.

4.1.5.1. C Programming Language

The standard naming practice for coding in C is to have descriptive variable names. If the variable represents a unit or measurement, such as weight or distance, it is good practice to include the units that are being measured. For instance, if a variable for a program is representing the distance of a path in kilometers, it might be called distance_km. The same goes for naming functions; the name of the function should match the purpose of the function.

As for the placement of braces, it is standard to put the starting brace on the same line as the conditional or loop statement, and to put the closing brace on its own separate line, as shown below.

```
1  if (condition) {
2      ...
3  }
4  while (condition) {
5      ...
6  }
```

Figure 26. Code snippet 1

Although if statements and loop statements don't need braces if the code is a single line, it is always a good practice to utilize them anyway, in case an extra line of code is added to the statement at a later time. This also provides a more consistent look and promotes unambiguity. Furthermore, after the closing brace, it is good practice to add a comment at the end of the statement to leave a reminder of the functionality of that statement.

For parentheses, the following common practices are used:

- Parentheses should not be placed next to keywords; instead separate them with a space.
- Place parentheses next to function names
- Do not place parentheses in return statements unless it is absolutely needed.
  An example of these practices can be seen below:

```
if (condition) {
...
}

while (condition) {
...
}

create_maze();

return 0;
```

Figure 27. Code snippet 2

An additional standard when coding in C is that a line should not exceed 78 characters. This is because even though large monitors can stretch the windows so that they can view more than 78 characters per line, not all systems have the same window size, which means that not all readers will be able to see all characters. For readers with smaller window sizes, they may be required to utilize a horizontal scroll bar, which is not very programmer friendly. Also, having only 78 characters per line allows all the code to still be seen even when multiple windows are open.

Another standard when coding in C is that there should only be one statement per line, with the caveat being that it makes the code easier to read and understand. The only exception to this is if the statements are closely related, as demonstrated with the if and else if statement below:

```
if (condition) {
...
} else if (condition) {
...
}
```

Figure 28. Code snippet 3

Similarly, variable declarations should only contain one variable per line. This is because documentation can be added later for the variable on that line. This also makes the declarations clear, so that the programmer does not accidentally declare a pointer when it should have been simply a char. It also allows the reader to clearly view which variables have been initialized.

```
//Don't do
int count, i;
```

```
//Do
int count = 0;
int i = 1;
```

Figures 29 & 30. Code snippets 4 & 5

4.1.5.2. Node.js

In Node.js, it is standard practice to avoid the "this" and "new" when coding. This is because Node.js has a lot of higher-level functions, and involves passing many callbacks. By avoiding using "this" and "new", it makes the functions easy to move around without the need to bind to a certain context. Overall, closures are preferred over object methods, and explicit arguments are preferred over closures.

```
1  //Don't use
2  function Website(url) {
3      this.url = url;
4  }
5
6  //Instead use
7  function Website(url) {
8      return { info: async.apply(http.get,url + '/info')};
9  }
```

Figure 31. Code snippet 6

Another common practice in Node.js is to break the code up into small functions. Breaking up the code into smaller parts makes it easier to recombine them in new ways later on, and helps when doing callbacks in the code.

```
1  //Don't do
2  function convertJsonToCsv(filename, target, callback) {
3      readFile(filename, function (err, content) {
4          if (err) {
5              return callback(err);
6          }
7          parseJson(content, function (err, data) {
8              if (err) {
9                  return callback(err);
10             }
11             convertToCsv(data, function (err, csv) {
12                 if (err) {
13                     return callback(err);
14                 }
15                 writeFile(target, csv, callback);
16             });
17         });
18     });
19 }
```

Figure 32. Code snippet 7

```
//Instead do
function convertJsonToCsv(filename, target, callback) {
    readJsonFile(filename, function (err, data) {
        if (err) {
            return callback(err);
        }
        writeCsvFile(target, data, callback);
    });
}

function readJsonFile(filename, callback) {
    readFile(filename, function (err, content) {
        if (err) {
            return callback(err);
        }
        parseJson(content, callback);
    });
}

function writeCsvFile(target, data, callback) {
    convertToCsv(data, function (err, csv) {
        if (err) {
            return callback(err);
        }
        writeFile(target, csv, callback);
    });
}
```

Figure 33. Code snippet 8

In Node.js, a function should be either always synchronous or always asynchronous. If a function is both synchronous and asynchronous, then it can be nearly impossible to predict the codes behavior. When the code below is utilized, for example, it can be quite easy to miss the cached case causing the code to near complete.

```
1  //Don't do
2  var CACHE = {};
3
4  function getRecord(id, callback) {
5      if (CACHE[id]) {
6          return CACHE[id];
7      }
8      http.get('http://foo/' + id, callback);
9  }
10
11 //Instead do
12 var CACHE = {};
13
14 function getRecord(id, callback) {
15     if (CACHE[id]) {
16         return process.nextTick(function () {
17             return callback(null, CACHE[id]);
18         });
19     }
20     http.get('http://foo/' + id, callback);
21 }
```

Figure 34. Code snippet 9

It is also standard practice to always check for errors in callbacks, as not doing so can cause a long and tedious process. Often times the program will continue executing, but will fail later on causing a different error message to be displayed. Another standard is to return on callbacks. Callbacks tend to be the last thing in the code, similar to a return in C. However, a callback will not halt a function after it is reached. Therefore, in order to prevent execution from continuing after calling a callback, it is best if one returns the callback function call.

Try-catch blocks should only be used with synchronous functions. If it is attempted with asynchronous code, it will not work and it may kill the whole server process without an uncaughtException handler. Even then, the error will most likely have no meaningful context. Therefore, in asynchronous functions, always pass errors back to the callback. This practice will be fine as long as the code is checking for errors in callbacks as mentioned earlier.

4.1.6. Version Control

For version control, it is always good practice to write a descriptive commit message. This is useful because when another member on the team is examining the changes in the code, they will know the purpose of that change. It also makes it easier for a reader who is looking for changes related to a particular subject.

Another standard for version control is to have each commit have a single purpose. Once the purpose is decided, then the commit should be executed solely for that purpose. Doing this makes locating changes relating to a certain subject easier,

and it also simplifies the process of determining which changes are responsible for bugs. If one commit contains code that is for multiple purposes, or is code for a certain purpose, is spread across multiple different commits, then the version control history becomes compromised.

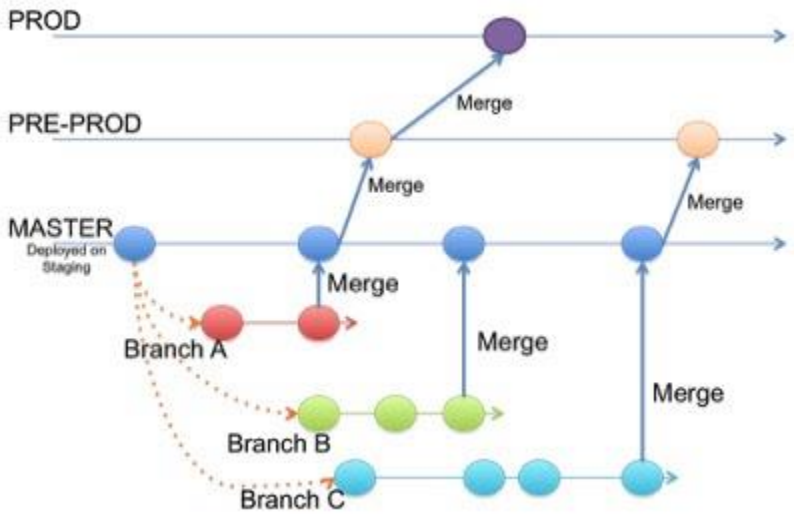


Figure 35. Version Control

Although commits are important, it is also important to avoid indiscriminate commits. When committing, it is good practice to only commit the specific files. This avoids making unintended changes to files and having a single commit with more than one purpose.

It is also standard to always have the most up-to-date version of the code. Since multiple people are working on the same files, that means that the programmers on the team should update their files often. This avoids the problem of having to resolve conflicts with differing versions of code. This tends to save a lot of time and aggravation.

Another common practice that aims at avoiding conflicts is sharing the changes that have been made in the code frequently. That means when a programmer on the team finishes their section of work, the code should be shared to the other members of their team as soon as possible. However, the code that is shared should still be checked to see that it doesn't have any bugs and doesn't produce any unexpected output.

4.2. Constraints

Our SmartKart shopping system includes several constraints in regards to the hardware and software component design. This section will discuss economic, size, environmental, safety, security, and ethical constraints considered by our team.

4.2.1. Economic

The cost for designing and producing our SmartKart shopping system is a primary constraint for our team. As this system is not being sponsored by any corporations, it must be funded by the group members themselves. This constraint inhibits the team from trial and error through testing different components to decide which ones are most useful for the system; instead, it requires the group to perform a great deal of research in order to determine the best components to purchase and implement in the system.

Additionally, our team's goal is to not only keep costs low for prototyping, but for manufacturing the system as well. Manufacturing costs need to be low to increase the system's potential for being marketed to shopping corporations. The SmartKart shopping system needs to be as affordable as possible, as it would need to be mass produced and marketed to such stores, and it would be used by thousands of customers daily. Ideally, our team would be able to purchase the highest quality components for building this product, however this may be inhibited in order to keep a balance between price and quality.

4.2.2. Size/Portability

While determining the kind of shopping cart to use with this project, a few things had to be considered. The shopping cart had to be modern-looking, and have some appeal to a potential customer that might one day the cart. The shopping cart also had to be relatively small and weigh a reasonable amount. This way, the shopping cart would not be too difficult to push and carry. The dimensions of the chosen shopping cart are 41.75 inches by 24 inches by 21 inches.

The microcontroller, battery, RFID sensor, Wifi module, voltage regulator, GPS module, and all associated PCBs and wiring would go fit neatly inside the smaller basket that exists on the shopping cart. These components are relatively small and would have excess space in the smaller basket. The largest component would be the battery, which is 3.5 inches by 2.8 inches by 4.1 inches. Additionally, the electromagnetic locking devices would reside near the wheels of the shopping cart.

4.2.3. Environmental

As the producer of SmartKart, we are responsible for creating a product that is environmentally friendly. The SmartKart system is not going to negatively affect the environment, as the original design calls for the power to come from a battery that is built into the system. Although it is not a primary goal, if time allows it, research will be done to see if an alternative energy source could power the battery, such as solar power to fuel the system. If it is found that such alternative energy sources could power the system without any drawbacks, then it will be implemented into the system.

4.2.4. Safety

The safety of the end users is of utmost concern to our team regarding the development of the SmartKart shopping system. The system's main purpose is to create a friendlier shopping experience for the users, which should not impede on their health or safety whatsoever. Our teams main focus of concern is that the user's personal information is protected at all costs. With our SmartKart shopping system, the users are required to input their personal data, including their credit card information, and as a result, our team has taken extra measures to ensure the safety of this data. This was accomplished by using a payment gateway, which allows us to send the user's information to other servers that are PCI compliant in order to safely secure their data.

Another focus of concern in regards to the users' safety is that the components chosen to produce the system are safe to use in all weather conditions. More specifically, our battery was chosen such that it may withstand extreme temperatures without issue. Originally a Lithium battery was considered for our power source, but upon further research, it was clear that this option was not reliably safe, and thus our team bypassed this option for an alternative, safer power source.

4.2.5. Security & Scalability

Security constraints would prevent developers from incorporating features without securing/encrypting user information being computers. While scalability constraints would prevent the system from including functionality that could hinder system performance. From a hardware perspective, the shopping cart devices must always be connected to a network to properly work. Any store that uses the devices must provide a robust secure wireless network that allows the devices to securely communicate with the system.

The devices must also have protocols to handle physical tampering so data on the device isn't divulged, but the device can still be used after robust testing. The system should be able to integrate into any retail companies' system without affecting either systems security. If the system is being used by many retail

companies, there could be a data storage and performance issues. The system would store all product and valid coupon information per store which could cause data management to become an issue requiring additional code to handle retail company's unique cases.

The system would also need to be resistant to high requests if many people are shopping or else the system could possibly go down, affecting all customers open shopping carts. There are some web service providers that dynamically allocate resources for our system if its CPU usage every exceeds a certain threshold and hosting the system over a cluster of computers across to solve performance issues that may result in system crashes.

The system must also be designed so retail stores can't access other retail companies' information and shopper's information without their authorization, this includes previous purchases, but they are only permitted to view purchases made at their company. This would require that every purchase and shopping cart device has unique id.

4.2.6. Ethical

The purpose of SmartKart is to improve the overall shopping experience of our users. Therefore, the protection of our users' personal information is of the utmost importance, and there will be safeguards to ensure that their information is protected. Furthermore, the SmartKart system will not use any toxic materials or any material that comes from an unknown origin in order to reduce the cost of the product. The features promised in SmartKart will also not be removed for purely financial reasons. In regards to patents, research will be done to ensure that no patents are infringed upon. Moreover, any protected designs or ideas will not be used in the SmartKart system without the proper credit given if applicable.

5. Project Design

This section describes, in detail, the overall design of our SmartKart shopping system. It includes the initial and current design of the system, as well as an overview of the system's software design architecture.

5.1. Hardware

This section will explain in detail the hardware components that will be used. Hardware is needed to read in store item RFIDs, sending the ID to a database, and locking the shopping cart.

5.1.1. Initial System Design

At the beginning of the semester, a document called "Divide and Conquer" was created for the Smart Cart, to go over some of the early stages of designing the project. To see and understand how the hardware components would operate with one another, an initial design block diagram was created, as can be seen in Figure X. This diagram provided a high-level overview of how the overall system would be constructed.

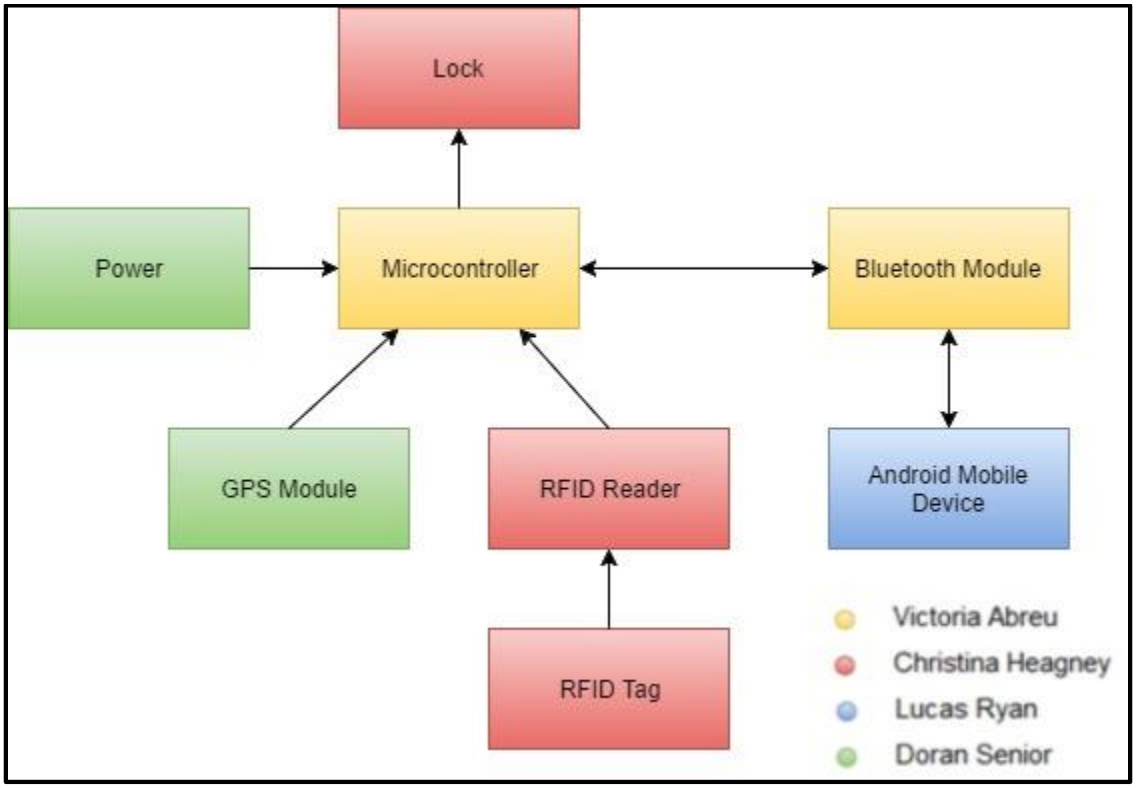


Figure 36: Initial Design Block Diagram

5.1.2. Updated System Design

Further research into the project led to some minor changes in the system design. Overall, the system still operates fundamentally in the same way.

First, the Bluetooth module would be replaced with a WiFi module. While the Bluetooth module is useful for communicating with an Android mobile device, it would potentially pose a problem on the system's software design. The Android device would not be running a regular mobile application to update the items and costs in the shopping cart. Rather, a progressive web application would be used to implement this. Progressive web applications operate in a similar fashion that a website does. A database is used to store relevant information, and insertions and queries can be performed. Because of this, the progressive web application is grabbing data from the database, and not via Bluetooth. An easy workaround to this is to replace the Bluetooth module with the WiFi module. Now instead of communicating directly with the Android device, the hardware is now inserting the RFIDs into the database. This provides a correct and efficient way for the progressive web application to pull information from the database and update itself accordingly.

Second, the actual implementation for the shopping cart's locking mechanism was widely unknown at the early stages of this project. After further research into various ways to lock a wheel, it was decided that placing electric linear solenoids in close proximity with the two rear wheels of the shopping cart would serve as the locking mechanism. When powered on, the rods inside the electric linear solenoids would be pushed outwards and would come into contact with the shopping cart wheels. The amount of friction and force applied to the wheel by the rod would prevent the wheel from rolling any further.

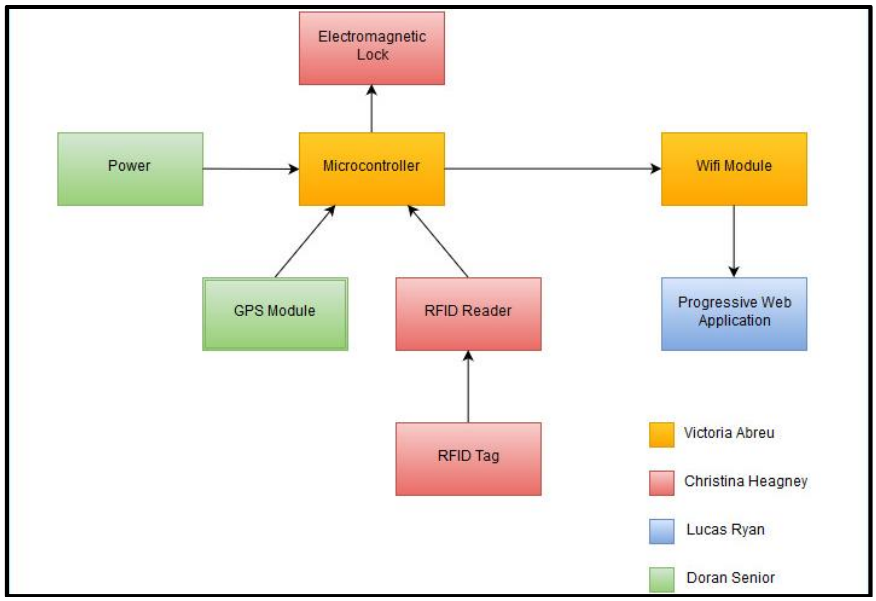


Figure 37. Hardware Block Diagram

5.1.3. Logic Diagram

The following figure is the logic diagram representation of our SmartKart shopping system.

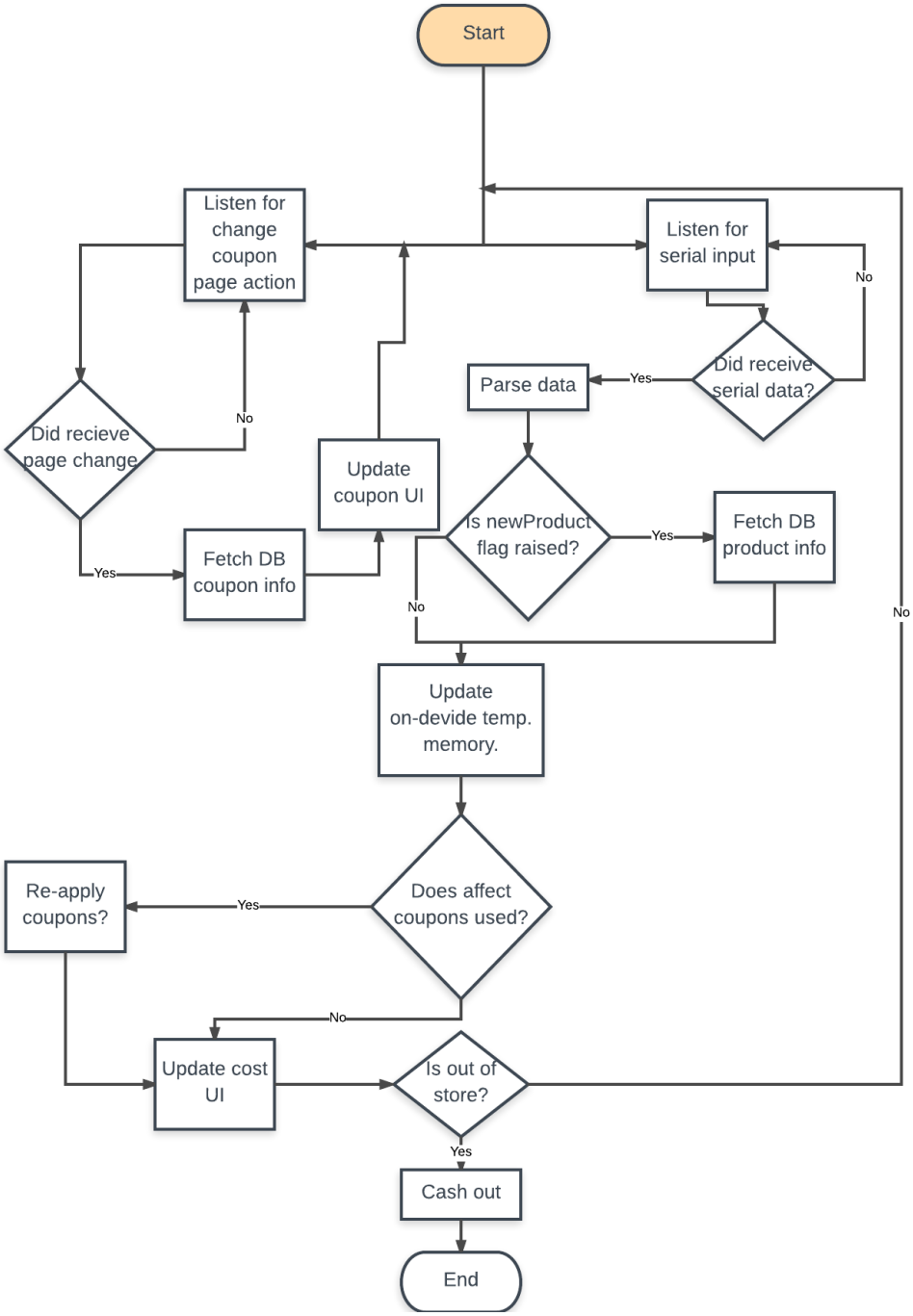


Figure 38. Software Block Diagram

5.2. Software

This section will explain in detail the user interface design, system's REST API, requests to other APIs, and architecture.

5.2.1. UML

Refer to the Appendices section 9.1 UML Diagram.

5.2.2. Action Sequence Diagrams

Users will be able to view a selected stores available coupons, connecting to a shopping cart to begin shopping, add and/or removing item from their shopping cart, making returns, and review previous purchases all from the application.

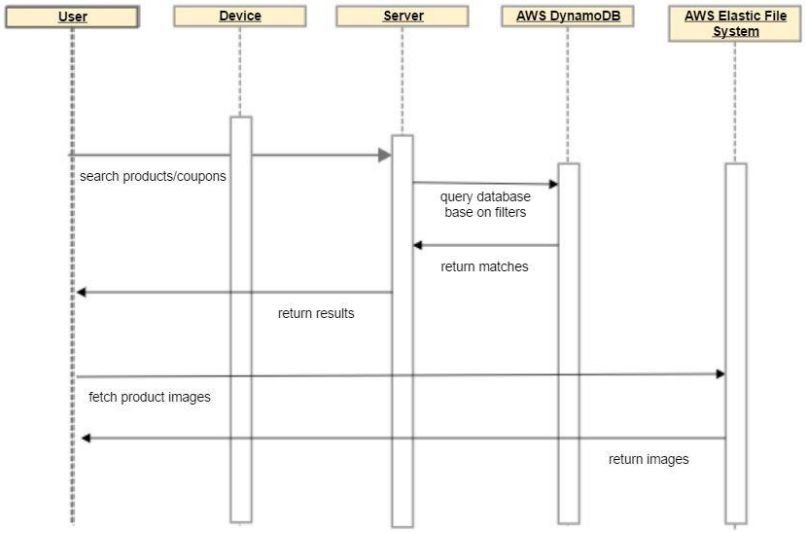


Figure 39. Review coupons & products sequence diagram

Unlike every other feature on the application, users are not required to have an account and login to see the products available and coupons offered at stores. User will first have to select a store from a list of stores to see the store items and promotions. Due to device storage constraints, the client's device will make constant requests to get the previous/next set of items/coupons. Items will display a name, picture, and cost of the it, while coupons will show the name, picture, and discount type. Users will be able to select a single item to get more information about it, like the isle it's on in the store and/or if its currently in stock.

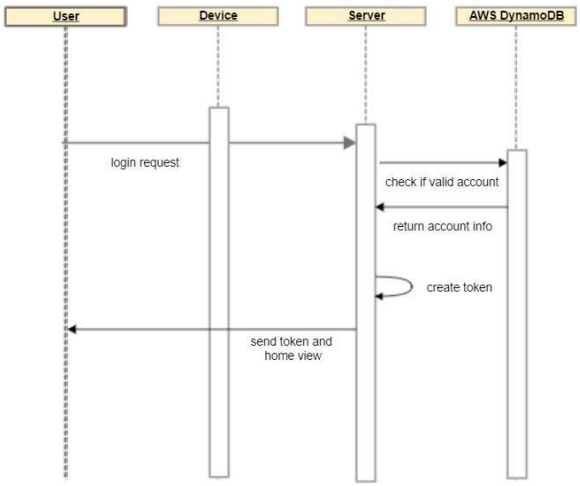


Figure 40. Login process sequence diagram

Besides searching coupons and items at a store, users will not be able to fully interface with the system. Users are required to login before they can make purchases, view previous purchases, or make returns. If a user enters credentials that are associated with an account on the system then they will be notified of an unsuccessful account login attempt. However, if the entered username is on the system, but the user fails entering in the correct password after 5 attempts the account should be locked. If the user successfully logs in then the will be directed to the home page. If the user doesn't have a verified credit card on their account then they will only be able to see previous purchases, otherwise, they can see a summary of their most recent purchase and image icon to take pictures of a shopping cart's QRcode.

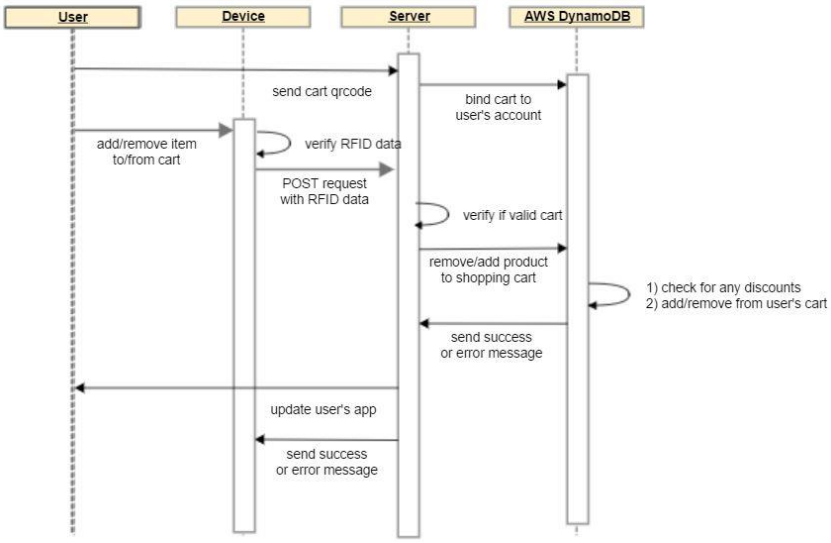


Figure 41. Connect & add/remove items from cart

Once a user is logged in they can connect to a shopping that way a user is temporarily bounded to a single shopping cart and begin shopping. The user will notify the system what cart they are using by sending a picture of the QRcode on the shopping cart which stores the information of what store the cart belongs to and the cart's unique id. As the user adds or removes item from their cart the data read from the item's RFID tag will be sent to the server to update the user's shopping cart. Every item can have a unique id stored in the tag, that way if the item is found to already be in the cart we know the user was removing it from the cart. As we add or remove the item from the cart we must reapply and discount associated with the item. Then, the user's device will receive updated data about the item.

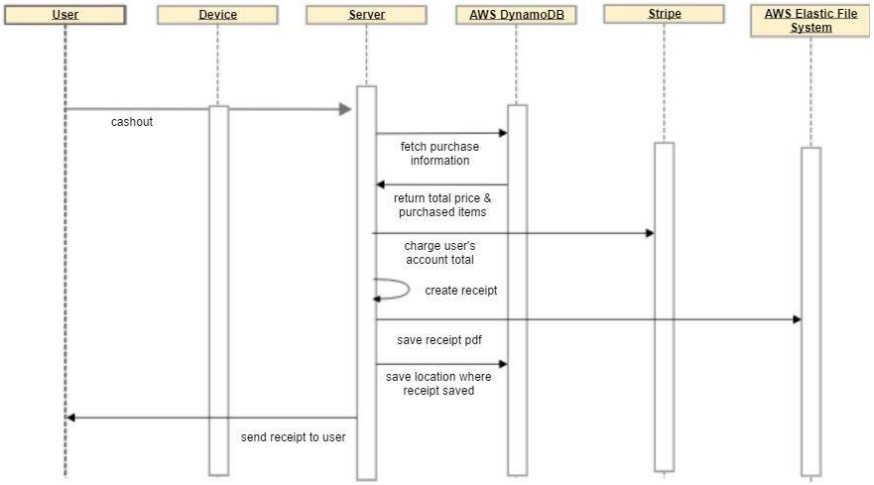


Figure 42. User cashes out sequence diagram

When the user notifies the system that they are done shopping then the server will fetch information about the items in their cart. The database will fetch all the items the user has in their cart and then check to see if any items have coupons that can be applied. Once this is complete we will calculate the transaction's total and send a request to charge the user's credit card via the Stripe API. Once the charge is successful the user will receive an electronic receipt showing everything they purchased and the discount applied. We will save the electronic receipt generated on the file system and update the transaction to show the where the receipt is stored.

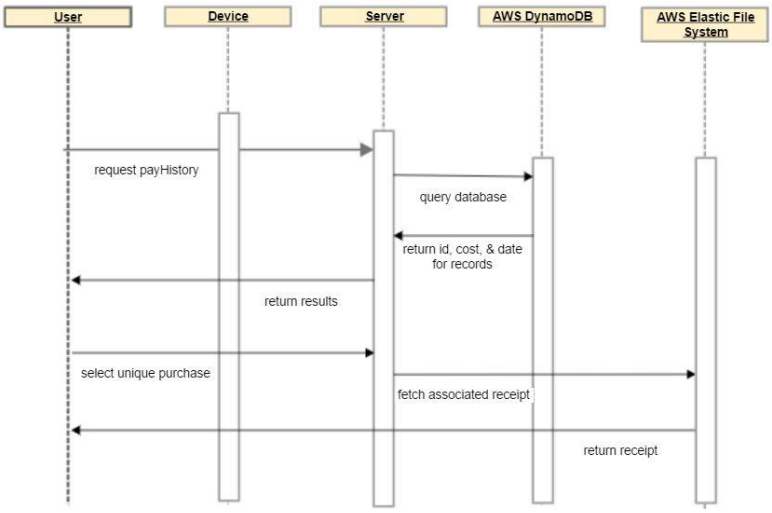


Figure 43. Review previous purchase(s) sequence diagram

Once a user is successfully logged in they can navigate to the 'payHistory' page to view a list previous purchases. Each element in the list will display the date the transaction was made, the amount, and what store this took place. If the user has more than 50 previous purchases we can implement paging instead of loading all the data at once so users don't notice device performance issues. When a user taps on a specific transaction the page will be loaded with the pdf generated when they made the purchase. The information on the pdf receipt shows the transaction id, store information, and a layout showing the items purchased and applied coupons.

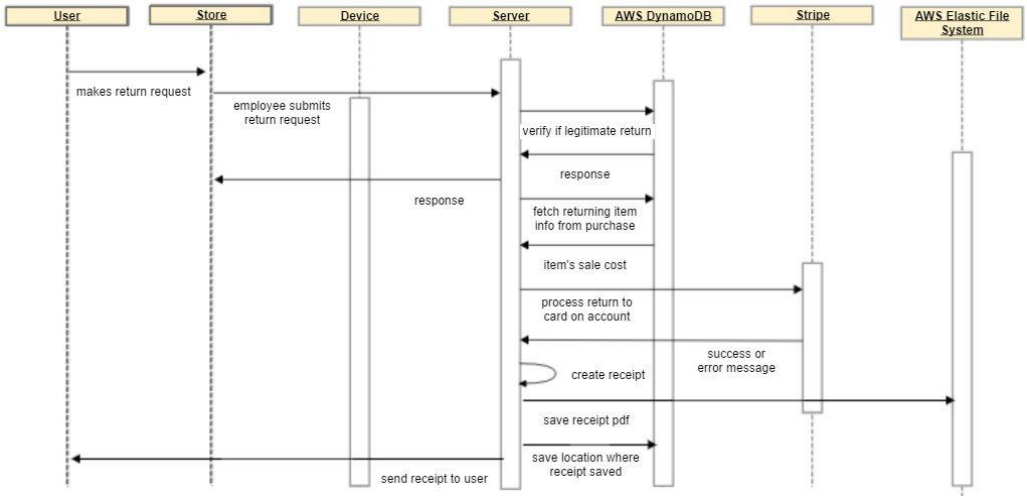


Figure 44. User attempts making return

When a user attempts to make a return to the store an employee must make the return request to the system sending the employee making the request, the item(s) being returned, and the id of the transaction these items were purchased under. The system must then verify if this return is allowed checking to see if the item is being returned in the allotted period, item was purchased under provided transaction, customer account has valid credit card, or the account making requesting a return isn't the same employee account making the return. If the system confirms this transaction as successful then the system must fetch information about the cost of the item and make a return charge to the credit card the customer has on file. Then we generate a new receipt showing the transactions performed save it and send a copy to the customer.

5.2.3. Application

We will be using Amazon's S3 for storing pdf receipts of user transactions this way only need to fetch a file instead of fetch records from database and compute transactions again saving time and resources. DynamoDB will be storing all information about customer accounts, store coupons and products, and previous transactions. For payment processing or user payment validation, Stripe will perform charges, returns, and checking if a user's card is valid. Once the user's card has been charged we use SendGrid to email the customer a receipt of their recent purchase.

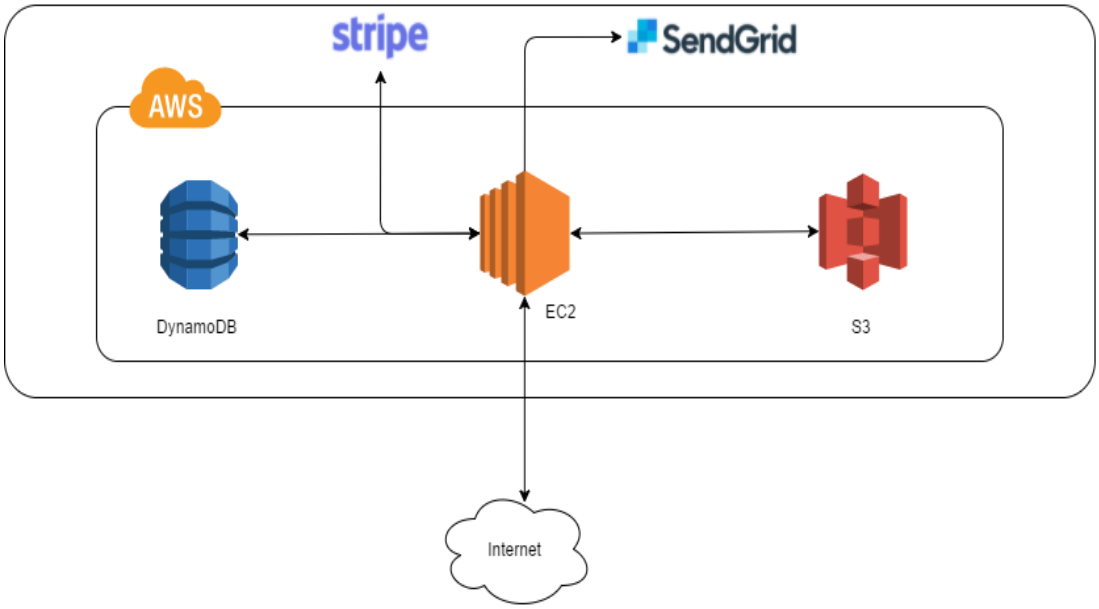


Figure 45. Architecture Diagram

5.2.3.1. UI/UX

The following section illustrates mockups of the application our team is designing for our SmartKart shopping system. Once logged in, the user will be able to access all pages on the app from the navigation icon, as shown in Figure 48.

a. Login

From this page, the user can login, reset their password, or create a new account. If the user clicks the register button they will be directed to the register page. However, if the user attempts to login it will either redirect them to the home page or display a message on the login screen alerting them of an unsuccessful login attempt.

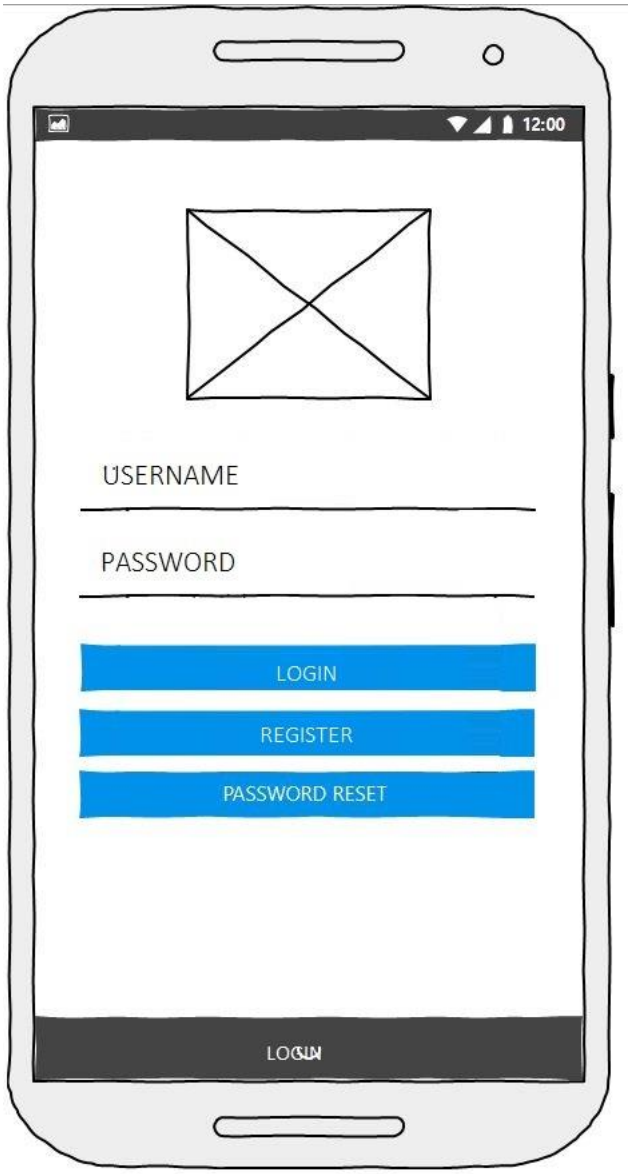


Figure 46. Login Page

b. Signup

Here the user will enter their email, password, desired username, and card information. The user will not be required to enter in their card information, however, this will prevent them from making purchases from the app.

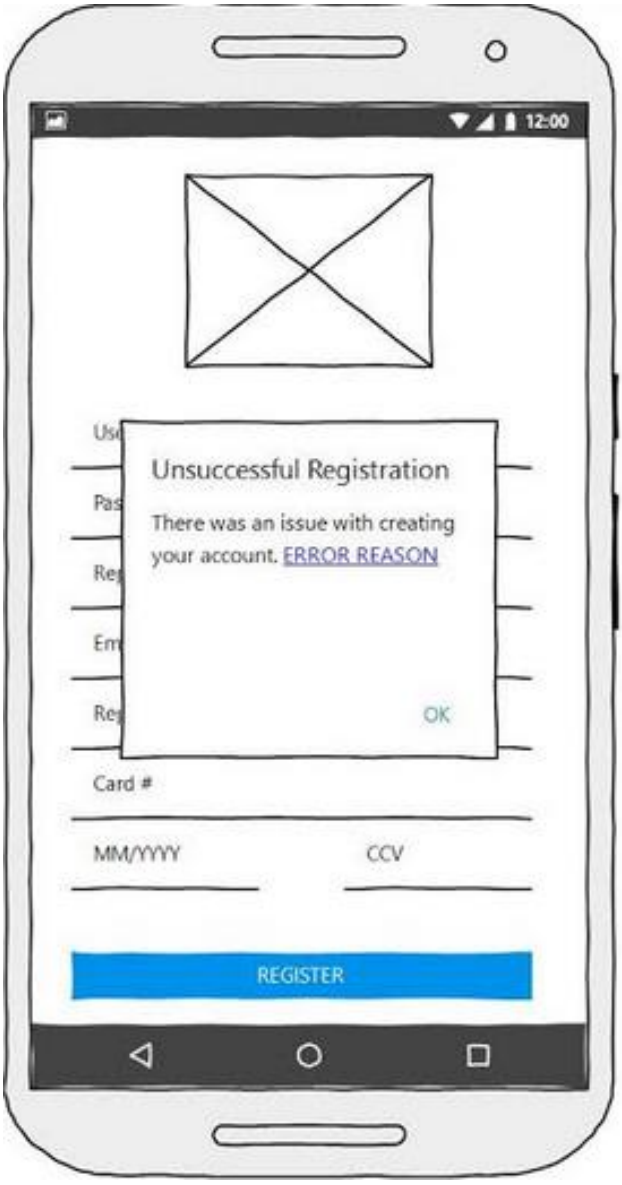


Figure 47. Account Registration Page

c. Profile

The user will be able to view their account information, except their card information. Everything displayed on the Profile page screen will be input fields with the default value of their current account information. Users can edit the information and click update to save the changes to their account, however, if they click close account their account will be marked as deleted.

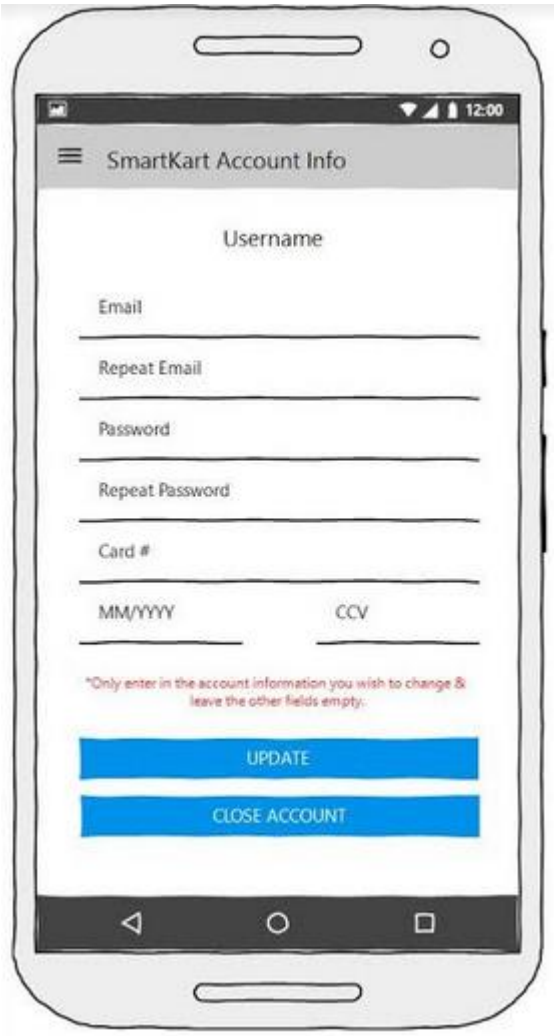


Figure 48. Profile Page

d. Home

On the home page, if the user has a valid card on their account then they will be able to see the Start Shopping box, which displays all the carts available. If their account does not have a valid card, then they will not be able to see the Start Shopping box and see the alert message in the diagram. If the user has made previous purchases they will be able to see a summary of their most recent purchase.

Figure 49. Home Page

e. Purchase History Summary

On the purchase history page, users can scroll through the summaries of the purchases they made. The summary will show the cost, date, how much they saved, and how many items they purchased. If the user clicks on the details label they will be redirect to the Purchase History Details page, which will display all the items they purchased and coupons applied to their purchase. If they select the PDF label the user will download a copy of their receipt. Users will be able to filter through their purchases by store, date, and amount spent.

Figure 50. Purchase History Page

f. Purchase History Details

In the Purchase History Details page the user can scroll through all the products they purchased and their price and coupons applied to their purchase. At the beginning of the page the user should be able to see the date, location of store, and purchase id related to this purchase. Although, the user can scroll through purchased products and purchase information, they should always be able to see the total cost of their purchase at the bottom of the screen.

Figure 51. Purchase History Details Page

g. Coupons

Users will not be required to have a valid card on their account in order to see the coupons offered by a selected store. Each coupon available should display an image, type of coupon, small description, and amount saving. Users will have the ability to search through coupons based on the type (BOGO, 50%, clearance, etc.). Note, users will only be able to see the current coupons being accepted by the store.

Figure 52. Coupons Page

h. Current Cart

Users should be able to view everything currently in their shopping cart. They will be able to scroll through the products in their shopping cart, applied coupons, while always being able to see the running total of the shopping cart. If the user notices they have an item in their cart that was not automatically removed, they can remove the item from their shopping cart.

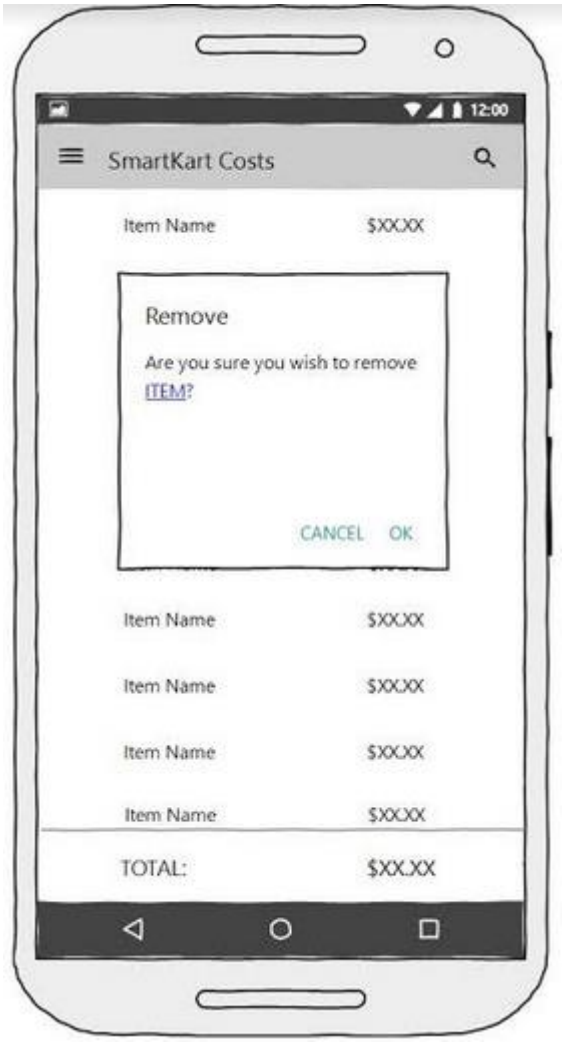


Figure 53. Current Cart Page

5.2.3.2. System's API

Desired API calls for system that allow users to interact with system. There are four different components for the API portion of the system, each intended for different sets of users interfacing with the system. The one of the components is for the devices on the shopping cart, while two components allow users to get information about available coupons/items at the store or edit/review account information and review previous purchases, and the last component is strictly for store employees. If a request is successful we set the isError flag to false, but if there is an error we set it to true and return an error message. For the GET requests the data being received is in the URL query string, while for the other request types that data is being wrapped in the body.

a. Cart

API calls that allow the devices on the shopping cart to communicate with system to add and/or remove products from current customer using the shopping cart. Route to access these requests is: /cart/<NAME>

Table 8. Cart API calls

METHOD	NAME	RECEIVE	RESPONSE
POST	Connect	cartId	token
PUT	Add	cart's token itemId	success/error message
DELETE	Remove	cart's token itemId	success/error message

- connect will generate a token with the cartId's for the system to validate the cart whenever it makes a request to the system
- add will tell system to add the item to the user's running shopping cart
- remove will tell system to remove the item from the user's running shopping cart

b. Shopping

API calls to allow users to see products and coupons available at current store. Route to access these requests is: /shopping/<NAME>. As you can see, we do not require users to send an accountId to make these API requests. This is because, it would be poor business to require people to have an account to see coupons and products available at a selected store.

Table 9. Shopping API calls

METHOD	NAME	RECIEVE	RESPONSE
GET	coupons	storeId date types: [] pageIndex	list of coupons that match types selected success/error message
GET	products	storeId filter: minCost maxCost types: [] showOutOfStock pageIndex	products: Image URL name cost success/error message
GET	products/{id}	storeId	isle number name image URL stock count success/error message

- coupons will fetch the next 25 coupons available at specified store, default date sent is current date so shows current coupons
- products without an itemId will fetch summary for next 25 products available at specified store that match provided filters
- products with itemId will fetch detailed information of specified item

c. User

API calls to allow users to update account information and fetch purchase history. Users can only see information related to their account. Every API request here requires the user's accountId so it can make sure other users cannot see other user's account information. Route to access these requests is: /user/<NAME>

Table 10. User API calls

METHOD	**NAME**	**RECIEVE**	**RESPONSE**
POST	Login	username password	Token success/error message
POST	Connect	token cartImage	success/error message
PUT	updateAccount	token accountInfo:	success/error message

METHOD	NAME	RECIEVE	RESPONSE
POST	Create	accountInfo:	Token success/error message
POST	Logout	token	success/error message
GET	payHistory	token filter: dateRange storeId pageIndex	list of previous purchases summaries dateTime total store storeId checkoutId
GET	payHistory/{id}	token checkoutId	list of every item purchased in that checkout name cost checkout total location
DELETE	removeAccount	token	success/error message

- login will generate a token for the user if they entered the correct credentials, else return error message
- connect will notify system to bind user to a shopping cart, so every time a device adds/removes item to cart it updates user's shopping cart
- updateAccount allows user to update certain aspects of account, if they change their card information, we must validate it before allowing users to make purchases
- create notifies system to add user to system, must validate user's payment card
- logout notifies system to get rid of session/token for current user
- payHistory without checkoutId will fetch next 25 previous purchases based on filter options
- payHistory with checkoutId fetches all items bought, applied coupons, total, date, location, and purchaseId for selected purchase
- removeAccount will mark user's account as deleted so it may no longer be accessible

d. Store

This will allow stores to easily interact with the system allowing them to accept customer returns and edit product and coupon information. Every request to these

URLs require somebody with a store employee account. For security purposes, we must not allow customers that work at the selected store to make returns, however, somebody else can make a return for them. For instance, if Bob worked at Publix A they must not be allowed to make returns on their purchases with their account, they would need another employee to accept their return. For the update APIs, users can just send the elements they wish to change for the selected product and coupon. Although, it is mandatory for users to send all the required information when they wish to add new products and coupons to the store. Route to access these requests is: /store/{storeId}/<NAME>

Table 11. Store API calls

METHOD	NAME	RECEIVE	RESPONSE
POST	makeReturn	employeeId checkoutId products: [{ids, cost}]	amount reimbursed new total success/error message
POST	coupon	employeeId info: type productid	couponId success/error message
PUT	coupon/{id}	employeeId info: type productId	success/error message
DELETE	coupon/{id}	employeeId	success/error message
POST	coupon	employeeId productInfo: isle # cost name type amountInStock	productId success/error message
PUT	product/{id}	employeeId info: name cost type amountInStock	success/error message
DELETE	product/{id}	employeeId	success/error message
POST	pushCouponUpdates	employeeId couponId	list of all coupons updated success/error message

METHOD	NAME	RECEIVE	RESPONSE
POST	pushProductUpdates	employeeId productId	list of all products updated success/error message

- makeReturn: will first verify if the products the customer is trying to return is on the provided checkout. If it is we will then use the Stripe API to make a return to the customer's provided credit card
- coupon(POST): will allow store to create new coupons. New coupons must be bound to a product currently sold by the store and is not marked as having changes.
- coupon(PUT): will make changes to store coupons. This is intended to be used to make new coupons for the next week, but it can be used to make changes to previous or current coupons. Any coupon updated will be flagged as having changes and will not be available to customers until the coupon are validated.
- coupon(DELETE): will mark coupon as no longer to be used at store. Can be used to remove coupons that were not verified.
- Product(POST): will be used to add new products to the store. If successful in adding it to the database, we will return the products id
- product(PUT): will make changes to products currently sold at the store.
- product(DELETE): will mark product as no longer sold at the store. Should only be used if store knows they will no longer be selling a product, should not be used if a product runs out of stock.
- pushCouponUpdates: will push the changed and newly added coupon(s) to the public. If provided a single oouponId will only push that single coupon, but if no couponId is provided all newly and changed coupons will be pushed.
- pushProductUpdates: will push the changed and newly added product(s) to the public. If provided a single productId will only push that single product, but if no productId is provided all newly and changed products will be pushed.

5.2.3.3. Third-Party API Calls

The bellow section covers the functions we will need to call from the APIs we intend on using.

Table 12. Package installation commands

API	Installation instruction
Bcrypt	npm install bcrypt
Express	npm install express
Passport-local	npm install passport npm install passport-local
pdfKit	npm install pdfkit
QRcode-reader	npm install qrcode-reader
sendGrid	npm install sendgrid
socket.io	npm install socket.io
Stripe	npm install stripe

a. Bcrypt

```
// load modules & create instance
var bcrypt = require('bcrypt');


// define # of rounds going to salt data const
saltRounds = 8;


// autogenerate salt and hash

bcrypt.hash(myPlaintextPassword, saltRounds, function(err, hash) {

  ...

});

// check password, will go inside defined callback function for database call
bcrypt.compare(unhashedPassword, hash, function(err, res) {

   ... // additional logic for creating session

});
```

b. Express

```
// create instance
```

```
var express = require('express');
var app = express();

// routing

   // get, doesn't have body so everything must be in url
app.get('/user/:id', function(req, res) {        res.send('user '
+ req.params.id);

   });

   // post

   app.post('/', function (req, res) {        res.send('POST request
to homepage, body: ' + req.body);     });


   // delete

   app.delete('/', function (req, res) {        res.send('DELETE
request to homepage, body: ' + req.body);

   });

   // put

   app.put('/', function (req, res) {        res.send('PUT request
to homepage, body: ' + req.body);     });


   // route     app.route('/events').get(function(req, res, next)
{...});


// response     // download file
res.download('path/to/file');


   // render     res.render('<view_name>', {<data_load_in_view>},
function(err, html) {        ...

   })


   // json
res.json(<data>);


   // send
```

```
res.send([BODY]);
```

c. Passport-local

```
// create instance
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;


// validate if user is who they say they are
passport.use(new LocalStrategy({
usernameField: 'email',     passwordField:
'passwd'
  },
  function(username, password, done) {
    // ...
  }
));
[CALL FROM EXPRESS]
app.post('/login',
passport.authenticate(
    'local',
{
      successRedirect: '/',
failureRedirect: '/login',      failureFlash:
true
    }
  )
);
```

d. PdfKit

```
// load modules & create instance
var PDFDocument =
require('pdfkit');
var doc = new PDFDocument;
```

```
// automatically starts with a page //
add new page to document
doc.addPage();


// switch to different page
doc.switchToPage(<int>);


// done defining data writing to pdf doc.end();

// include image

// if don't define any position cordinates then it is placed at current cursor

// if don't define a width and heigth, image is displayed at full size
doc.image('<image>', <x_pos>, <y_pos>, width, height);


// include links doc.link(<x_pos>, <y_pos>, width, height,
'http://google.com/');


// define file name doc.info.Title
= "<file_name>";


// write to buffer

var stream =
doc.pipe(blobStream());
stream.on('finish', function() {
...

});
```

e. QRcode-reader

```
// load modules & create instance
var QrCode = require('qrcode-
reader'); var qr = new QrCode();


// define callback to be ran after decodes
image qr.callback = function(result, err) {
if(err) {
```

```
console.log(err);     }
else {        ...
   }
}

// decode data
// passing it the image obtained from the qr code scan
var data = canvas.getContext("2d").getImageData(0, 0, width, height);
dr.decode(data);
```

f. SendGrid

```
// load modules & create instance var sg =
require('sendgrid')(process.env.SENDGRID_API_KEY)
// create request
var request = sg.emptyRequest();
// define content
request.body = {
"attachments": [
   {
     "content": "[BASE64 encoded content block here]",
     "filename": "receipt_<transaction_id>.pdf",
     "name": "receipt",
     "type": "pdf"
   }
 ],
 "content": [
   {
     "type": "text/html",
     "value": "<html><p>Hello, world!</p><img src=[CID GOES
HERE]></img></html>"
```

```
    }
  ],
  "from": {
    "email": "example@example.com",
    "name": "Smart Kart"
  },
  "personalizations": [
    {
      "subject": "Hello, World!",
      "to": [
        {
          "email": "john.doe@example.com",
          "name": "John Doe"
        }
      ]
    }
  ],
  "subject": "Hello, World!",
  "template_id": "[YOUR TEMPLATE ID GOES HERE]" };


// send request to sendgrid to send email via sendgrid
request.method = 'POST' request.path = '/v3/mail/send'

sg.API(request, function (error, response) {
if(error) {


console.log(error);     }
else {        ...

    }

});
```

g. Socket.io

```
// server api calls
// create instance
   var server = require('https').Server(app);
   var io = require('socket.io')(server);
   // connect to server
   io.on('connection', function (socket) {
socket.emit('news', { hello: 'world' });
   });


   // send data to specified user/device     socket.to(<socketid>).emit('hey', 'I
just met you');


   // send to all users     socket.broadcast.emit('broadcast',
'hello friends!');


// client api calls
<script src="/socket.io/socket.io.js"></script>
<script>   var socket =
io.connect('http://localhost');
socket.on('<chatroom>', function (data) {
...
  });
</script>
```

h. Stripe

```
// create instance var stripe =
require("stripe")("API_KEY");


// create customer so can charge them for later
use stripe.customers.create({   email:
"paying.user@example.com",   source: token,
}).then(function(customer) {
  // save customer's id our database so can make charges to account later times
```

```
});


// charge customer's card, call when retrieve customer's customerId when
checking out stripe.charges.create({   amount: <amount>, // in cents
currency: "usd",   customer: customerId,

});


// fetch customer info

stripe.customers.retrieve(<customerId>,
function(err, customer) {

   // logic

 }

);

// update account

stripe.customers.update(<customerId>, {

      description: "Customer for sofia.wilson@example.com",
source: {          object: "card",          exp_month: MM,
exp_year: YYYY,          number: "<card_number>",
cvc: XXX          name: "<card_holders_name>"

      }

   }, function(err, customer) {
...

});


// returns, can call as many times until charge is completely refunded
stripe.refunds.create({     charge: <customer_selected_charge>,
amount: XXXX, // in cents

   },     function(err,
refund) {       ...

   }

);


// response errors from request
card_error: card can't be charged
```

// codes

invalid_number: not valid credit card #
invalid_expiry_month: not valid expiration
month invalid_expiry_year: not valid expiration
year invalid_cvc: not valid CVC code
expired_card: card has expired card_declined:
card was declined

6. Hardware Prototyping and Testing

This section gives a detailed overview of the component testing and prototyping that our group implemented for our SmartKart shopping system, including the hardware and software designs integrated in our system.

The first subsystem consists of the RFID sensor and the WiFi module. The RFID sensor will read in store item RFID tags and will pass the RFIDs to the WiFi module. The Wifi module will then send the RFIDs to the database, so that the progressive web application can them to update the items and overall pricing. Further information on these two components will be covered below.

The second subsystem consists of the GPS module and the electromagnetic lock. The GPS module will receive data from at most 4 satellites to generate its location and based on the location the cart's locking mechanism will be enabled. If the GPS on the shopping cart off store property it will send the signal to lock the cart. Further information on these two components will be covered below.

6.1. Voltage Regulator Design

For this project, the microcontroller and its components will need to be provided with a voltage range of 3.3V to 5V. The voltage needs to be constant DC voltage and should remain constant even when the input voltage varies slightly. A 12V DC battery will provide the input voltage to the regulator. Once a suitable voltage regulator design has been developed, it will then be implemented in a PCB and will be used to power the microcontroller.

6.1.1. Linear Voltage Regulator

A simple step-down linear voltage regulator was first considered to supply a suitable input voltage to the microcontroller and its components. The design uses a L7805 IC and two capacitors C1 and C2. A schematic of this can be seen in Figure 54 below. The regulator takes in an input voltage of 9V to 15V, and supplies an output voltage of 5V. The capacitors values C1 is 1.0 microfarads and C2 is 10.0 microfarads.

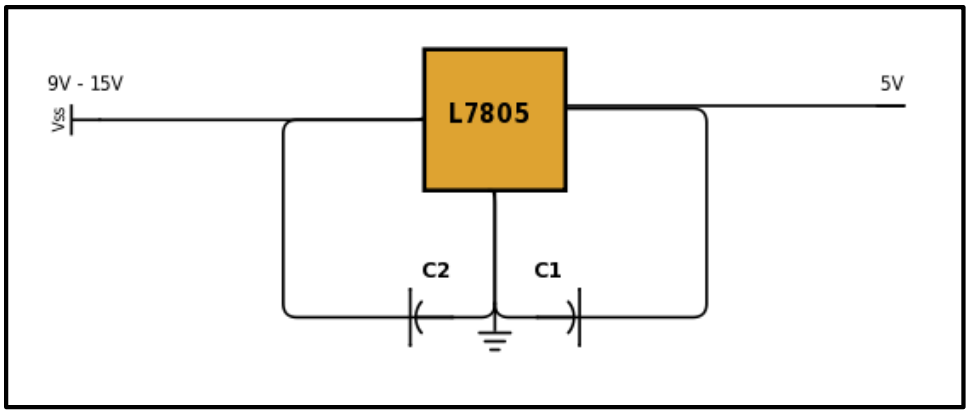


Figure 54: Linear Voltage Regulator Schematic

Note that in the Figure Y, C1 is equal to Cx in series with Cy. During testing, a 1.0 microfarad capacitor could not be found, so Cx and Cy, which are both 2.2 microfarads, were placed in series. This creates a capacitance value approximately equal to 1.1 microfarads. The input voltage range was tested, and from 9V to 15V, the output voltage stayed at a constant 5V. These results are satisfactory for the project, and should meet the needs of any component that requires an unchanging voltage.

6.1.2. Step-Down Voltage Regulator

Another regulator that was considered to supply power for the system was a step-down voltage regulator. Using a step-down voltage regulator over the previous linear voltage regulator would yield positive impacts. Step-down voltage regulators dissipate less power, which means minimal effort would be needed in designing a heat sink. Since less power is dissipated, a step-down regulator is able to achieve a much higher efficiency than a linear voltage regulator. Step-down voltage regulators are generally more stable than linear voltage regulators, with greater protection against sudden changes in input voltage.

To aid in the design of a step-down voltage regulator for the power supply, Texas Instrument's WEBENCH software was used. An input voltage range of 11.5V to 12.5V was sought after with an output voltage of 3.3V and a maximum load current of 2A. The efficiency function of the circuit is very logarithmic and shows that as the load current increases, the efficiency increases. The power module used for this regulator is the LMZ12010. This module provides an implementable step-down DC-to-DC converter and can support a maximum load current of 10A.

Shown below is an EAGLE CAD schematic of the step-down regulator.

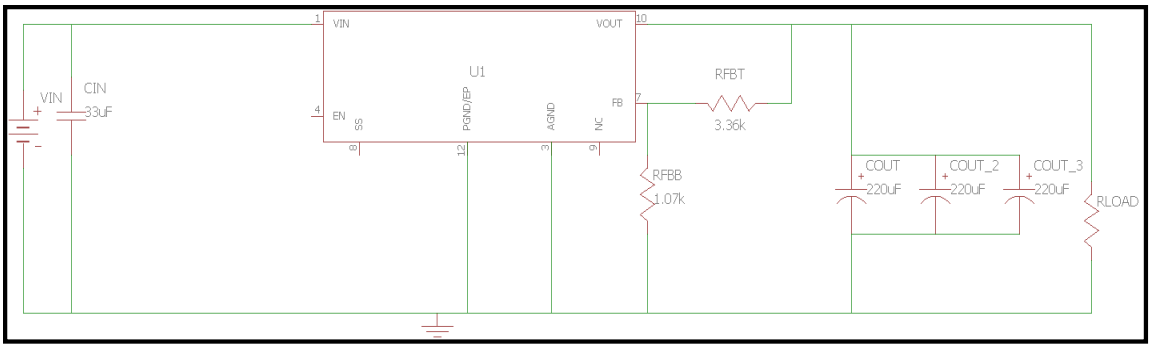


Figure 55: Step-down Voltage Regulator Schematic

A table depicting the component details of the schematic is shown below.

Table 13: Step-down Voltage Regulator Schematic Components Description

Reference Designator	Value	Part Number	Type
CIN	33µF	3M0802	Electrolytic Capacitor
RFBT	3.36KΩ	-	Carbon Composite Resistor
RFBB	1.07KΩ	-	Carbon Composite Resistor
COUT, COUT_2, COUT_3	220µF	3M0802	Electrolytic Capacitor
RLOAD	1.65Ω	-	Load Resistance
U1	-	LMZ12010	Step-down DC-to-DC Converter

The components of this step-down regulator should be easily obtainable and fairly easy to build both on the breadboard and PCB. The LMZ12010 is 15 millimeters squared, which is relatively small and will take up minimal space on the PCB.

6.1.3. Comparison

To decide which voltage regulator would be the better choice to supply power for the system, the datasheets for the LM7805 and LMZ12010 were perused. Some of the key features to consider were efficiency, input and output voltage, operating temperature, maximum load current, and pricing. The follow table is depicted below to show a comparison of some the features these ICs exhibit.

Table 14. Voltage Regulator chip comparisons

Features	LMZ12010	LM7805
Operating Voltage	6V to 20V	7V to 35V
Maximum Output Current	10A	1.5A
Output Voltage	Adjustable	Fixed
Operating Temperature	-40 °C to 125 °C	-65 °C to 150 °C
Maximum Efficiency	92%	41%
Switching Frequency	350KHz	N/A
Unit Price	$2.45	$0.46

It is seen in the comparison that the price of the LMZ12010 is significantly higher than the LM7805. Pricing is not much of a concern for this project, and spending 2 dollars more for a chip should not be seen as a negative factor. The operating voltages for both chips are within our need 3.3V to 5.0V range and the maximum output currents greatly exceed the 2mA load current the microcontroller will need. The output voltage for LM7805 always remained constant, while the LMZ12010 adjust its output voltage. An adjustable output voltage could prove useful for instances where 5V is needed for components and 3.3V is needed for others. Probably the most important factor of all in this comparison is the efficiency. The LM7805 is only able to produce a maximum efficiency of 41%, while the LMZ12010 is able to produce one of 92%. The LMZ12010's maximum efficiency is more than twice that of the LM7805. As was said earlier, a lower efficiency value means higher power dissipation. This means the chip is likely to heat up to high temperatures, and a heat sink might have to be introduced to negate this. However, the LMZ12010 has a very high efficiency, and although it will still dissipate power, the amount of generated heat within the chip is relatively low. It is for this reason that the step-down voltage regulator will be used to supply power to the microcontroller.

6.2. Components

As seen below, Figure X shows the assorted parts required for building the system.

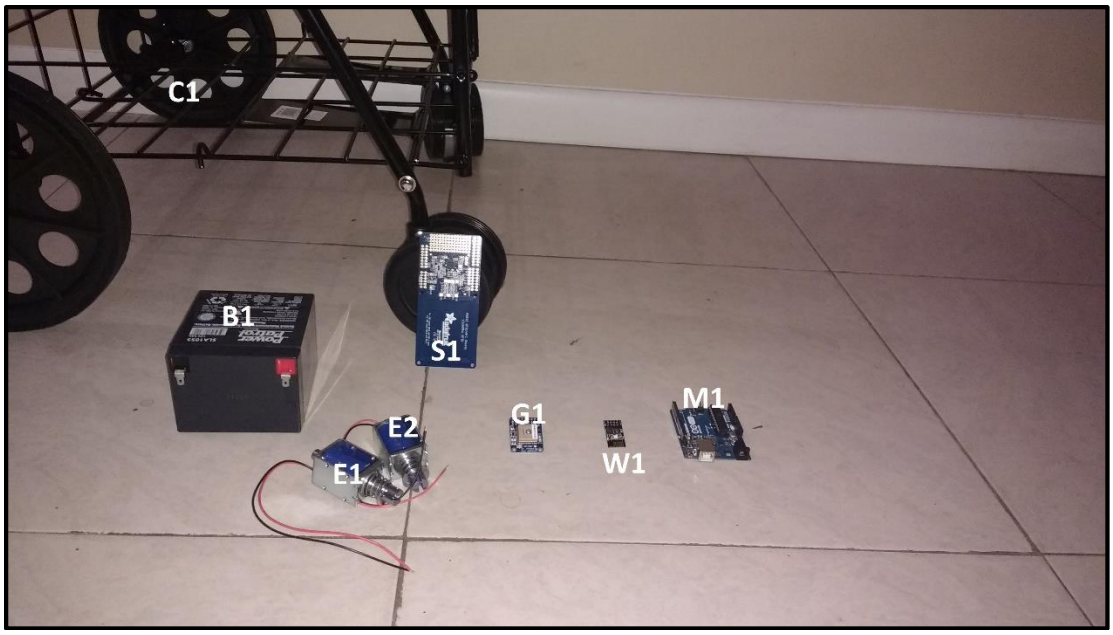


Figure 56: Assorted System Parts

A table detailing the name of each part is listed below.

Table 15. Parts list

Reference	**Part Name**
C1	OxGord Folding Shopping Cart, Black
B1	Power Patrol 12V 5A Sealed Lead Acid Battery
S1	PN532 RFID/NFC Shield 13.56MHz RFID
E1, E2	Uxcell Stroke Open Frame Electric Linear Solenoid, DC 12V, 5.6 Ohm, 1 kg Force, 10 mm
G1	Adafruit Ultimate GPS Breakout
W1	ESP8266 Wifi Module
M1	ATmega328 (Arduino Uno)

6.2.1. Breadboard Testing

Testing was conducted for each component to see the overall functionality of the components. The pins for each component were connected to a breadboard, and from there, connected to Arduino UNO for testing. Shown below are three figures that depict the breadboard component testing.

Figure 57: PN532 Test Setup

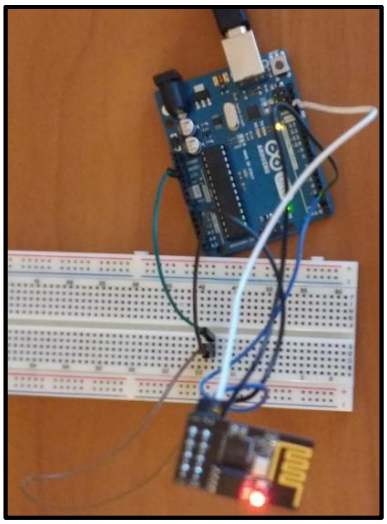


Figure 58: ESP8266

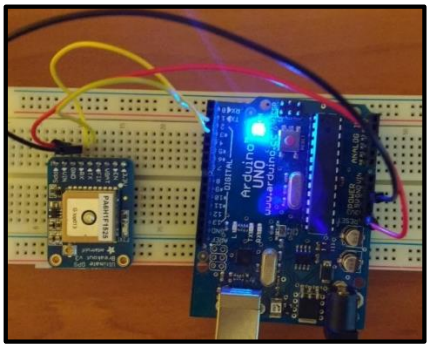


Figure 59: GPS Module Test Setup

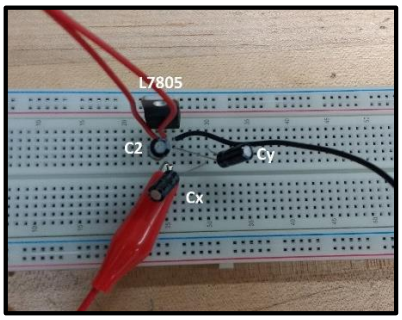


Figure 60: Linear Voltage Regulator Breadboard Test

6.2.2. GPS

This section details the GPS module of our SmartKart shopping system. It includes the design of the module, as well as the test procedures and results.

6.2.2.1. Design

Seen below are the pin configurations for the Adafruit's Ultimate GPS Breakout v3 module.

Figure 61: Adafruit Ultimate GPS Breakout Pin Configuration

Seen below is a table which describes the use of each pin.

Table 16. Adafruit Ultimate GPS Breakout pin description

Pin Number	Pin Configuration
1	PPS
2	VIN
3	GND
4	RX
5	TX
6	FIX
7	VBAT
8	EN
9	3.3v

6.2.2.2. Testing

Adafruit provides a library of functions that aid in interfacing with the GPS module. Establish a connection when call SoftwareSerial mySerial(TXPin,RxPin). The GPS module's TX and RX pins transmit data at a baudRate of 9600, requiring the read frequency to be lower to lower chances of background interrupts affect main thread operations. Calling the GPS.sendCommand() function with a checksummed NMEA sentences to define the data wish to acquire and configure number of times to perform reads per second. The GPS can return the time and

date, number of satellites connected to, and longitude and latitude via Google Maps and perform logging. Test results show this GPS module is within a ten-meter range of the actual location. The test set up for the Ultimate GPS Breakout can be seen above in Figure 59.

6.2.3. Locking

The Uxcell Electric Linear Solenoid takes in 12V DC to push the rod out a distance of 8 millimeters. The electromagnetic has two wires. One wire supplies the load current to directly power the electromagnet, and the other wire goes out to ground. During testing, the electromagnet pushed out the rod with a decent amount of force. The amount of friction and force was enough to lock the rear wheel of the shopping cart when it came into contact with the rod. The electromagnet's input voltage will not be powered by the buck converter. Rather, the input voltage will be supplied directly by the 12V sealed lead acid battery. The microcontroller will have control over when 12V will be supplied to the electromagnet. Further testing will need to be done to see how this approach will be implemented.

6.2.4. RFID

This section details the RFID module of our SmartKart shopping system. It includes the design of the module, as well as the test procedures and results.

6.2.4.1. Design

Seen below are the pin configurations for the Adafruit PN532 RFID/NFC Shield:

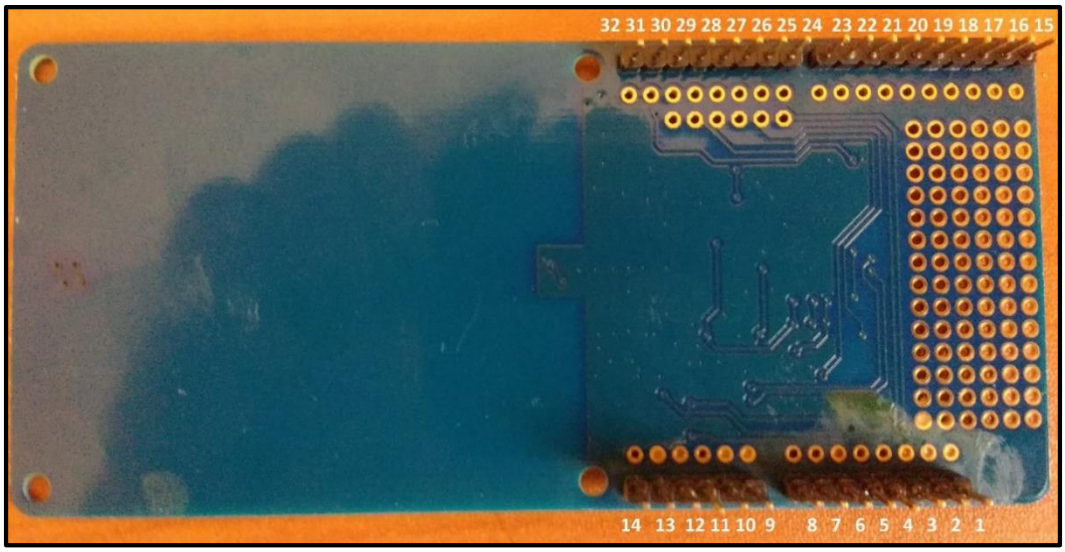


Figure 62: Adafruit PN532 RFID/NFC Shield Pin Configuration

Shown below is a table detailing the description of each pin.

Table 17. Adafruit PN532 RFID/NFC Shield pin description

Pin Number	Pin Description	Pin Number	Pin Description
1	N/A	**17**	ARef
2	IQR	**18**	GND
3	RST	**19**	Pin 13, Digital I/O
4	3V	**20**	Pin 12, Digital I/O
5	5V	**21**	Pin 11, Digital I/O
6	GND	**22**	Pin 10, Digital I/O
7	GND	**23**	Pin 9, Digital I/O
8	Vin	**24**	Pin 8, Digital I/O
9	Pin 0, Analog In	**25**	Pin 7, Digital I/O
10	Pin 1, Analog In	**26**	Pin 6, Digital I/O
11	Pin 2, Analog In	**27**	Pin 5, Digital I/O
12	Pin 3, Analog In	**28**	Pin 4, Digital I/O
13	Pin 4, Analog In	**29**	Pin 3, Digital I/O
14	Pin 5, Analog In	**30**	Pin 2, Digital I/O
15	SCL	**31**	Pin 1, Digital I/O
16	SDA	**32**	Pin 0, Digital I/O

The RFID sensor can communicate using I2C by default, but can also communicate using SPI. For this project, I2C will be used because less pins are

used. I2C uses 3 pins overall. Analog input pins 4 and 5 are used to communicate and digital I/O pin 2 is used as to generate interrupt service routines (ISR).

6.2.4.2. Testing

The Adafruit PN532 RFID/NFC Shield was attached to an Arduino UNO and some code was uploaded to the UNO to test if the shield would read RFID tags. Additionally, maximum distance in which the PN532 was able to read RFID tags was also sought after. The test setup can be seen in Figure 57.

Adafruit provides a library of functions that aid in reading and writing RFIDs. An I2C connection can be established by calling on Adafruit_PN532 nfc(PN532_IRQ, PN532_RESET). PN532_IRQ is denoted as the interrupt pin (pin 2) and PN532_RESET as the reset pin (pin 3). The readPassiveTargetID() function is responsible for reading RFID tags that come into contact with the PN532. The PN532 is compliant with ISO 14443A and MIFARE tags. The maximum distance in which an RFID tag was read was approximately 3.5 inches.

6.2.5. Wi-Fi Communication

This section details the WiFi module of our SmartKart shopping system. It includes the design of the module, as well as the test procedures and results.

6.2.5.1. Design

Seen below are the pin configurations for the ESP8266 WiFi Module.

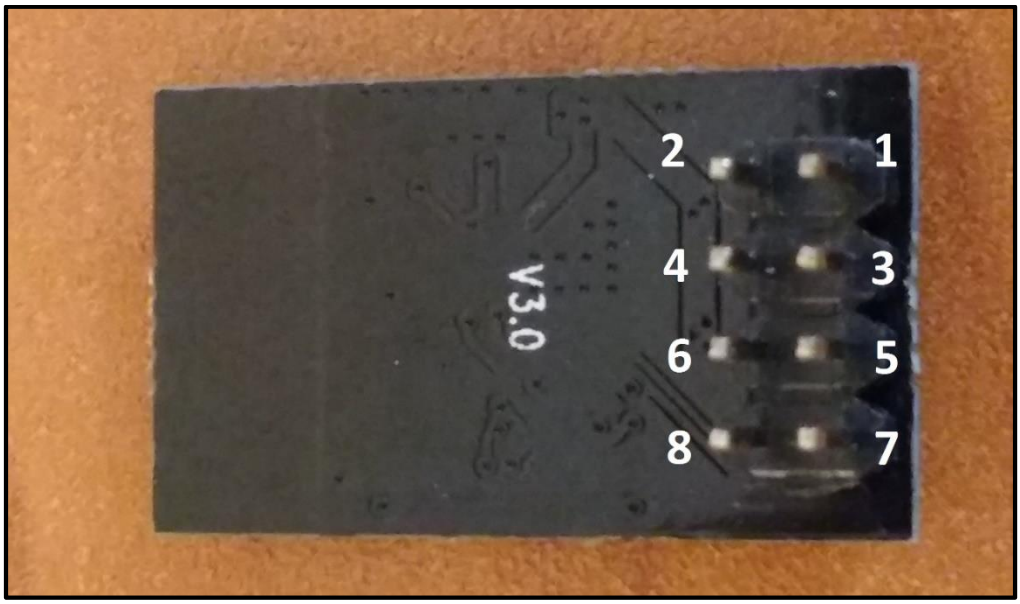


Figure 63: ESP8266 WiFi Module Pin Configuration

Seen below is a table which describes the use of each pin.

Table 18. ESP Wi-Fi module pin description

Pin Number	Pin Configuration
1	TX
2	GND
3	CH_PD
4	GPIO2
5	RST
6	GPIO0
7	VCC
8	RX

6.2.5.2. Testing

Pins 3, 4, and 6 are not used for any purpose in the testing, and are therefore left unconnected. The VCC pin is only able to within an input voltage of 3.3V. Anything above this might damage the hardware of the ESP8266. During testing, the ESP8266 was able to stay connected to a router over varying distances. The longest recorded range the ESP8266 was able to stay connected was roughly over 1,000 feet. Arduino libraries are provided to easily connect to a network. The ESP8266 library provides a begin() function, where the network name and password are passed in to establish a connection. Figure 58 shows the setup of the WiFi module along with the Arduino UNO.

6.3. Schematic Design

A schematic of the overall system wired around the ATmega328P has been produced. It was important to consider some details when designing the microcontroller schematic. First, the microcontroller and other integrated circuits are prone to being damaged by ripple voltage when present. To resolve this, decoupling capacitors were used at the pins where the primary voltage for the microcontroller and integrated circuits were supplied. A secondary external clock is desirable to control interrupts within the system. A 32MHz crystal oscillator is used to achieve this, with a two 6 picofarad capacitors used as the load

capacitance. The headers for each integrated circuit are wired to the respective matching pin on the microcontroller.

To drive the 12V electronic locks, a sealed relay, diode, and PNP bipolar junction transistor is used to switch the input voltage of the locks on and off. Digital pin 7 on the microcontroller is used to switch the electronic locks on. When PD7 is switched to HIGH, 3.3 volts is supplied to the base of the transistor, thus providing 12 volts to reach positive terminal of the electronic locks.

Figure X below depicts the microcontroller schematic. The schematic was made through the EAGLE CAD software. The headers and pin outlets for the WIFI module, GPS module, RFID sensor, and electronic locks can be seen below.

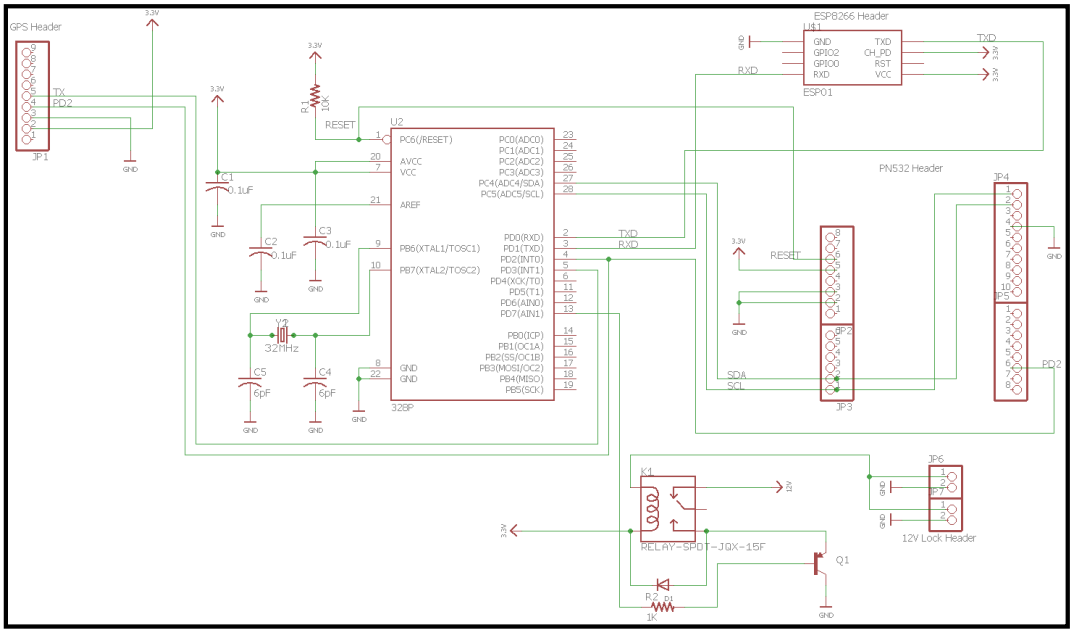


Figure 64: Microcontroller Schematic

A table depicting the component details of the schematic can be seen below.

Table 19. Microcontroller Schematic Components Description

Reference Designator	Value	Part Number	Type
C1, C2, C3	0.1µF	3M0802	Electrolytic Capacitor
C4, C5	6pF	3M0802	Electrolytic Capacitor
U2	-	ATmega328P	28 Pin
R1	10KΩ	-	Carbon Composite Resistor

Reference Designator	Value	Part Number	Type
R2	1KΩ	-	Carbon Composite Resistor
Y2	32MHz	XTAL13459	Crystal Oscillator
D1	-	1N4004	General-Purpose Diode
Q1	-	P2N2222AGOS-ND	PNP BJT Transistor
K1	-	COM00100	Relay SPDT Sealed
JP1	-	-	GPS Module
ESP01	-	ESP8266	WIFI Module
JP2, JP3, JP4, JP5	-	PN532	RFID Sensor
JP6, JP7	-	AL12120600UX0144	Electric Lock

7. Administrative content

This section contains all portions of the project related to management, including our team's milestones, budget plans, tools used, and the division of labor among team members.

7.1. Milestones

The dates set for the milestones were determined based on the task and each individual member's individual schedules and duties in regards to that task. The first set of milestones accomplished in Senior Design I are heavily related to researching and designing the system, whereas the Senior Design II milestones focus on the functionality portion of our project.

Table 20. Senior Design I Milestones

Number	Task	Start Date	End Date	Status
1	Ideas	1/9/17	1/17/17	Completed
2	Project Selection & Role Divisions	1/18/17	1/21/17	Completed
	Project Documentation			
3	Divide & Conquer	1/22/17	2/3/17	Completed
4	Updated Divide & Conquer	1/22/17	2/17/17	Completed
6	Draft of Documentation	1/22/17	3/31/17	Completed
7	Final Documentation	1/22/17	4/27/17	Completed
	Research & Design			

Number	Task	Start Date	End Date	Status
8	Research Components	1/25/17	4/26/17	Completed
9	Purchase Components	2/4/17	4/26/17	Completed
10	Prototype Hardware and Software	4/5/17	4/26/17	Completed

Table 21. Senior Design II Milestones

Number	Task	Start Date	End Date	Status
	Administrative Tasks			
1	CDR Presentation	TBD	TBD	N/A
2	Peer Review	TBD	TBD	N/A
3	Midterm Demo	TBD	TBD	N/A
4	Final Presentation	TBD	TBD	N/A
	Technical Development			
5	Software Development	TBD	TBD	N/A
6	Hardware Development	TBD	TBD	N/A
7	PCB Testing	TBD	TBD	N/A

Number	Task	Start Date	End Date	Status
8	Software and Hardware Integration	TBD	TBD	N/A
9	Finalize System Integration and Testing	TBD	TBD	N/A

7.2. Budget Analysis

The following table displays the estimated costs for each individual component, as well as the total cost for the system. As this project will be funded by our team, the total cost will be evenly split between all four group members.

Table 22. Project budget

OBJECT	QUANTITY	EST. COST
RFID Scanner	1	$39.95
RFID Tags	50	$19.20
GPS 66 Channel Breakout Board	1	$39.95
Arduino Uno	1	$0.00
PCB	2	$70.00
Amazon AWS Backend	1	$50.00
Payment System	1	$50.00

OBJECT	QUANTITY	EST. COST
Shopping Cart	1	$40.00
BreadBoard (830 pins)	1	$2.99
40W Soldering Iron	1	$40.68
Solder Soldering Wire	1	$7.00
	Total:	$359.77
Est. Total	$359.77	
Number Group Members	4	
Est. Indv. Total	$89.94	

7.3. Tools

Here we discuss the tools our group used for writing and testing our code, communicating, and tracking assigned tasks.

7.3.1. POSTMAN

POSTMAN is available as a desktop and chrome application. Allows the developers to do testing for the system's RESTful API by sending body data, selecting header type, defining route URL, and include authentication information. With POSTMAN, the developers can perform testing the API running on localhost and in production and store a log of the requests made and responses returned. With POSTMAN's runner functionality, the developers can easily import test cases to run on the API to perform in-depth testing. With POSTMAN's Collection feature, data from the request and responses can be organized in a way that mirrors the layout of the API architecture. For instance, every request made to the API route:

api.example.com/math/multiply?lhs=<value>&rhs=<value> would generate a folder with the test data used and responses called multiply inside the math folder.

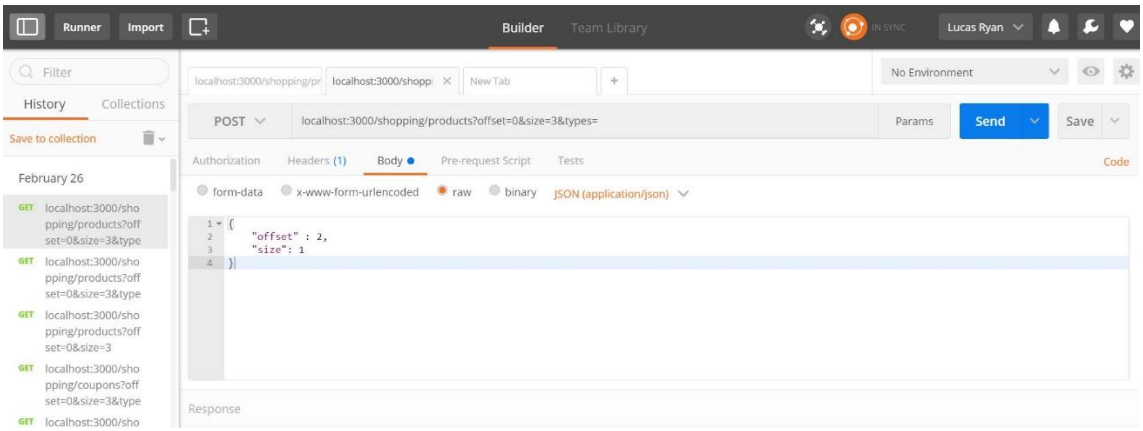


Figure 65. Postman Layout

7.3.2. GitHub

A web-based Git version control repository. GitHub has free accounts, however, if one requires private repository so the public cannot access their code we would require accounts that offer private repositories. As students, we can apply for a student developer account that allows students to create an unlimited number of private repositories. One must create an account with an academic email and provide evidence of current enrollment to receive the account. Once their information has been verified the student can make free private repositories and have access to a vast number of discounts for APIs and resources to build applications. With GitHub, the developers can work on the system without having to meet up or modify the system's current build. Whenever a developer commits code to the repository, they must include a link to the task their code relates to, that way it is easy to keep track of changes in the case a revert is necessary.

7.3.3. Chrome Developer Tools

Set of web debugging tools that assist developers with modifying webpage layouts, finding bugs in their code, and monitor how data is being transmitted between the client and server. There is a feature to modify the view size of the page to easily test how the web app looks and performs on different layout dimension. There are nine main Developer Tool groups, however, we would be using the following ones listed below:

Elements: allows for developers to make changes to the webpage's DOM and review element properties locally. This is the best tool to use when need to test

modifications of layout instead of modifying the source code and refreshing the page.

Resources: allows developers to inspect and manage how data from the site is being stored on the client's device. Developers can review how session data, and cached data is being stored. This can help developers check if the service workers created are properly working.

Network: allow developers to review detailed information on network operations like data sent, time to send and receive data, the destination sending the data and data type. With the record feature, we can get real-time results of network operations to do simple benchmarking and check if we are successfully receiving the desired data.

Timelines: while the network panel shows a timeline of how long it takes for data to be fetched, it is also important to know how long it takes for the application to run on the device. This allows the developers to know the runtime of the program and make decision for optimizing code.

Console: allows developers to interact with frontend code from browser. From here developers can create/modify data, call functions, and send requests to the server for testing. While the network tool lets developers know if the data was successfully received, the developers can use the console tool to check if the received data is correct.

7.3.4. Visual Studio Code

Microsoft's open-source, lightweight platform independent IDE that with built-in functionality and allows for 3rd party extension/plugin installation to improve development. With built-in IntelliSense, developers can use the smart completion feature to save time typing function definitions, variables, and imported classes/modules. With built-in debugging, developers can launch and run the app and review call stacks and variables at defined breakpoints in an interactive console. Visual Studio Code provides embedded Git control, allowing for developers to push, pull, and review commits. With extensions, developers can import additional features that Visual Studio Code doesn't provide like provide language support for programming languages that are not supported with general installation.

7.3.5. Slack

A group messaging application with an Android, iPhone, and Web App. We can create group chats as well as individual chats, and alert selected people in the group whenever there is a message we wish for them to read. Slack can communicate with Trello to send notifications on status of tasks, allowing the group to always know where they are in the development process. Unlike other group messaging apps, Slack makes it easily to share code between each other with

their snippet feature. Users can upload code in any programming language and provide a comment with the code snippet message. This saves time reviewing code when discussing bugs are sharing information about new helper functions to improve development process.

7.3.6. Trello

Trello is a simple solution to managing tasks, providing an easy to use elegant user interface with powerful features. In Trello, managers can organize and get detailed information on tasks. Managers can create a task, assign employees to it, set a deadline when it must be completed, attach resource document from various resources (Dropbox, Google Drive, OneDrive, computer storage, etc.), checklists of requirements that must be meet for the task to be completed, and view a log of any changes made to the card. Users can leave comments on the status of the task on the card and every user assigned to the task will get a notification of the comment. With Trello, we can easily see how everybody is doing on their assigned tasks and make any changes to the assigned workload easily.

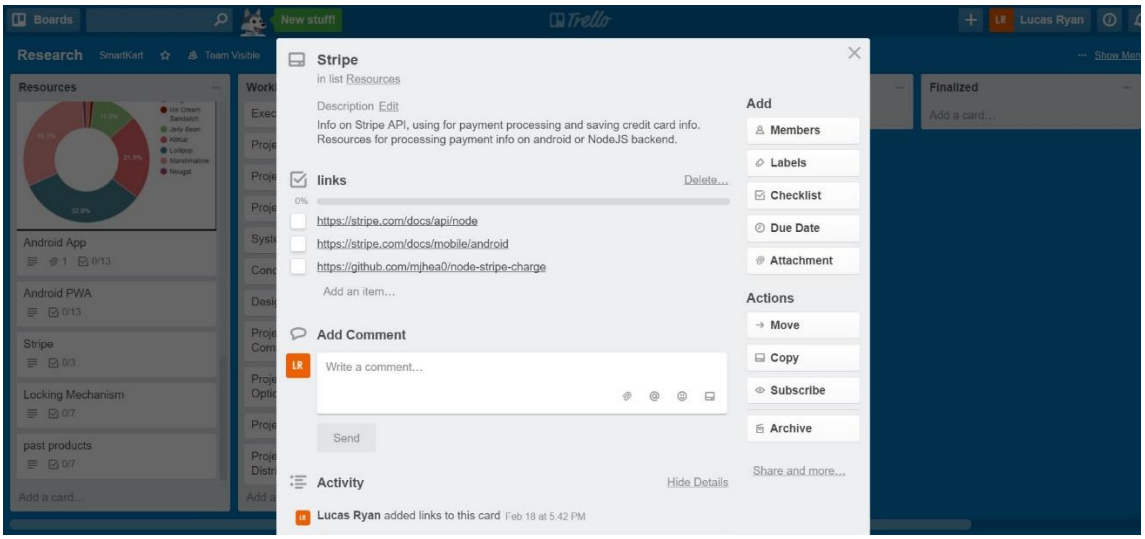


Figure 66. Trello Layout & Features

7.4. Division of Labor

This section displays the division of labor among team members. Each section of the project was assigned a primary member to assume responsibilities, as well as a secondary member to assist with that component.

Table 23. Division of Labor

COMPONENTS	PRIMARY	SECONDARY

Database	Lucas Ryan (CS)	Victoria Abreu (CS)
Email & Payment System	Lucas Ryan (CS)	Victoria Abreu (CS)
Communication Btw. Arduino	Victoria Abreu (CS)	Lucas Ryan (CS)
User interface	Victoria Abreu (CS)	Lucas Ryan (CS)
Power Distribution	Doran Senior (CpE)	Christina Heagney (CpE)
GPS	Doran Senior (CpE)	Christina Heagney (CpE)
Locking Mechanism	Christina Heagney (CpE)	Doran Senior (CpE)
RFID System	Christina Heagney (CpE)	Doran Senior (CpE)
PCB	Christina Heagney (CpE)	Doran Senior (CpE)

8. Project Summary

Our team's motivation and goals served as the primary factor in envisioning the SmartKart shopping system, while the requirements and specifications that were shaped in our design process helped bring the project to life.

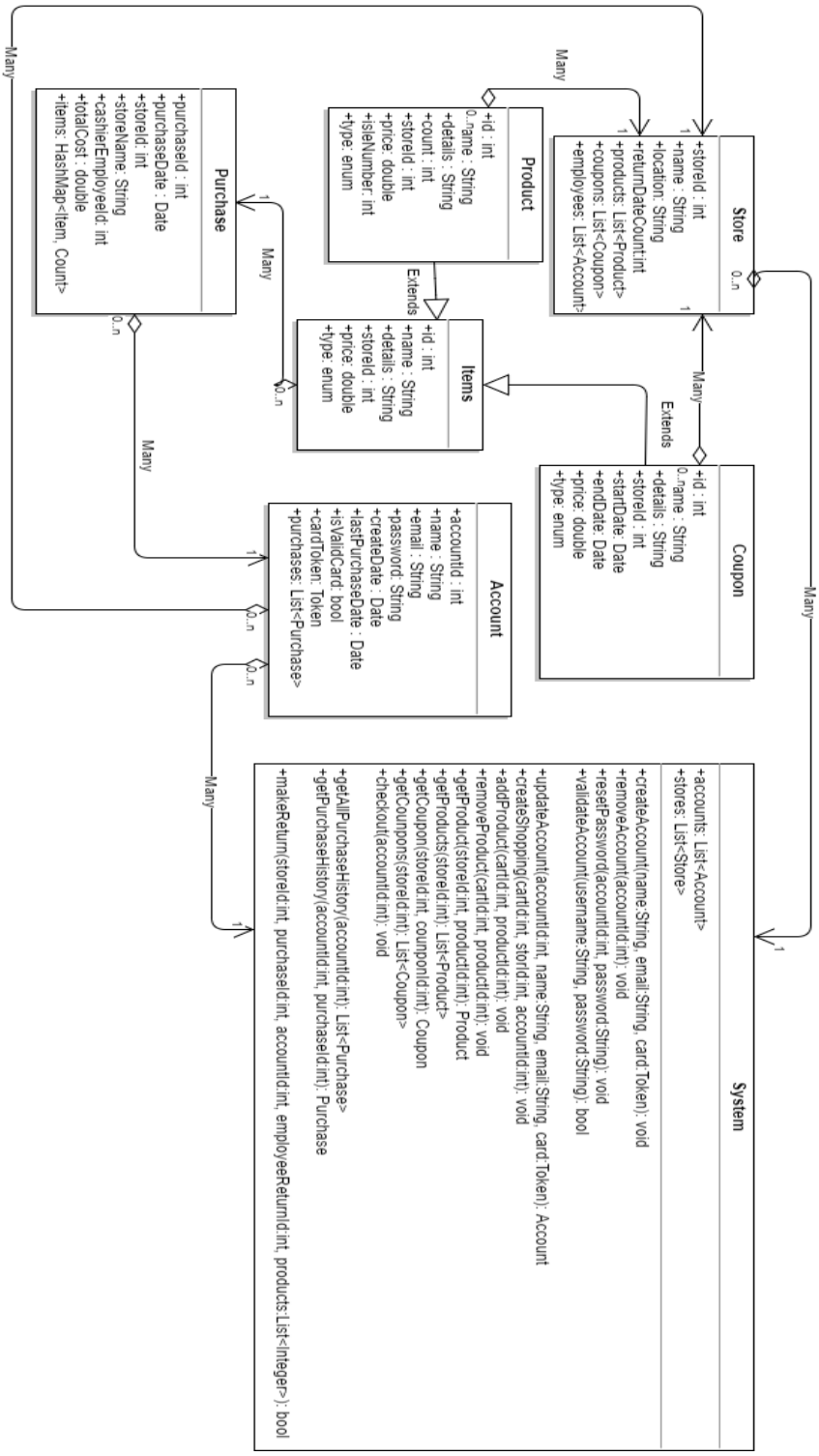
The SmartKart's infrastructure is made up of a multitude of different components and modules, including technology such as wireless communication, GPS, and RFID sensors. The integration of these modules into a single, uniform system will allow for the creation of a useful and marketable product for the retail industry and its consumers.

Our team has taken extra measures to ensure that the user's safety is our top priority. This includes building a hardware system that is safe to integrate into shopping carts, as well as ensuring the software portion of our system is safe for shoppers in regards to the security of their personal information, such as their credit card numbers. The other top concern was the cost of the system; our team was determined to create this product on a reasonable budget, in order to help ensure its marketability potential in the industry.

The SmartKart shopping system was designed with the skills of the team members in mind, and thus, the workload has been equally distributed among the members of the computer science and computer engineering disciplines. The hardware design aspects of the project were largely handled by the computer engineering students, whereas the software design was primarily handled by the computer science students. This workload division allowed for the team members to propose unique ideas, and offer different technical skills and perspectives in regards to the design and implementation of the overall project.

9. Appendices

9.1. UML Diagram



9.2. References

[1] "Amazon DynamoDB – NoSQL Cloud Database Service." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/dynamodb/?hp=tile&so-exp=below>.

[2] "Amazon Relational Database Service (RDS) – AWS." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/rds/>.

[3] "App Engine - Build Scalable Web & Mobile Backends in Any Language | Google Cloud Platform." *Google*. Google, n.d. Web. 10 Apr. 2017. <https://cloud.google.com/appengine/>.

[4] "AWS IoT - Amazon Web Services." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/iot/>.

[5] "AWS Lambda | Product Details." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/lambda/details/>.

[6] *C Coding Standard*. N.p., n.d. Web. 10 Apr. 2017. <https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>.

[7] "Cloud Security – Amazon Web Services (AWS)." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/security/>.

[8] "Evaluating the Different Types of DBMS Products." *SearchDataManagement*. N.p., n.d. Web. 10 Apr. 2017. <http://searchdatamanagement.techtarget.com/feature/Evaluating-the-different-types-of-DBMS-products>.

[9] *Google Cloud Computing, Hosting Services & APIs | Google Cloud Platform*. Google, n.d. Web. 10 Apr. 2017. <https://cloud.google.com/>.

[10] *Google Cloud Computing, Hosting Services & APIs | Google Cloud Platform*. Google, n.d. Web. 10 Apr. 2017. <https://cloud.google.com/?utm_source>.

[11] *Google Infrastructure Security Design Overview | Solutions | Google Cloud Platform*. Google, n.d. Web. 10 Apr. 2017. <https://cloud.google.com/security/security-design/>.

[12] McMahon, Caolan. *Node.js: Style and Structure*. N.p., n.d. Web. 10 Apr. 2017. <https://caolan.org/posts/nodejs_style_and_structure.html>.

[13] "NoSQL Databases Explained." *MongoDB*. N.p., n.d. Web. 10 Apr. 2017. <https://www.mongodb.com/nosql-explained>.

[14] *Version Control Concepts and Best Practices*. N.p., n.d. Web. 10 Apr. 2017. <https://homes.cs.washington.edu/~mernst/advice/version-control.html#best-practices>.

[15] "What Is Amazon Web Services (AWS)? - Definition from WhatIs.com." *WhatIs.com*. N.p., n.d. Web. 10 Apr. 2017. <http://whatis.techtarget.com/definition/Amazon-Web-Services-AWS>.

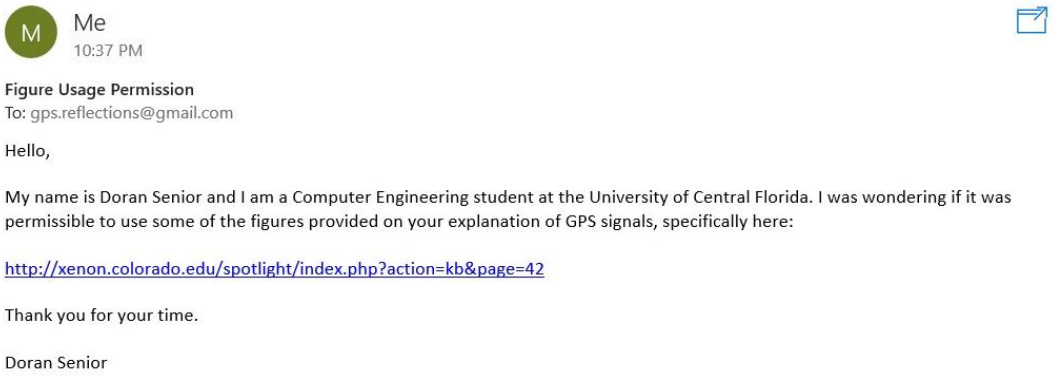
[16] "What Is Cloud Computing? - Amazon Web Services." *Amazon Web Services, Inc.* N.p., n.d. Web. 10 Apr. 2017. <https://aws.amazon.com/what-is-cloud-computing/?sc_channel=PS&sc_campaign=acquisition_US&sc_publisher=google&sc_medium=cloud_computing_b&sc_content=sitelink&sc_detail=amazon%2Bwebservices&sc_category=cloud_computing&sc_segment=what_is_cloud_computing&sc_matchtype=p&sc_country=US&s_kwcid=AL%214422%21

3%21175074967375%21p%21%21g%21%21amazon%2Bwebservices&ef_id=V0PHGAAABBS0lBcb%3A20170324171055%3As>.
[17] "What Is NoSQL (Not Only SQL Database)? - Definition from WhatIs.com." *SearchDataManagement*. N.p., n.d. Web. 10 Apr. 2017. <http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL>.
[18] "What Is Relational Database Management System (RDBMS)? - Definition from WhatIs.com." *SearchSQLServer*. N.p., n.d. Web. 10 Apr. 2017. <http://searchsqlserver.techtarget.com/definition/relational-database-management-system>.
[19] Sam Davis, Editor-in-Chief | May 24, 2013. "Back to Basics: Voltage Regulator ICs, Part 1."Power Electronics. N.p., 10 June 2013. Web. 24 Apr. 2017. <http://powerelectronics.com/regulators/back-basics-voltage-regulator-ics-part-1>
[20] Understanding Boost Power Stages in Switchmode Power Supplies (n.d.): n. pag. Ti.com. Texas Instruments. Web. 20 Mar. 2017. <http://www.ti.com/lit/an/slva061/slva061.pdf>.
[21] Zhang, Henry J. "Basic Concepts of Linear Regulator and Switching Mode Power Supplies."Cds.linear.com. Linear Technology, Oct. 2013. Web. 20 Mar. 2017. <http://cds.linear.com/docs/en/application-note/AN140fa.pdf+>.
[22] "Linear and Switching Voltage Regulator Fundamental Part 1." Ti.com. Texas Instruments, n.d. Web. 20 Mar. 2017. <http://www.ti.com/lit/an/slva061/slva061.pdf+>
[23] Johnson, Dana J. "Overcoming Challenges to Transformational Space Programs: The Global Positioning System (GPS)." Northropgrumman.com. Northrop Grumman, Oct. 2006. Web. 20 Mar. 2017. <http://www.northropgrumman.com/AboutUs/AnalysisCenter/Documents/pdfs/Overcoming-Challenges-to-Trans.pdf+>.
[24] Marshall Brain & Tom Harris "How GPS Receivers Work" 25 September 2006.
[25] HowStuffWorks.com. <http://electronics.howstuffworks.com/gadgets/travel/gps.htm> 25 April 2017
[26] Pugh, Evan. "Decoding the GPS Signal." Xenon.colorado.edu. GPS Reflections Research Group, n.d. Web. 20 Mar. 2017. <http://xenon.colorado.edu/spotlight/index.php?action=kb&page=42>.
[27] "NEO-6 U-blox 6 GPS Modules Data Sheet." NEO-6 (n.d.): n. pag. U-blox.com. Web. 20 Mar. 2017. <NEO-6 u-blox 6 GPS Modules Data Sheet>.
[28] Dahl, Øyvind Nydal. "Microcontroller Tutorial 3/5: How To Design the Circuit." Build Electronic Circuits. Microcontroller Tutorial 3/5: How To Design a Microcontroller Circuit, 09 Oct. 2015. Web. 20 Mar. 2017. <http://www.build-electronic-circuits.com/microcontroller-tutorial-part3/>.
[29] "1187-2002 - IEEE Recommended Practice for Installation Design and Installation of Valve-Regulated Lead-Acid Batteries for Stationary Applications." IEEE SA - 1187-2002 - IEEE Recommended Practice for Installation Design and Installation of Valve-Regulated Lead-Acid Batteries for

Stationary Applications. IEEE, 01 Aug. 2002. Web. 30 Mar. 2017. <https://standards.ieee.org/findstds/standard/1187-2002.html>.
[30] "PN532/C1 NFC Controller." (n.d.): n. pag. Cdn-shop.adafruit.com. Adafruit, 31 Mar. 2011. Web. 30 Mar. 2017. <https://cdn-shop.adafruit.com/datasheets/pn532ds.pdf>.
[31] (n.d.): n. pag. Atmel.com. Atmel Corporation. Web. 30 Mar. 2017. <http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>.
[32] (n.d.): n. pag. Sparkfun.com. Texas Instruments, May 2003. Web. 30 Mar. 2017. <https://www.sparkfun.com/datasheets/Components/LM7805.pdf>.
[33] "LMZ12010 10-A SIMPLE SWITCHER® Power Module With 20-V Maximum Input Voltage." 10.1 (1986): 118. Ti.com. Texas Instruments, Aug. 2015. Web. 25 Apr. 2017. <http://www.ti.com/lit/ds/symlink/lmz12010.pdf>.
[34] *AngularJS - Superheroic JavaScript MVW Framework*. N.p., n.d. Web. 24 Apr. 2017.
[35] "Application Fundamentals." *Android Developers*. N.p., n.d. Web. 24 Apr. 2017.
[36] "Documentation." *Stripe*. N.p., n.d. Web. 24 Apr. 2017.
[37] ECollege, Todd Fredrich Pearson. "REST API Quick Tips." *RESTful Services Quick Tips*. N.p., n.d. Web. 24 Apr. 2017.
[38] *Express - Node.js Web Application Framework*. N.p., n.d. Web. 24 Apr. 2017.
[39] Foundation, Node.js. "About Node.js®." *About | Node.js*. N.p., n.d. Web. 24 Apr. 2017.
[40] Kaazing. *HTML5 WebSocket - A Quantum Leap in Scalability for the Web*. N.p., n.d. Web. 24 Apr. 2017.
[41] "Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards." *PCI Security Standards Council®*. N.p., n.d. Web. 24 Apr. 2017.
[42] Otwell, Taylor. "Installation." *Installation - Laravel - The PHP Framework For Web Artisans*. N.p., n.d. Web. 24 Apr. 2017.
[43] "Overview." *Architectural Overview of Cordova Platform - Apache Cordova*. N.p., n.d. Web. 24 Apr. 2017.
[44] "Representational State Transfer." *Wikipedia*. Wikimedia Foundation, 20 Apr. 2017. Web. 24 Apr. 2017.
[45] *Socket.IO - Docs*. N.p., n.d. Web. 24 Apr. 2017.
[46] "Top 10 2013-Top 10." *Top 10 2013-Top 10 - OWASP*. N.p., n.d. Web. 24 Apr. 2017.
[47] Trachier, Kevin. "API Basics: What Is SOAP?" *API Basics: What Is SOAP? | SoftLayer Blog*. N.p., n.d. Web. 24 Apr. 2017.
[48] "Web | Google Developers." *Google*. Google, n.d. Web. 24 Apr. 2017.
[49] Digital image. N.p., n.d. Web. 24 Apr. 2017. <https://cordova.apache.org/static/img/guide/cordovaapparchitecture.png>.
[50] Digital image. N.p., n.d. Web. 24 Apr. 2017. <https://angular.io/resources/images/devguide/architecture/overview2.png>.

[51] Mitchell, Bradley. "What's 802.11? What These Wireless Standards Mean." *Lifewire*. N.p., 23 Feb. 2017. Web. 25 Apr. 2017. <https://www.lifewire.com/wireless-standards-802-11a-802-11b-g-n-and-802-11ac-816553>.
[52] "How does an Electromagnetic Lock Operate?" *YLI Electronic*. N.p., n.d. Web. 25 Apr. 2017. <http://www.yli.cn/en/customer/FAQ/2013-02-01/474.html>.
[53] Solutions, LLC ITech. "Gatekeeper Systems, Inc." *Gatekeeper Systems, Inc. | Solutions | CartControl® Shopping Cart Retention FAQ's*. N.p., n.d. Web. 25 Apr. 2017.
[54] Carttronics. "Innovate. save. profit.™." *Carttronics*. N.p., n.d. Web. 25 Apr. 2017. <https://www.carttronics.com/CAPS.asp>.
[55] Industries, Adafruit. "Adafruit Bluefruit LE SPI Friend - Bluetooth Low Energy (BLE)." *Adafruit industries blog RSS*. N.p., n.d. Web. 25 Apr. 2017.
[56] "BlueTooth-HC05-HC06-Modules-How-To." *Arduino-info*. N.p., n.d. Web. 25 Apr. 2017. <https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To>.
[57] Industries, Adafruit. "Adafruit ATWINC1500 WiFi Breakout." *Adafruit industries blog RSS*. N.p., n.d. Web. 25 Apr. 2017.
[58] "WiFi Module - ESP8266." *WRL-13678 - SparkFun Electronics*. N.p., n.d. Web. 25 Apr. 2017. <https://www.sparkfun.com/products/13678>.

9.3. Copyright Permissions



M Me
10:37 PM

Figure Usage Permission
To: gps.reflections@gmail.com

Hello,

My name is Doran Senior and I am a Computer Engineering student at the University of Central Florida. I was wondering if it was permissible to use some of the figures provided on your explanation of GPS signals, specifically here:

http://xenon.colorado.edu/spotlight/index.php?action=kb&page=42

Thank you for your time.

Doran Senior

M Me
11:00 PM

Figure Usage Permissions
To: ti-cares@ti.com

Hello,

My name is Doran Senior and I am a Computer Engineering major at the University of Central Florida. I was wondering if it was permissible to use some of the figures on voltage regulators in a paper I am writing for school. The links that contain these figures can be seen here:

http://www.ti.com/lit/an/snva558/snva558.pdf

http://www.ti.com/lit/an/slva061/slva061.pdf

Thank you for your time.

Doran Senior

M Me
11:09 PM

Figure Usage Permissions
To: Sam.Davis@penton.com

Hello,

My name is Doran Senior and I am a Computer Engineering major at the University of Central Florida. I was wondering if it was permissible to use some of the figures on voltage regulators in a paper I am writing for school. The links that contain these figures can be seen here:

http://www.powerelectronics.com/regulators/back-basics-voltage-regulator-ics-part-1

Thank you for your time.

Doran Senior

CH Christina Heagney
Today, 2:40 PM
customerservice@sparkfun.com

Reply all

Hello,

My name is Christina Heagney and I am an undergraduate computer engineering student currently enrolled at the University of Central Florida. I would like to request permission to use some of the images on your website for a research paper my group is writing. The links containing the images are listed below:

https://www.sparkfun.com/products/13678
https://www.sparkfun.com/products/11862

Thank you for your time,
Christina Heagney

CH Christina Heagney
Today, 2:37 PM
support@adafruit.com

Reply all

Hello,

My name is Christina Heagney and I am an undergraduate computer engineering student currently enrolled at the University of Central Florida. I would like to request permission to use some of the images on your website for a research paper my group is writing. The links containing the images are listed below:

https://www.adafruit.com/product/2633
https://www.adafruit.com/product/2999
https://www.adafruit.com/product/364

Thank you for your time,
Christina Heagney