RewardWallet

School of Engineering Science | Burnaby, BC · V5A 1S6
https://github.com/RewardWallet

February 21, 2018

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 405W Functional Specification for a Customizable Rewards Allocation System

Dear Dr. Rawicz:

Attached you will find our ENSC 405W functional proposal for a Customizable Rewards Allocation System. This project is designed to add reward allocation to any business by integrating their POS system into our cloud system. This easy and affordable solution is poised to give small Canadian businesses the tools needed to propel their businesses into the digital space.

These functional specifications detail the core architectural requirements that will be demonstrated in the prototype. They will also serve as the bedrock foundation that can be refined and built upon for production models. It is from these requirements that our engineers will be able to follow during development

RewardWallet is now made up of three undergraduate engineering students: Molly Bin, Nathan Tannar, Mandy Xiao. Wilson Chen has since left the company. Should you have any questions or concerns, please feel free to reach out to myself at (604) 355-6292 or ntannar@sfu.ca

Sincerely,

Nathan Tannar

President and CEO
RewardWallet

Enclosure: *Functional Specification for a Dynamic Reward Allocating POS Add-On*

Functional Specification for the

CUSTOMIZABLE REWARDS ALLOCATION SYSTEM

Project Team:

Molly Bin
Nathan Tannar
Jia Hui (Mandy) Xiao

Contact Person:

Nathan Tannar
ntannar@sfu.ca

Submitted to:

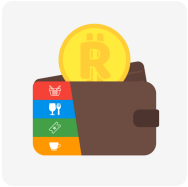
Dr. Andrew Rawicz - ENSC 405W
Steve Whitmore - ENSC 405W
School of Engineering Science
Simon Fraser University

Issued date:

February 21, 2018

Revision:

1.0



RewardWallet

Functional Specification for the Customizable Rewards Allocation System

Executive Summary

In the modern world businesses must move quickly if they want to get an edge or keep up with the competition. Loyalty programs show no signs of slowing down. “The multi-billion-dollar business is just as popular as ever with Canadian consumers and companies alike” [1]. However, most businesses don’t have a loyalty program of their own. A common setback that prevents a business from implementing their own digital loyalty system is the cost.

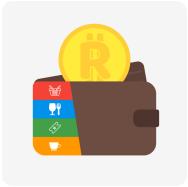
Hiring a development agency or freelancers to build a custom app can reach $100,000 [2] plus maintenance costs. This results in owners either opting out of offering a loyalty program or reverting to physical card with special stamps. To help smaller Canadian businesses keep up with larger corporations we have designed an end-to-end solution - RewardWallet. This easy to use solution will not require any maintenance or technical knowledge as it will all be covered under the businesses’ monthly subscription fee.

The development of the RewardWallet system will be split into three subsystems. Each will have completed with unit testing before system wide integration testing takes place. The three subsystems are:

- Embedded System (RewardBeamer)
- Cloud Server and API
- Native iOS application

During the first phase of development the minimum operational requirements will be implemented. These requirements, detailed later in this document, will form the bedrock of the systems architecture and allow for future features and advancements to be built on top of it. The 10-week development cycle will conclude with an operational prototype that can process a transaction for a businesses and correctly allocating reward points to the user’s account.

After the prototype is developed the system can be modelled to businesses where further market research can take place to determine if there are any must-have features that should be implemented. The second phase of development is where these requested features will be added along with a set of features currently being planned, to have a competitive edge over the competition. These features and the design of the system will conform to any pertinent standards or guidelines set by the industry.



RewardWallet

Functional Specification for the Customizable Rewards Allocation System

1 Table of Contents

2 Introduction

RewardWallet is a system that keeps track of transactions users make at a business to allocate rewards to a customer's profile based on how the business customizes their distribution model. Each customer will have a unique digital card for each business which stores the reward points. By creating the hardware needed by a business, a mobile app for the user and a cloud system to bring it all together, RewardWallet gives business owners an end-to-end solution for adding a customizable digital loyalty system to their store. The requirements for RewardWallet, as previously proposed, are detailed in this functional specification document.

2.1 Scope

This document outlines the functional specifications to the components that make up the RewardWallet system. The specifications go as far as to outline the base architecture of the system that will be captured in the initial development phase as a prototype. The listed requirements will later drive future improvements that can be made possible for the production model.

2.2 Intended Audience

This functional specification was written with the intent of being read by all members of RewardWallet. Product Managers will be able to use these specifications as a reference to the overall progress made by the developers. Test Engineers will be use them when defining test cases to ensure each subsystem meets reliability and performance requirements. Designers can use these specifications to gauge a sense of how data will flow between systems and how the user interface should be designed around that. Lastly, business owners can refer to this document as an aid should they ever need it.

2.3 Classification

Throughout this document, the following convention will be used to denote a functional requirement:

[R-n] p

Where n is the numerical representation of said requirement and p is the priority level of the requirement. The priority levels are:

I. Required for core architecture; modeled in the prototyped scheduled for April 2018
II. Required for production; modeled in the pre-production model scheduled for August 2018
III. Required for post-production feature add-ons as software updates

3 System Requirements

General requirements for how the different layers of the system, such as the POS Add-On, cloud server and mobile app, should operate in parallel to complete the design.

3.1 System Overview

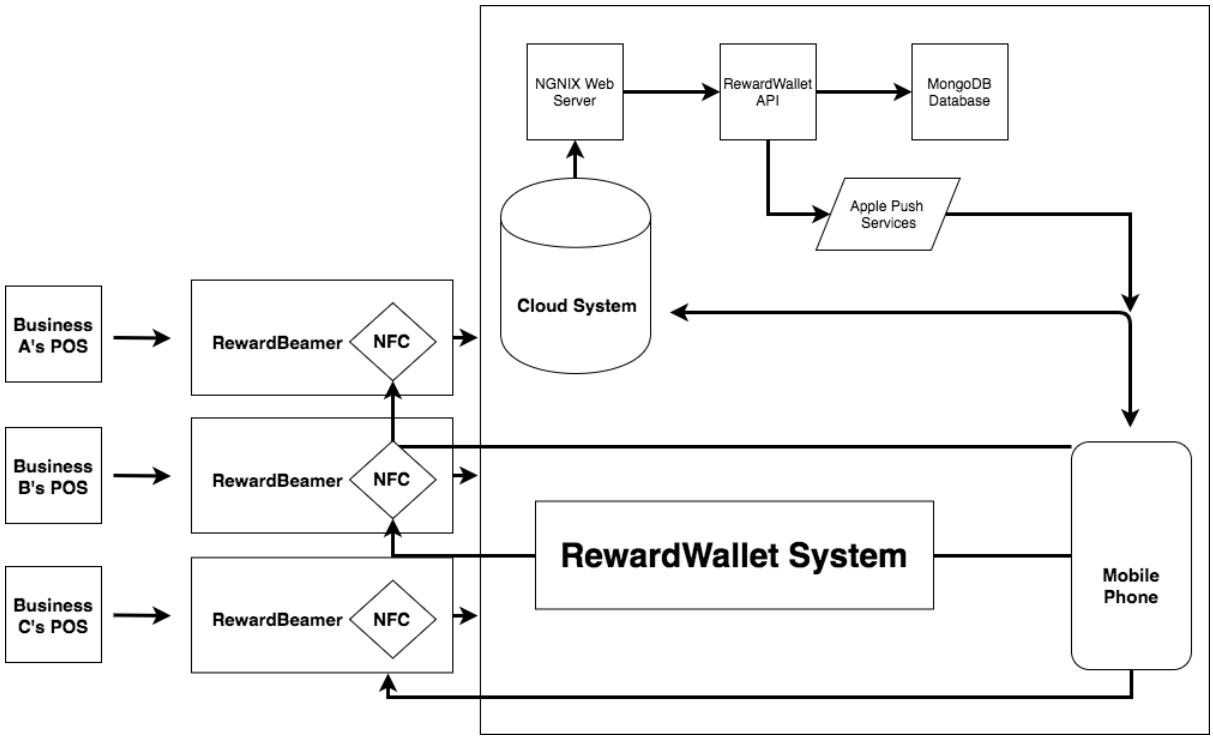


Figure 1: High Level RewardWallet System Model

Each sub component will need to be developed and tested to complete a working prototype of the design. Due to time constraints and available funding, efforts will be prioritized on establishing the core architecture of the systems requirements detailed in the next sections. Once the necessary protocols are in place with the prototype future work could be done on each sub component to increase features, usability and performance.

Reliable and secure communication between the subcomponents is essential and a top priority. Transaction records, personal data and reward point balance integrity is sensitive data which should not, under any circumstance, be viewed or edited by a third party. If the system were to become compromise it would put businesses at risk to redeem points that were not obtained legitimately. Customer transaction data could also

be used maliciously if ever in the wrong hands. These concerns require that appropriate protocols be put in place to protect RewardWallet, the businesses and the customers.

Most importantly, the system should be dynamic and scalable to any business. To make this reliable a plethora of unit tests need to be written for each module. This modularity of the system makes it easier to test and isolate issues while also giving RewardWallet a competitive edge over its competitors.

3.2 General Requirements

[R3.2.1-i] Each subcomponent must be developed as a “black box” that can be tested independently from the overall system.

[R3.2.2-i] The monthly operational cost of the system shall remain under $20 per 1000 daily active users.

[R3.2.3-iii] The system must be able to integrate with an existing POS system.

[R3.2.4-ii] The system shall be compatible with any mobile phone that has NFC.

3.3 Standards

[R3.3.1-i] Data transfer between sub components shall conform to RESTful design.

[R3.3.2-i] Data transfer between sub components shall be encrypted to preserve its integrity.

[R3.3.3-i] NFC shall conform to the ISO/IEC 18000-3 standards at 13.56MHz.

4 Embedded POS Add-On Requirements

These are the requirements of the embedded system, herby named RewardBeamer, that will serve as an add-on to a business's existing POS landscape. By having a small device that can accept HTTP POST requests from the POS system it can act as a bridge between a business and RewardWallet's cloud server. RewardBeamer will be built on a Raspberry Pi microcontroller as a system that can perform network requests is required.

4.1 System Overview

At the high level, the RewardBeamer is a finite state machine that can be triggered to request a reward be allocated to the customer based on the businesses distribution model.

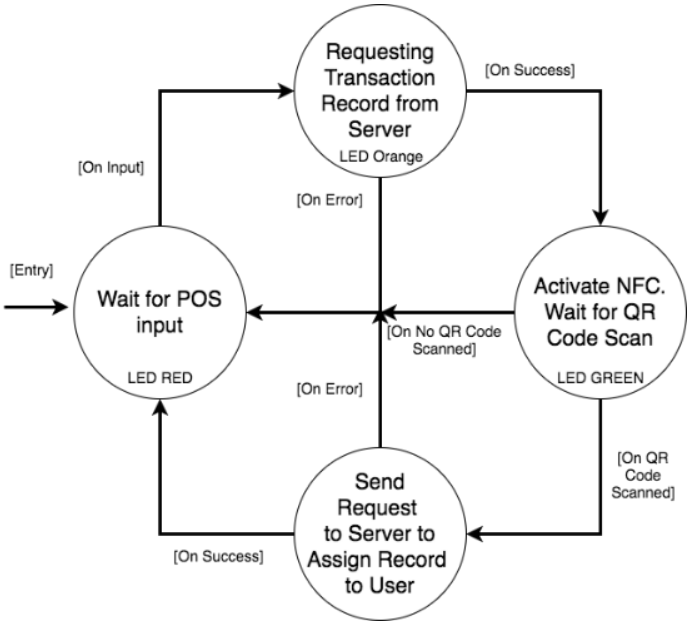


Figure 2: High Level RewardBeamer System Model

This model shows that if a user does not have a phone with NFC it can alternatively allocate rewards through the use of a QR code. While we have thought through the design on how this would work, due to time constraints it will not be implemented as a feature in the prototype.

4.2 General requirements

[R4.2.1-i] The device shall be able to receive a POST message to initiate a transaction.

[R4.2.2-i] The device shall be able to transmit records to a receiving mobile phone through NFC.

[R4.2.3-ii] The device shall be able to retrieve records from a sending phone through a QR code scan.

[R4.2.4-ii] The device shall have a manufacturing cost of less than $50.

[R4.2.5-i] The device shall be powered through micro USB, allowing for portability with batteries.

[R4.2.6-i] The device must consume a 5V power supply to operate.

[R4.2.7-ii] The device shall have an LED light to indicate states.

[R4.2.8-i] The device shall be programmed on a Raspberry Pi board.

[R4.2.9-i] The device must not be become stuck in a particular state or loop.

[R4.2.10-i] The device shall have a codebase written primarily in Python and JavaScript.

[R4.2.11-i] The device must not incorrectly allocate rewards.

4.3 Security Requirements

[R4.3.1-ii] The device shall take precautions to prevent fraud.

[R4.3.2-ii] The device must be able to prevent unauthorized access.

[R4.3.3-i] The device must be able to connect to the cloud server securely.

4.4 Standards

[R4.4.1-i] The device will comply with IEEE 802.11 b/g/n wireless LAN standards.

[R4.4.2-i] The device will support Bluetooth 4.1 and Bluetooth Low Energy (BLE).

4.5 Reliability Requirements

[R4.5.1-i] The device must be able to resume operation in the event of a failure.

[R4.5.2-i] The device must be able to operate continuously for up to 24 hours.

4.6 Performance Requirements

[R4.6.1-i] The device shall be able to process one transaction at a time.

[R4.6.2-i] The device shall operate and perform in real time.

[R4.6.3-i] The device must timeout after 20 seconds after beginning a NFC record transfer.

4.7 Usability Requirements

[R4.7.1-ii] The device must be easy to set up to begin operation.

[R4.7.2-ii] Once set up, the device shall only need to be given a power source to resume operations under the same configuration.

[R4.7.3-i] The device shall require a data network in to operate and make HTTP requests.

[R4.7.4-ii] The device will be configurable over a local network.

5 Cloud Server Requirements

5.1 System Overview

The Cloud Server manages shared resources between the Embedded System and the IOS application. It services requests sent by the Embedded System and respond upon completion [8]. Figure 3 shows the architecture of the Cloud Server with respect to other components within the system. Various business clients with their own business models will send their requests to the server. Then, the server will respond by executing those requests based on the unique business models. Appropriate amounts of rewards points are then allocated to customers.

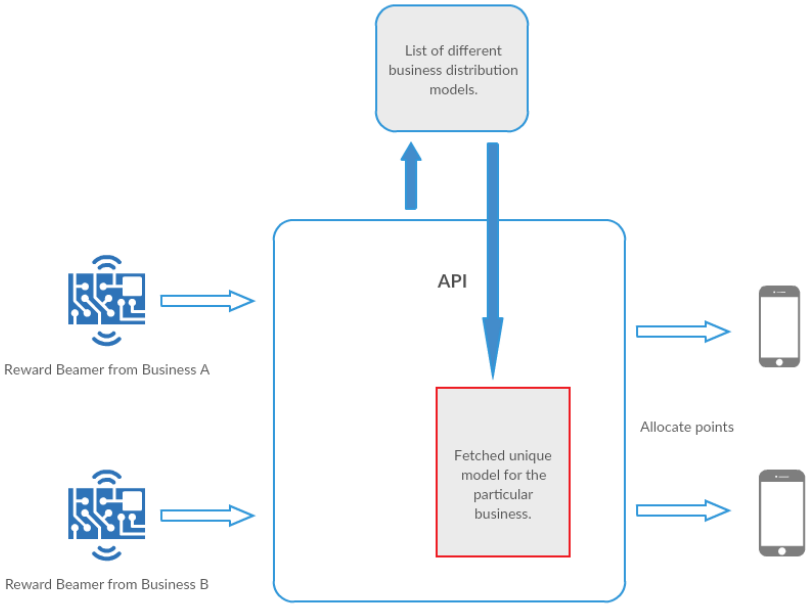


Figure 3: Cloud Server architecture

5.2 General Requirements

[R5.2.1-i] The Cloud shall be able to receive and handle requests from phones and Embedded Systems.

[R5.2.2-iii]The Cloud enables business clients to manipulate their customers' data.

[R5.2.3-i] The Cloud allows customers to redeem their rewards collected at different vendors.

[R5.2.4-i] The Cloud is integrated with MongoDB Database for storing client data.

[R5.2.5-i] The Could shall be able to store transaction records sent from the Embedded System.

[R5.2.6-i] The Cloud shall be able to distribute the appropriate rewards to customers following a successful transaction.

[R5.2.7-i] The Cloud services requests via the RewardWallet API.

5.3 Security Requirements

[R5.3.1-iii] The Cloud implements different access permissions for different users.

[R5.3.2-i] The Cloud requires user authentication upon attempted access.

[R5.3.3-i] The Cloud shall be able to control access through configurable security controls.

5.4 Reliability Requirements

[R5.4.1-i] The Cloud shall only complete the allocation of customer rewards upon matched transaction records.

[R5.4.2-i] The Cloud shall be able to reject non-valid or incomplete transactions.

[R5.4.3-iii] The Cloud shall have recovery protocols in place in an event of server crash.

[R5.4.4-ii] In case of heavy incoming requests, the Cloud responses shall be delayed but does not fail.

5.5 Performance Requirements

[R5.5.1-i] The Cloud shall be able to receive transaction records from the Embedded System in real-time.

[R5.5.2-i] The Cloud shall be able to create verified transaction records instantaneously.

[R5.5.3-i] Subjected to the network connection, the Cloud shall be able to allocate rewards to customers' phones within seconds.

[R5.5.4-ii] The Cloud shall be able to match available resource to incoming requests.

[R5.5.5-iii] The Cloud utilizes asynchronous architecture to achieve high-performance [9].

[R5.5.6-i] The Cloud requires data network for receiving requests and serving data.

5.6 Usability Requirements

[R5.6.1-i] The Cloud shall have a visual dashboard to modify data records

6 Mobile App Requirements

6.1 System Overview

This is a high-level system overview for iOS app and doesn't represent the final structure of the system. It is merely a starting point and will be adjusted as seen fit during the development of the project.

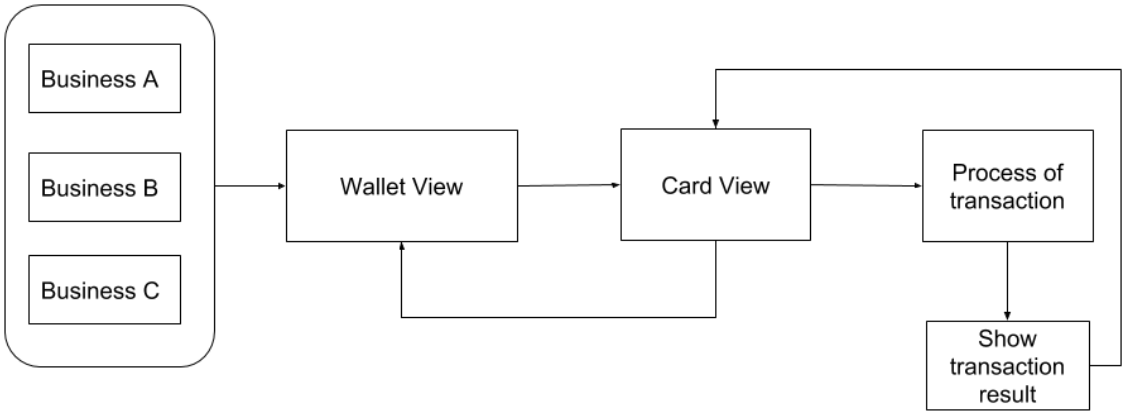


Figure 4: High Level IOS App architecture

6.2 General Requirements

[R6.2.1-i] All development of the iOS application shall be done with IDE XCode version 9.1.1 or newer on a Mac OS X machine.

[R6.2.2-i] The application shall be coded with Apple's latest and greatest Swift programming language.

[R6.2.3-ii] The application shall only be compatible with iPhone 6, iPhone 6 Plus, iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, and iPhone X. All devices require an active internet condition.

[R6.2.4-ii] The application shall only be compatible with devices on iOS 10 or later

6.3 Safety and Security Requirements

[R6.3.1-ii] The app shall not have any malicious code embedded or programmed in such a way that can be used maliciously [3].

[R6.3.2-i] The app shall not include content that is offensive, inappropriate, and abusive [3].

[R6.3.3-i] An encrypted file system that can be enabled to prevent data on lost [3].

6.4 Standards

[R6.4.1-i] The app must follow the guiding principle of the App Store to provide a safe experience for users to use. User interface of the app shall follow iOS Human interface guidelines [4].

6.5 Reliability Requirements

[R6.5.1-ii] The app must automatically inform developers in the event of a crash.

6.6 Performance Requirements

[R6.6.1-i] The app shall use power efficiently or generate excessive heat [3].

[R6.6.2-i] The app will request a reward in the background and then update in real time.

[R6.6.3-ii] The app must have smooth animation transitions.

6.7 Usability Requirements

[R6.7.1-i] The app shall provide register page for users to sign up.

[R6.7.2-i] The app shall be able to show all the company cards in Wallet View after users sign in.

[R6.7.3-i] The app shall be able to read NFC data.

[R6.7.4-i] The app shall be able to send requests to cloud system to assign record to users.

[R6.7.5-i] The app shall be able to process transaction in Card View.

[R6.7.6-i] The app shall be able to show the record of the transaction.

[R6.7.7-ii] The app shall be able to generate QR code for scanning.

[R6.7.8-iii] The app shall have push notification sent from businesses to inform promotions.

[R6.7.9-i] The app shall be able to show the total reward for each company card.

[R6.7.10-iii] The app shall be able to perform mobile ordering.

7 SUSTAINABILITY SAFETY

In addition to primary goals determined for this project, RewardWallet dedicates to develop a sustainable product. The prototype and final product produced by RewardWallet aim to have zero waste as required by Cradle to Cradle Design [5]. The reuse helps us to minimize the impact of the material on the environment after the life cycle is over. After producing our first prototype, we will assess the rest of the components and split into two categories: components that can be reused for final product and components that are required to be recycled. Reuse of wires is a feasible approach in Cradle to Cradle Design that could have less negative impacts on the environment. We will accept any RewardBeamer system components that fail or break so that they do not end up in the landfill. RewardWallet will also use this strategy after developing our final product.

On the other hand, our intention is to prevent critical safety risks to end user. To achieve this goal, we focus on minimize the harmful impact of the product by mounting no sharp edges on the RewardBeamer. The hardware components shall be secured properly to prevent loose components. RewardWallet will also ensure the voltage that used for powered device follows Electrical Utility Safety Rules [6].

One of the benefits of using a Raspberry Pi microcontroller is it can be easily programmed or repurposed for a new cause in the event that a business chooses to cancel their RewardWallet subscription. Not only can the Raspberry Pi be factory reset and given to a new business, but it could also be donated to one of many charitable organizations that use Raspberry Pi’s to teach programming to young students; thus, preventing the microcontroller from being sent to any landfill.

Moreover, the chosen of cloud server is also a green solution. From a resource efficiency perspective, less equipment is needed to run workloads, which proactively reduce data center space and eventual e-waste footprint. The pay-per-user of cloud based encourage user to only consume when they need. Renewable energy resource, design for environmental policies, and environmentally sound end-of-life are three dimensions that will maximize the contribution of cloud computing [7].

8 System Test Plan

8.1 Embedded System

Initial testing of the embedded system will primarily focus on stress testing. While there will also be some basic unit testing to simulate responses from the Cloud Server and Mobile App, a focus will be put on reliability. It is important that under heavy and prolonged use the system does not fail due memory overflow, system overheating or thread locking. By building scripts that will cause the system to run indefinitely in a loop these peak use cases can be analyzed to determine if there are any potential problems. Once reliability has been established the focus can be shifted to performance. Using real responses from the Cloud Server and Mobile App integration testing can begin to test for any timeout issues, failed network requests or failed NFC data transfers.

When we have ensured reliability and performance of the system and corrected any bugs the last step is to test the system with real users. It is important to test the usability to see if we have made any incorrect user experience or interface assumptions. The goal would be to determine if a user can accurately and effectively complete the action of tapping their phone on the embedded system to collect a reward. We hope that users will walk away enjoying the simplicity and satisfaction with our product.

8.2 Cloud Server

All units of the Could Server will be tested before conducting integration testing. They include stable connections to the NGNIX Web Server, successful executions of the RewardWallet API, and maintainability of the MongoDB Database. Then, test clients will be created to further test the communication between clients and server. A test business client can request the creation of a transaction record from the Embedded System after a purchase is made. The Cloud Server will serve such request and send the transaction record reference back. At this point, developers are able to verify on the server that the correct transaction record is made. Communication between mobile user and the server can then be tested by verifying that the same record match the one that is assigned to the mobile user.

To guarantee that the Could Server maintains functional under heavy load, load tests are to be performed on the production version of the Cloud Server. Automated test scripts will be written for the purpose of mimicking multiple business and customer clients. Multiple API requests are sent to the server. Tests will expose any issues related to create, update and read operations. Appropriate fixes will then be performed to ensure efficiency of the Cloud Server.

8.3 Mobile App

XCode features a built-in unit test framework since the release of XCode 9 and latest Swift. Unit testing will ensure that every single API in the app's ecosystem will pass under every possible scenario without crashing. It will also ensure that all the features listed in the above requirement are working as expected and allow us to validate our method and make any adjustments if necessary. Unit testing will also ensure that all of the functions will work fine separately from each other before integrating the whole system. Unit testing will be performed throughout the entire development process whenever a new function is added to the project to ensure that integration of new functionality breaks nothing.

For our mobile app to be as user friendly as possible, we plan to have user acceptance testing for our final product. Users will make some purchases with our RewardWallet solution and earn rewards for each transaction in different business. Users will verify the transaction is made correctly and the reward points shown on the mobile app are matched with the record in the cloud server.

9 CONCLUSION

In conclusion, the RewardWallet system is composed of three main subsystems. Namely, the Embedded System, Cloud Server and Mobile Application. Each subsystem is comprised of their component-specific requirements. The system emphasizes on the functionality that it is dynamic and scalable to any business.

As aforementioned, communication between subsystems is important to the success of the RewardWallet system. The Embedded System is equipped with the capabilities to perform network requests for initiating transaction and transmitting records through NFC and QR code. The Cloud Server not only enables business clients to execute their customized distribution models, but also allocate appropriate amounts of reward points to customers. The Mobile Application elegantly displays the custom reward points from different company cards. Security requirements are also discussed in detail as the system contains highly sensitive information for both the business clients and their customers.

For clarity of the functional specifications for different stages of the project, each requirement is identified with priority levels. Core requirements are to be completed for the prototype, while production requirements are to be completed for August 2018.

10 GLOSSARY

API: Application Programming Interface, is a set of instructions or protocols.

HTTPS: Hypertext Transfer Protocol for secure communication over a network.

RESTful: Representational state transfer web services architectural design protocol.

RewardBeamer: An embedded system running a small API that can be used to control reward allocation through its NFC module.

Distribution Model: The customizable model businesses can tune to determine how they wish to give rewards to customers; for example: cash back of 5% or 10 purchases for a free item.

POS: Point-of-Sale, referring to the instance a transaction takes place between a business and a customer.

Cloud Server: A logical server that is build, hosted and delivered through a cloud computing platform over the internet.

Embedded System: Combinations of hardware and software that perform a specific function within a larger system.

NFC: Near-field communication is a set of communication protocols that enable two electronic devices to establish communication by bringing them closer of each other.

Prototype: A first preliminary model of something.

QR Code: the trademark for a type of matrix barcode that is a machine-readable optical label contains information about the item.

11 Sources and References

[1] Shaw, H. (2017). Canadians just can't seem to quit loyalty cards, despite all of the data breaches and PR headaches. [online] Financial Post. Available at: http://business.financialpost.com/news/retail-marketing/canadians-just-cant-seem-to-quit-loyalty-cards-despite-all-of-the-data-breaches-and-pr-headaches [Accessed 20 Feb 2018].

[2] How Much to Make an App. (2018). How much does it cost to make an app? - App Cost Calculator. [online] Available at: http://howmuchtomakeanapp.com [Accessed 20 Feb. 2018].

[3] Developer.apple.com. (2018). App Store Review Guidelines - Apple Developer. [online] Available at: https://developer.apple.com/app-store/review/guidelines/#safety [Accessed 20 Feb. 2018].

[4] Developer.apple.com. (2018). App Review - App Store - Apple Developer. [online] Available at: https://developer.apple.com/app-store/review/ [Accessed 20 Feb. 2018].

[5] C2ccertified.org. (2017). Resources - Cradle to Cradle Products Innovation Institute. [online] Available at: http://www.c2ccertified.org/resources/collection-page/cradle-to-cradle-certified-resources-public [Accessed 20 Feb. 2018].

[6] Electrical Utility Safety Rules. (2014). [ebook] ISHA.ca, pp.1-58. Available at: https://www.ihsa.ca/PDFs/Products/Id/RB-ELEC.pdf [Accessed 20 Feb. 2018].

[7] Mines, C. (2011). Inside Green IT 4 Reasons Why Cloud Computing is Also a Green Solution. [Blog] greenbiz. Available at: https://www.greenbiz.com/blog/2011/07/27/4-reasons-why-cloud-computing-also-green-solution [Accessed 20 Feb. 2018].

[8] Nginx.com. (2018). Welcome to NGINX Wiki! | NGINX. [online] Available at: https://www.nginx.com/resources/wiki/ [Accessed 20 Feb. 2018].

[9] Slideshare.net. (2012). Asynchronous Architectures for Implementing Scalable Cloud Services -…. [online] Available at: https://www.slideshare.net/twilio/asynchronous-architectures-for-implementing-scalable-cloud-services-evan-cooke-gluecon-2012 [Accessed 20 Feb. 2018].

[10] Standards.ieee.org. (2010). IEEE 1003.13-2003 - IEEE Standard for Information Technology - Standardized Application Environment Profile (AEP) - POSIX(R) Realtime and Embedded Application Support. [online] Available at: http://standards.ieee.org/findstds/standard/1003.13-2003.html [Accessed 20 Feb. 2018].

[11] Standards.ieee.org. (2009). IEEE 1625-2008 - IEEE Standard for Rechargeable Batteries for Multi-Cell Mobile Computing Devices. [online] Available at: http://standards.ieee.org/findstds/standard/1625-2008.html [Accessed 20 Feb. 2018].

[12] Standards.ieee.org. (2017). IEEE 2410-2017 - IEEE Standard for Biometric Open Protocol. [online] Available at: http://standards.ieee.org/findstds/standard/2410-2017.html [Accessed 20 Feb. 2018].

[13] Standards.ieee.org. (2009). IEEE 802.15.5-2009 - IEEE Recommended Practice for Information technology-- Telecommunications and information exchange between systems-- Local and metropolitan area networks-- Specific requirements Part 15.5: Mesh Topology Capability in Wireless Personal Area Networks (WPANs). [online] Available at: http://standards.ieee.org/findstds/standard/802.15.5-2009.html [Accessed 20 Feb. 2018].

APPENDIX I – PRESENTATION DEMO PLAN

During the ENSC 405W poster presentation in April 2018, three different use case examples will be demoed. Each use case will simulate different businesses each with their own distribution model. This will show the robustness of our system and how it can be adapted to any business.

In each scenario, a test user will be presented with a phone preloaded with a reward card for the business. Using a laptop to simulate a POS a HTTP POST request will be sent to the RewardBeamer which will allow the test user to tap the phone on the NFC tag to collect rewards.

SCENARIO A

In this scenario the business is a small coffee shop. They have decided to integrate RewardWallet into their business and have decided to go with the cash back distribution model of 5%. They chose this to promote that buying a more expensive latte should yield more points.

Distribution Model: Cash Back - X% of each transaction appears as points to the customer.

Given the 5% cash back model, if the HTTP POST request indicated the transaction amount was $3.63, the test user should expect to see an additional 18 points added to the digital card.

Table 1: Cash Back model example

Distribution Model	Transaction Amount	Reward Points
5% Cash Back	$3.63	18

SCENARIO B

In this scenario the business is a small pizzeria. They have decided to integrate RewardWallet into their business and have decided to go with the purchase-based distribution model of 1 point. They chose this as they wanted a simplified system where after 5 purchases customers could enjoy any pizza for free.

Distribution Model: Purchase Based - X points will be given to the customer for each transaction.

Given the 1-point purchase based model, if the HTTP POST request indicated the transaction amount was $14.23, the test user should expect to see an additional 1 point added to the digital card. It will then be shown that a transaction amount of $18.45 will also yield only 1 reward point.

Table 2: 1-Point / Purchase model example

Distribution Model	Transaction Amount	Reward Points
1 Point / Purchase	$14.23	1
1 Point / Purchase	$18.45	1

Scenario C

In this scenario the business is a large grocery store. They have decided to integrate RewardWallet into their business and have decided to go with the quota based distribution model of 25 points for every $250. They chose this as they wanted to digitize their “$25 gift card for every $250 spent” flyer ad.

Distribution Model: Quota Based - X points will be given to the customer if they purchase $Y or more.

Given the 25-point quota based model, if the HTTP POST request indicated the transaction amount was $204.93, the test user should expect to see no additional points added to the digital card. It will then be shown that a transaction amount of $268.45 will yield 25 reward points and $512.76 will yield 50 reward points.

Table 3: Quota Based model example

Distribution Model	Transaction Amount	Reward Points
25 Points / $250 Purchase	$204.93	0
25 Points / $250 Purchase	$268.45	25
25 Points / $250 Purchase	$512.76	50

APPENDIX II – ENGINEERING STANDARDS

The Embedded System will comply with the IEEE Std 1003.13 standards for portable Realtime and Embedded Applications [10].

The Embedded System and mobile device will comply with the IEEE 1625 standards for Batteries in Mobile Computing Devices [11].

The standards ensure safety of the system devices under environments.

The Cloud Server will comply with the IEEE Std 2410 standards for Biometric Open Protocol [12].

The standards ensure security regarding cloud communication with devices.

The Cloud Server, Embedded System and mobile application will comply with the IEEE 802.15.5 standards for Telecommunications and information exchange between systems [13].

The standards ensure security of component communications using wireless networks.

RewardWallet