5/10/2017

Hoarder

Technical Report

STUDENT NAME: NIALL CURRAN

STUDENT NUMBER: X13440572

SUPERVISOR: CRISTINA HAVA MUNTEAN

PROGRAMME TITLE: BSC. IN COMPUTING

Declaration Cover Sheet for Project Submission

SECTION 1 - Student to complete

Name:
Student ID:
Supervisor:

SECTION 2 - Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: _____ Date: _____

CONTENTS

EXECUTIVE SUMMARY

This report outlines my idea to design and develop an everyday shopping mobile application, which utilises a phone camera to scan and recognise barcodes to be used in place of the standard checkout process of a grocery store. At the moment, there is a lack of commercially acceptable shopping apps for in-person shopping, so I have decided to look into different solutions and develop an easily implemented application for both the store and the users alike. Given that the application only requires a phone's camera and access to the specific shops database for item data, it is very easy to integrate into day-to-day life. In this report, I will explain how the application is intended to be used as well as what happens in the background from when the user scans an item to when they checkout and leave the store.

The inspiration for the application comes from the SuperValu/SuperQuinn device called "SuperScan" which allows a customer to register and pick up a scanning device on entry to the store and use it to scan each of their items to assist with the checkout process. This was a very innovative way of shopping, however, with the development of mobile phone technology it has become outdated, given that we now have a far more powerful machine in all of our pockets, which with the help of the Hoarder application, will allow us to complete the entire checkout process without human interaction. The SuperScan device for example, cannot accommodate payment/checkout and requires staff of the shop having to read the data from the device before manually processing the payment from the customer.

The project consists of a mobile application, a server side application and a NoSQL database. The mobile application allows the user to create an account to use the service, scan each item, pay for their purchases, review any receipts they have from previous purchases and hold the user's loyalty credit for use in future transactions. The server side application will act as a bridge between the mobile application and the database. You could refer to this application as an API as it carries out all of the main functionality for the mobile application. The android application will send HTTP requests to the server side application and the server side will need to deal with each request accordingly. These requests may include receiving item data when an item is scanned, retrieving user data, adding user data and retrieving the user's receipts and order details. The database for the Hoarder application is a NoSQL Mongo database which consists of 4 collections. The user collection, the receipt collection, the item collection and the store collection. This database makes contact with the Node.js server to be able to complete the base functions of the application.

INTRODUCTION

Hoarder is a barcode scanning shopping application with the primary goal of enhancing the shopping experience for the customer and allowing for quick and easy processing of transactions within the stores supported by the application. The app utilises a phone camera to scan the items and allows for secure payment using the MasterCard simplify API when completing the purchase.

The intended customer base for this application is vast and includes every possible type of shopper. For example, a customer getting their weekly groceries would find this system very beneficial. It would cut the time spent queuing at the checkout waiting for a member of staff to become available, as well as the time spent scanning each item and processing the payment. It would also be incredibly helpful for a person who wishes to purchase a small number of items quickly while for example, waiting on a bus to arrive, as their transaction would be fast yet secure and ensure that they don't risk missing their bus while waiting to pay.

BACKGROUND

Back in the late 90's a device was introduced into all SuperQuinn stores called 'SuperScan' which aimed "to take the queuing out of shopping". Shoppers would pick up a SuperScan scanner device from a shelf on entry to the store. They would scan each item they intended to purchase before placing it into their cart. When they had completed their shopping, they would go to the SuperScan checkout to pay for their purchases. As all of the item data was gathered by the device while the customer was shopping, the checkout process was much quicker than the conventional method of scanning each item at the checkout. This worked out very well for the store itself as they required less staff to work on the tills and instead could focus on the overall running of the store.

This solution worked well for a time and is still in use, even after the takeover of SuperQuinn by SuperValu, who have integrated the device into their existing stores. However, it has become rather outdated given that the device most people would have every day in their pocket would be far more powerful than the simple scanning device currently in use. The main disadvantage of the SuperScan device is that it cannot process a payment, so although the shop requires less staff at the checkout, they are still required at all times. This is where Hoarder could be very useful.

Hoarder is not confined to one specific store, it also allows for registration, scanning, payment and receipt generation. This makes the application far more advanced than the simple SuperScan device. Of course, the store would need to integrate some security for shop lifting and theft as the Hoarder application is not designed to combat the pre-existing shoplifting threat. However, it is very easy to check if a user has paid, as a receipt is generated and can be compared to the items in the customers' bags. This is similar to what was used by SuperQuinn with the SuperScan device for managing theft/shoplifting in their stores.

While online shopping has become quite popular, many people still prefer to visit the store rather than having their items delivered, as they like the idea of browsing and may see other items they would not have noticed when shopping online. This is particularly true when applied to grocery shopping. With this in mind, I researched to see if there were any other shopping apps already in place. Apart from NFC shopping which is used in Japan (it is also in an experimental phase outside of Japan), and the recently announced Amazon shopping solution "Amazon Go" which requires a lot of modification to the store itself, there aren't any commercially used in-person shopping solutions on the market. I find this quite unusual. The Amazon solution looks to be very innovative, but it has some drawbacks which include the store having to implement RFID tags into every item as well as specific entrances/exits which interact with the user's phone when they enter and exit the shop. The Hoarder application does not require any such technology as it is based on technology that is already in the store. It also has the capability to add NFC integration alongside the existing Barcode scanning as a future development.

AIMS

The aim of this project is to develop a barcode scanning shopping application in android studio, with a Node.js and MongoDB backend, which will allow the user to scan each item they wish to purchase before placing it in their physical shopping cart. It will also allow them to checkout and pay for their goods using the simplify payment API developed by MasterCard. This API grabs the credit/debit card information, encrypts the data in transit and generates a token for that particular payment which can only be used once. The application will have to send HTTP POST, GET and PUT requests to the Node.js server to be able to insert and read data to/from the Mongo database. The requests will include grabbing user and item data, creating and updating user accounts and sending payments to the API.

The application will rely on the store committing to using the application so their database can be accessed to retrieve all of the relevant data. However, with no cost to the shops other than the initial set up costs, and the need to hire less staff to operate the checkouts, there would be no reason why they should not consider implementing the system. The idea was originally conceived as a way of replacing and improving upon the existing SuperScan devices in SuperValu stores nationwide as that system is now outdated. The various stores would need to implement their own security precautions to prevent shoplifting, for example, checking one in ten customers' receipts when exiting the shop as was the case with the SuperScan system. The

principle of the application is to improve the shopping experience for the customers leaving the stores to implement security precautions as they see fit.

TECHNOLOGIES

In developing this application, I have utilised many different technologies and languages which I have outlined below. When completing this project, I tried to use the technology that would be best to achieve the aims of the project. I specifically used technologies that were interesting to me but I also tried to use new technologies where possible, in order to broaden my knowledge of these new technologies.

DEVELOPMENT

Android Studio:

The Hoarder Client side application was developed using Java in the Android Studio IDE. This is where I designed the look and developed the functionality for the mobile application.

Node.js:

The server side application for Hoarder was developed using Node.js which was used to make contact with the database and the client side application to allow for the core functionality of the overall application.

MongoDB:

For the database, I used the NoSQL database called MongoDB which, unlike SQL databases, uses collections instead of the conventional database tables and stores the data in JSON objects.

Simplify API:

For the payment processing, I used the simplify payment API developed by MasterCard which is integrated into the server side application. When the card details are sent to the server, the server will be able to encrypt the data and generate a token for that specific transaction before sending it to the stores simplify dashboard for payment processing.

TESTING

Mockito:

Mockito is a common mocking framework that allows me to mock classes, or in my case activities. This was used in my android unit tests to send mock data to different functions in the application.

Python:

Python was used to send requests to the server repeatedly to test performance under load.

DEPLOYMENT

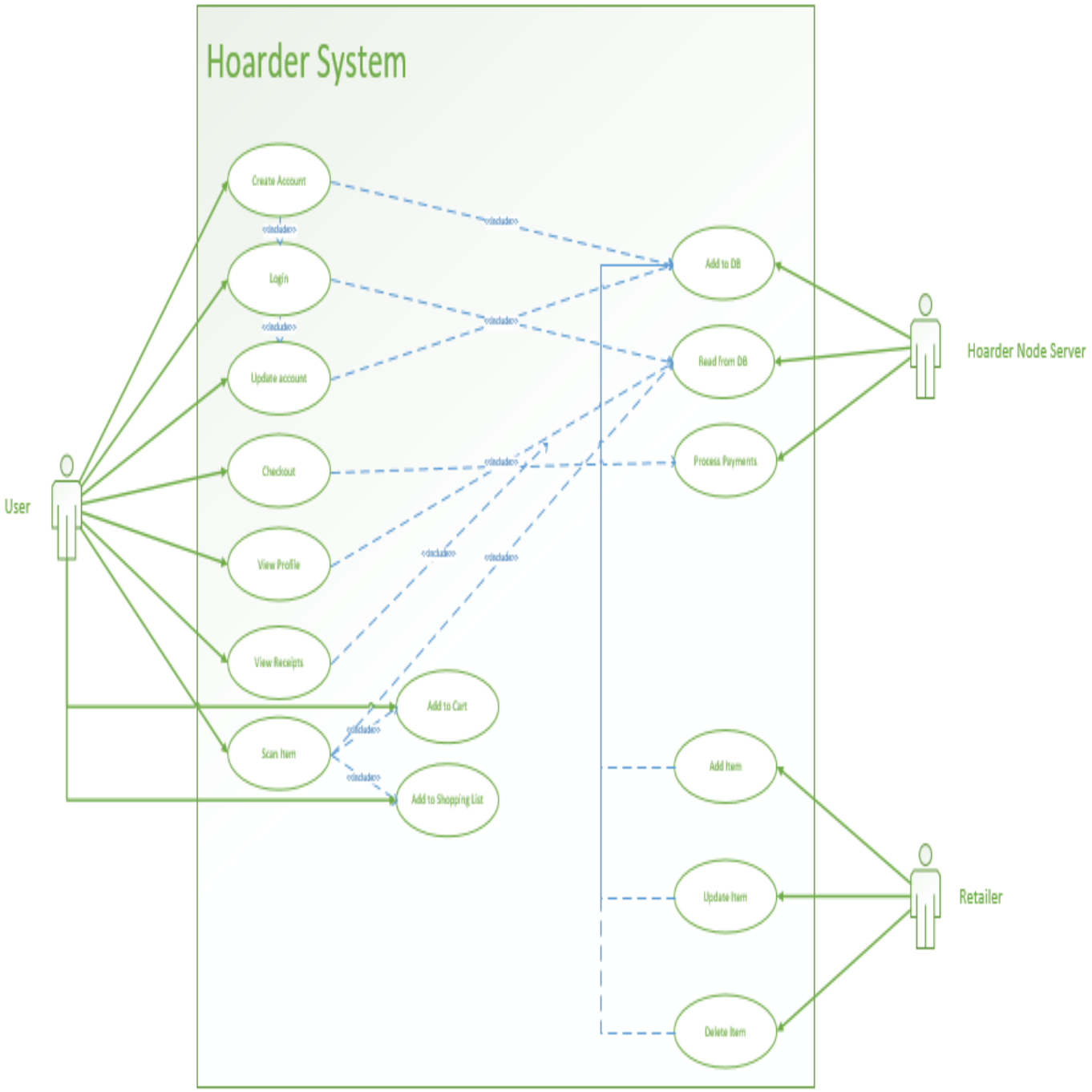
Heroku:

Heroku is a cloud based hosting platform which can easily deploy code from a git branch. I decided to use this to host my Node.js API as it works quite well with Node.js and is quite reliable.

REQUIREMENTS

FUNCTIONAL REQUIREMENTS

USE CASE DIAGRAM:



REQUIREMENT 1 <CREATE ACCOUNT>:

DESCRIPTION & PRIORITY

The account creation function will allow the user to create a Hoarder account which will then give them access to the rest of the application. The scanning, payment and overall server side functionality cannot be accessed until the user has both created an account and logged in.

The user will be prompted to enter their credentials (Email address, password etc.) and then click submit, which will send a HTTP POST request to the server to add an account to the database. If the email address does not match any currently in the database, a response will be sent back to the application on whether the request was successful or not. If the user creation was successful, the user will be able to login and access all other functions of the application.

Priority: Medium

USE CASE

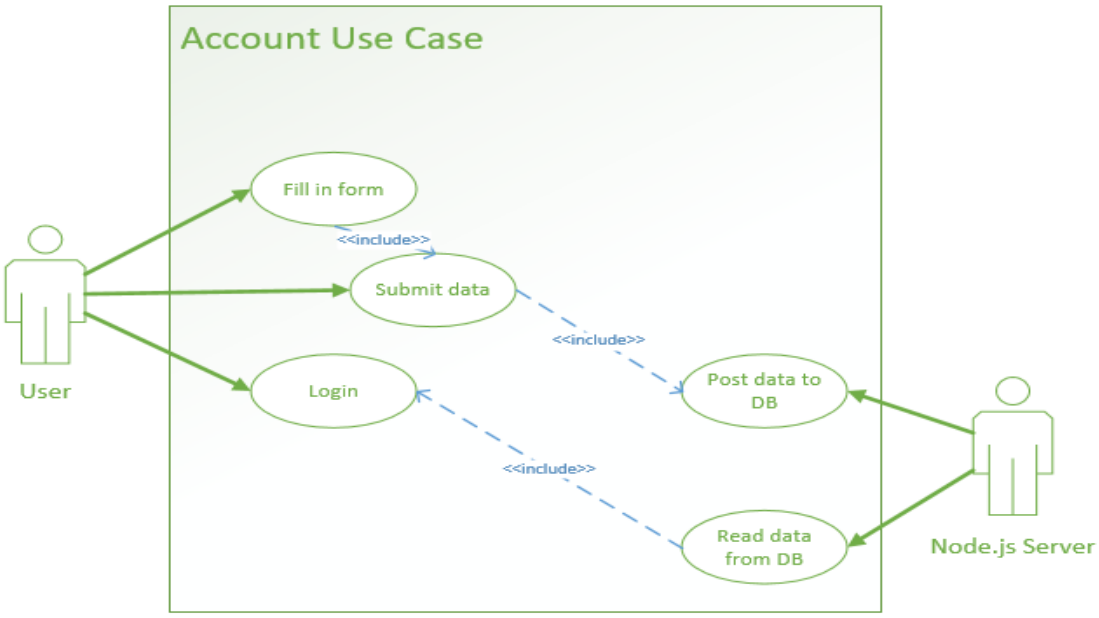
Scope:

The scope of this use case is to take the users credentials and create a user account in the database based on their input

Description:

This use case describes the user creation process by which the user will be asked to input a list of credentials and once validation of the information is complete this information will be sent on to the node server for processing and adding of data to the database.

Use Case Diagram:



REQUIREMENT 2 <CHECKOUT >:

DESCRIPTION & PRIORITY

The checkout function is a very important function within the Hoarder system. This functionality will allow the user to input their payment credentials and make the payment fast and securely. The server will receive a HTTP POST request from the application side which will be used to make contact with the MasterCard Simplify API. The API will receive the payment details and total cost of the order, encrypt the data in transit and generate a token based on that specific payment, which can only be used once.

If the request is successful in reaching the payment API, a response will be sent back to the application on whether the payment was successful or not. If the payment creation was successful, a receipt will be generated from the cart data and added to the users account.

After the payment and checkout process is complete, the cart is cleared and the user will be redirected back to the main/scan screen and the receipt will be available when the user views their profile.

Priority: Medium

USE CASE

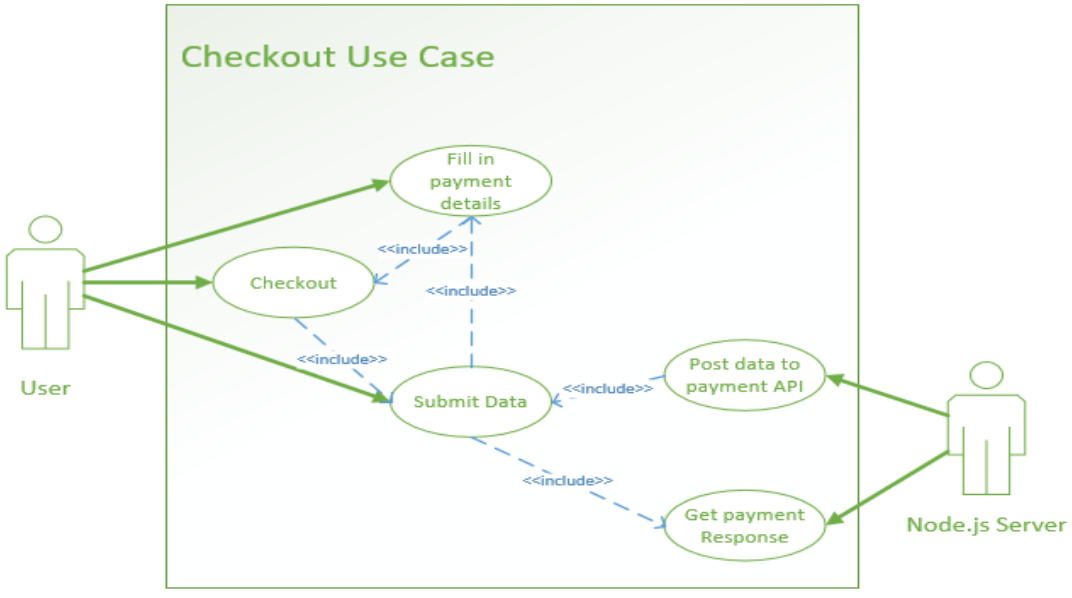
Scope:

The scope of this use case is to take both the total cost of the order and the payment details of the user and send the data to the server which will make contact with the simplify payment API and store the payment data in the payment dashboard for simplify.

Description:

This use case describes the checkout process by which the user will be asked to input their payment details including credit card number, CVC, expiry date and full name of card holder. Once validation of the information is complete on the application side, the order and payment information will be sent on to the node server which will make contact with the payment API.

Use Case Diagram:



REQUIREMENT 3 <PROFILE >:

DESCRIPTION & PRIORITY

The profile function allows the user to freely browse all of their previous orders through use of the receipts page, Update user details and information and view the loyalty credit that the user may have.

Although I would have described this function as having a low priority I believe the receipt page is of utmost importance as it will be the only way of the store monitoring proof of purchase.

The users profile page will make an initial GET HTTP request to the server to grab all of the necessary details and populate the page based on this information.

If the user wishes to update any information they can do so by clicking an edit button which will prompt them to input the information to be updated into a text field which will send a POST HTTP request to the server, which will then update that particular users account information.

Priority: High

USE CASE

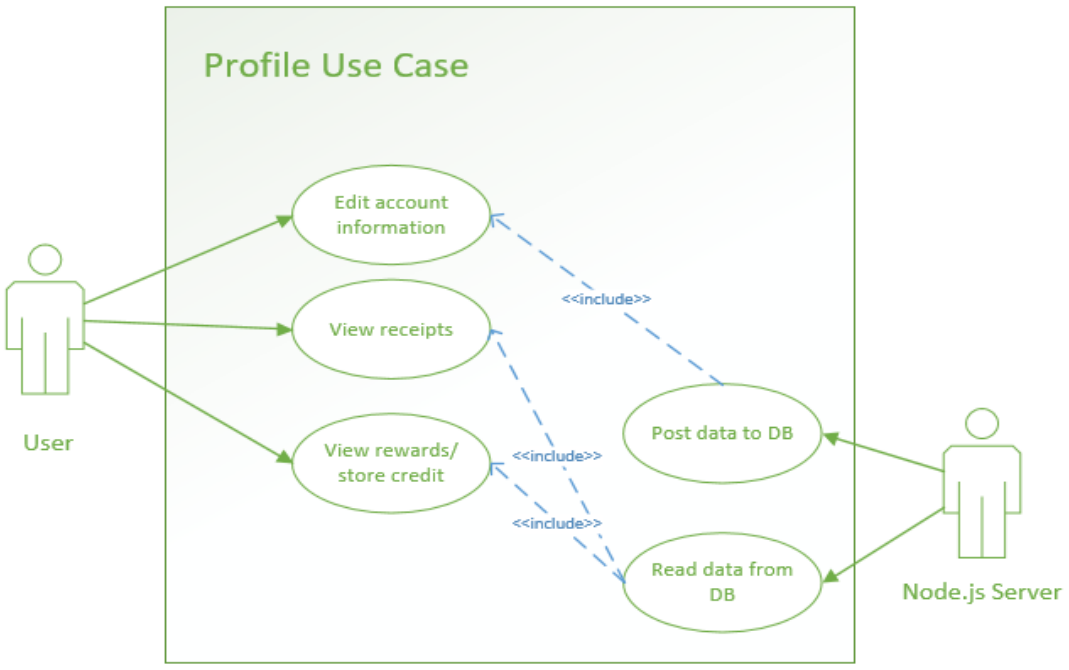
Scope:

The scope of this use case is to receive the users email address and make a GET request to the server to receive the users receipt information, reward information and allow the user to be able to update specific user information.

Description:

This use case describes the profile functions and the corresponding pages that apply to it. The user will be able to view all details that apply to their account including their overall profile information, their reward points/store credit and their receipts and order data. The user will also be able to update account information as they wish through use of their account page.

Use Case Diagram:



REQUIREMENT 4 <SCANNING >:

DESCRIPTION & PRIORITY

The scanning function is the biggest part of the application as it the singular thing that describes the application as a whole. Without the scanning functionality, the application is not meeting the minimum requirements set out initially so it is of the highest priority to get this system to work flawlessly in order to have the app commercially acceptable. The scanning functionality will be able to utilise the camera of a phone to recognise barcodes and read the data in a matter of milliseconds. Once the barcode content is received by the application, A HTTP GET request will be sent to the server with the barcode content value and gather the item name, item cost and item category and send it back to the application in JSON format.

Once the item data is received, it is displayed within a box in the centre of the page where the user can decide to add to their shopping list or their shopping cart.

Priority: Very High

USE CASE

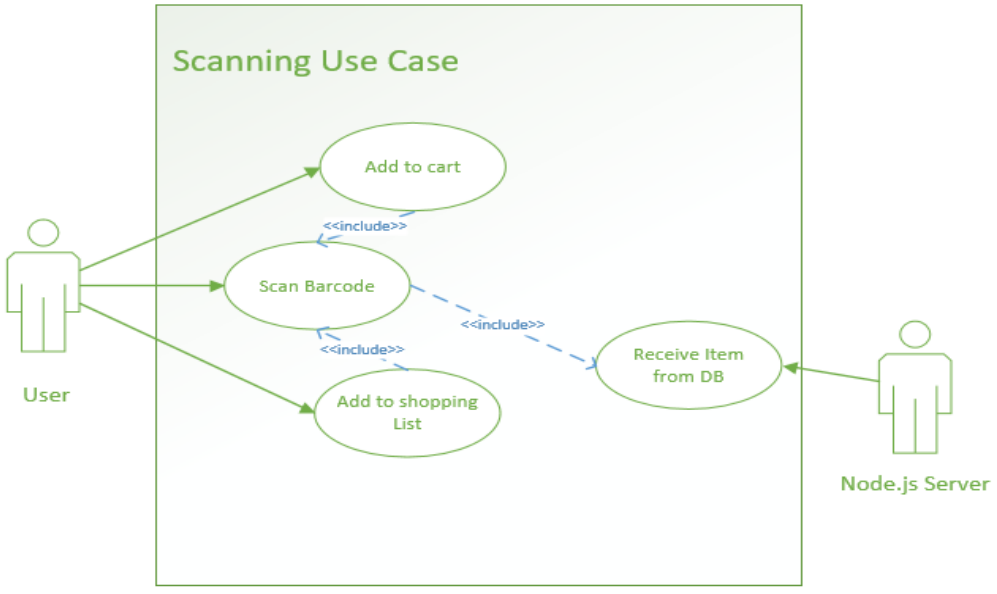
Scope:

The scope of this use case is to receive the barcode content after scanning a particular item and make a GET request to the server to receive the item data which will then be used to populate a box in the centre of the page allowing the item to be either added to the users shopping cart or their shopping list.

Description:

This use case describes the scanning function where the user will be able to scan a barcode, receive a barcode number, send the data to the server and receive the corresponding item data for that barcode. They will be able to add to their cart from this page and can also decide to add to a shopping list that is linked to their specific account.

Use Case Diagram:



REQUIREMENT 5 <SHOPPING LIST >:

DESCRIPTION & PRIORITY

The Shopping list page will simply store items that the user may frequently get or wish to get on their next visit. This is a low priority piece of functionality as it will not affect the overall application if it is left out however it is a desirable feature which I would like to implement and would not be overly time consuming as it would have very similar functionality to the cart function.

Priority: Low

USE CASE

Scope:

The scope of this use case is to store item data in a shopping list that can be used to assist a Hoarder user for their next visit to a specific store.

Description:

This use case describes the shopping list function which will simply display items the user may have stored for their next shop or frequently bought items that the user wishes to keep consistent.

REQUIREMENT 6 <SHOPPING CART>:

DESCRIPTION & PRIORITY

The Shopping cart page will simply store items that the user has scanned during their shop and items they wish to purchase on checkout. This is a high priority piece of functionality as it will affect the overall application if it is left out. This is part of the main screen of the application and allows the user to have access to the payment/checkout functionality.

Priority: High

USE CASE

Scope:

The scope of this use case is to store item data in a shopping cart page that the user can access at any time to monitor items they are collecting during their shop.

Description:

This use case describes the shopping cart function which will simply display items the user may have stored during their shop.

NON-FUNCTIONAL REQUIREMENTS

REQUIREMENT 1 <SECURITY >:

DESCRIPTION & PRIORITY

This requirement defines the security requirements. User information will need to be encrypted in the mongo database in order for customers and retailers to trust the system. Along with this the payment API will encrypt the payment information in transit and generates a token based on that transaction.

Priority: High

USE CASE

Scope:

The scope of this use case is to securely encrypt and decrypt user and payment data in order to make a commercially accessible system.

Description:

This use case describes the security precautions put in place by the system which include the encryption and decryption of user data and the encryption of payment information in transit by the simplify payment API.

REQUIREMENT 2 <RELIABILITY >:

DESCRIPTION & PRIORITY

This requirement defines the reliability of the system to ensure that the user is happy with the system. This is of utmost importance as I feel that without consistent reliability, a system will fail and no user will want to use it for that reason. I feel that in order to complete my aim of making a commercially acceptable application, this will need to be adhered to.

Priority: High

USE CASE

Scope:

The scope of this use case is to create an application that is reliable for the user so as to ensure their constant support of the system.

Description:

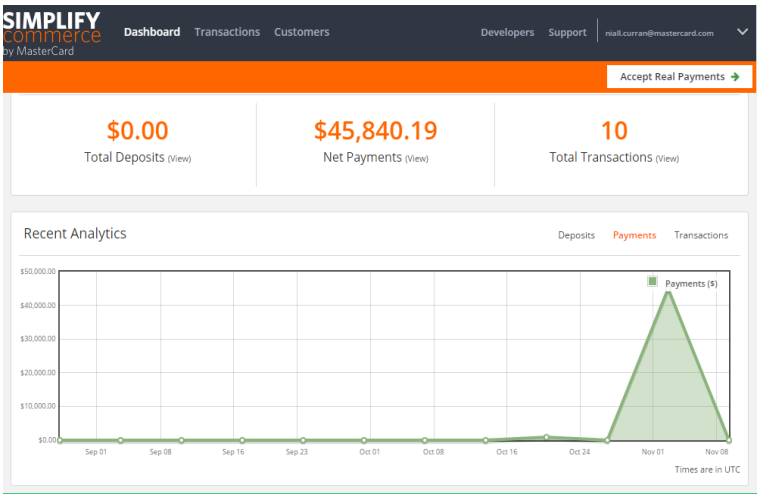
This use case describes the reliability of the system I am creating. This will include response times, secure connection to the Node.js server and overall reliability in terms of payment and user creation.

APPLICATION PROGRAMMING INTERFACES (API)

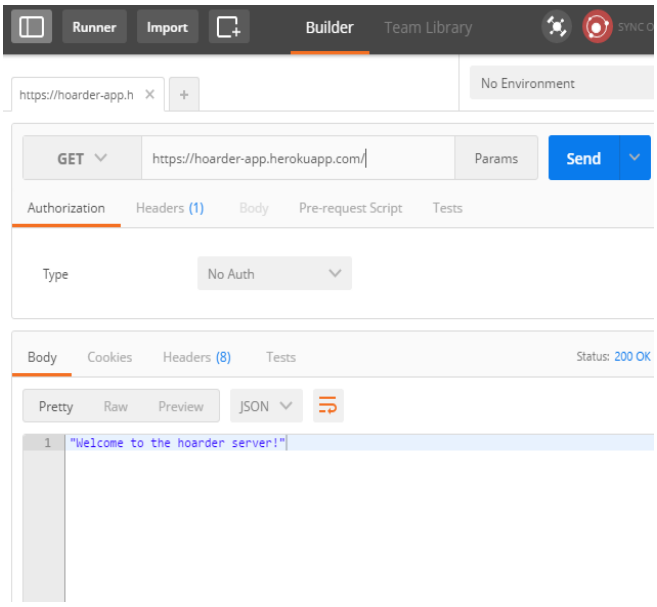
This section will define the external components of the Application including the Hoarder API which carries out all of the primary functionality and the Simplify API by MasterCard which handles the payment for the Hoarder application.

SIMPLIFY DASHBOARD

The not so common Simplify platform will be used by the Hoarder application to send payments and monitor income for the store. The image shown here is what the dashboard looks like for the developer sandbox which will allow me to be able to receive mock payment data and process the payment without having any real money moving between accounts. To use this API, I need to create an account with simplify (In this case I am using the account I used during my internship with MasterCard) and generate an API key which will be used alongside the simplify Node package to connect to the dashboard.



Once connected to the dashboard you can freely receive test payments in a realistic environment. You can view all of your payments to a specific date by clicking the net payments tab and monitor your income with the line graph supplied at the bottom of the page.



NODE.JS SERVER

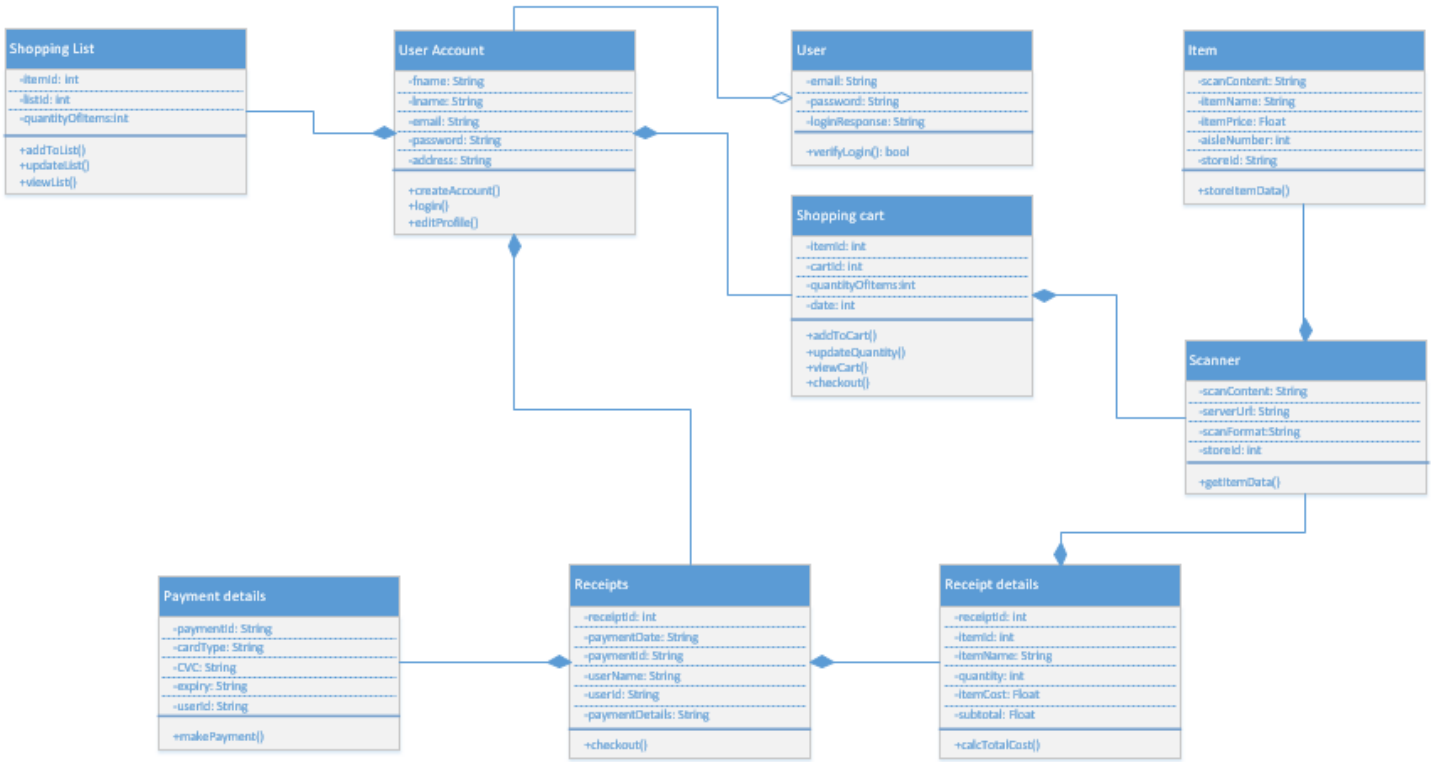
The Node.js server for the Hoarder application will do all of the work for the client side android application and carry out encryption of data, database communication and external API communication (Such as the simplify API). The image shown on the left shows a simple GET HTTP request to this server through Postman and the response the application will receive in JSON format.

The client side application will need to be able to send similar requests and be able to parse this JSON information in order to receive the required data from the database, so it is of utmost importance that I get this system working to a high standard to avoid long delays in receiving data from the server side.

APPLICATION AND SERVER DESIGN

CLASS DIAGRAM

The class diagram below outlines the initial architecture for the application. From this, you can see how the item data is gathered, user account is created and how the shopping cart gathers the information for each item contained within it. The scanner class gathers data from the barcode itself and makes a call to the item class to gather the required name and price of the item relating to that barcode content (Scan content). Once the item data has been found this is then added to the cart or used to generate the receipts. On checkout, the payment data will need to be gathered from the Payment details class before it can be sent to make contact with the payment API.



The reason I chose this architecture is because it seems to be the common architecture approach for e-commerce websites and applications. This architecture makes a lot of sense for this application and I felt, although small changes may have had to be made throughout the duration of the development process, the overall architecture would mostly remain the same.

SERVER ENDPOINTS

The following endpoints mimic that of the Node.js server which is currently deployed on Heroku.

Endpoint	**/login**
Description	When user data is posted, the server will compare the user information to registered user information and return a response based on if the data matches data in the database.
URL:	**https://hoarder-app.herokuapp.com/login**
HTTP verb	POST
Parameters	None
JSON Body	Email and password
Response	Static response in JSON format based on if the user is logged in. Status code of 200 OK or 404 Not Found used to determine if the user is logged in or not

Endpoint	**/user**
Description	When user data is posted, the server will compare the user information to registered user information and return an error if the user data already exists in the database. If not, the account will be added to the database.
URL:	**https://hoarder-app.herokuapp.com/user**
HTTP verb	POST
Parameters	None
JSON Body	Name, Email, Phone number and Password for the user
Response	Static response in JSON format based on if the account has been created. Status code of 200 OK or 400 Bad Request used to determine if the account creation was successful

Endpoint	**/finditem/{barcodeContent}**
Description	The barcode content parameter is compared to items in the database and item data is returned if the barcode matches an item in the database. Otherwise, a
URL:	**https://hoarder-app.herokuapp.com/finditem/{barcodeContent}**
HTTP verb	GET
Parameters	Barcode content
JSON Body	None
Response	Item data in JSON format if the item exists and an error code and response if it does not.

Endpoint	**/user/{email}**
Description	When an email is passed as the parameter for this endpoint, the user data is retrieved from the database and returned in JSON format. If a user with this email does not exist in the database, an error message in JSON will be returned with a status code of 400.
URL:	**https://hoarder-app.herokuapp.com/user/{email}**
HTTP verb	GET
Parameters	Email
JSON Body	None
Response	user data in JSON format if the user exists and an error code and response if it does not.

Endpoint	**/payment**
Description	Credit card information is passed as the JSON body for this endpoint and is sent to the simplify API for payment processing. A response is then returned from the Node.js server with details relating to whether the payment was a success or not.
URL:	**https://hoarder-app.herokuapp.com/payment**
HTTP verb	POST
Parameters	None
JSON Body	Amount, expMonth, expYear, cvc, number (card number)
Response	A success or failed response depending on the response from the simplify API. The response will also contain some of the errors that were encountered by the simplify API.

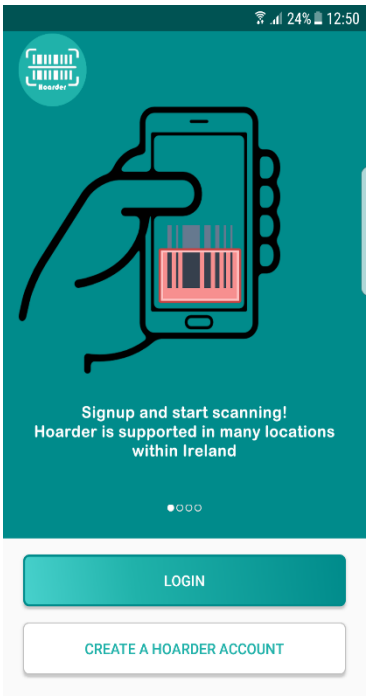
Endpoint	**/addItem**
Description	Item data is passed as the JSON body for this endpoint where the product name is compared to other items in the database and if it does not match any items in the database the item will be successfully added to the database.
URL:	**https://hoarder-app.herokuapp.com/addItem**
HTTP verb	POST
Parameters	None
JSON Body	ProductName, productPrice, productCategory, scanContent (Barcode number)
Response	A response is received from the server in JSON format telling the user whether the item has been added to the database or not

Endpoint	**/credit/{email}**
Description	The email address of a user is passed as the parameter for this endpoint as well as a credit amount as the JSON body. When this data is submitted, the credit field for the user with the supplied email address has its credit increased by the specified credit amount. If the user with that email does not exist, an error response is produced by the server.
URL:	**https://hoarder-app.herokuapp.com/credit/*{email}***
HTTP verb	POST
Parameters	Email
JSON Body	Credit
Response	A response is received from the server in JSON format telling the user whether the credit has been successfully loaded.

Endpoint	**/receipt/{email}**
Description	Receipt data is passed in JSON format along with the email parameter for the user making the payment. When submitted, the receipt data is added to the database with the users email address.
URL:	**https://hoarder-app.herokuapp.com/receipt/*{email}***
HTTP verb	POST
Parameters	Email
JSON Body	itemCount, totalCost, referenceNumber, items []
Response	A response is received from the server in JSON format telling the user whether the receipt has been generated or not

INTERFACE REQUIREMENTS

The interface of the Hoarder application is a big part of what will make the app commercially ready and because of this I will have to make an application that will look and feel like any other ecommerce application currently in use. Below I have outlined my GUI requirements as well as some information on what external libraries and API's I will be using.

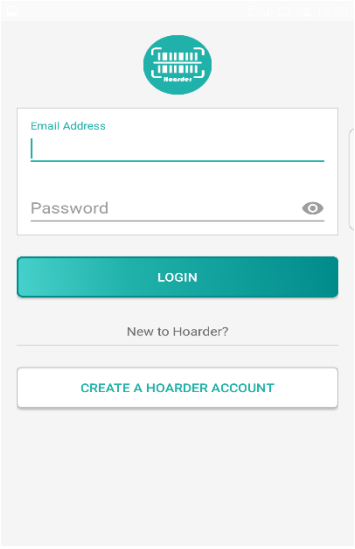


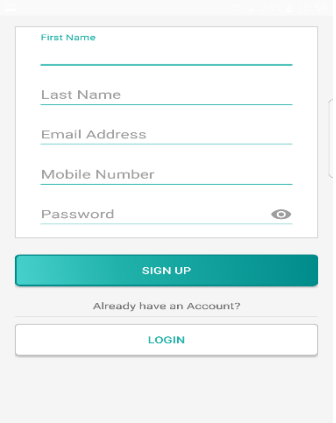
WELCOME SCREEN

The Welcome screen follows a Splash screen which displays the Hoarder logo on a white background. This is an easy to implement way of ensuring the user is not staring at a blank page while the application is loading and instead will get a small look at the applications overall colour scheme as well as the logo and title of the app. For this I, had researched the best ways to implement a system and it seemed the majority of applications used a thread sleep for this however this is not the correct way of doing this as the user will have to wait the same amount of time no matter what. Google's implementation of a splash screen however was very interesting to me as I found that you can easily implement the splash screen itself as a drawable, and incorporate a style all in xml which will enable the splash screen to load before even a single layout has initialised which makes the system very fast and responsive. When the app has loaded, the splash screen redirects to the welcome screen which holds links to the Login and Signup pages for the application as well as displaying some helpful information in the form of a set of images that the app will scroll through every few seconds.

LOGIN SCREEN

The login screen will be pretty standard however I have looked into how Facebook and other commonly used apps arrange their styles and images in a login and signup page and have designed my page around that. The login page will need to carry out one HTTP POST request which will be to login and will not require the response body as it will only need to get a 200 OK to ensure the user account exists. The login page will also need to display the two fields (Email and Password), the login button which will send the request, a link to the create account page and a link to a help page which I plan on implementing once I have the other pages completed as it will not be required for the presentation. When the user clicks the "create an account" button an intent is triggered which redirects the user to the signup page. Once the user is logged in they are redirected to the main scan page of the application.

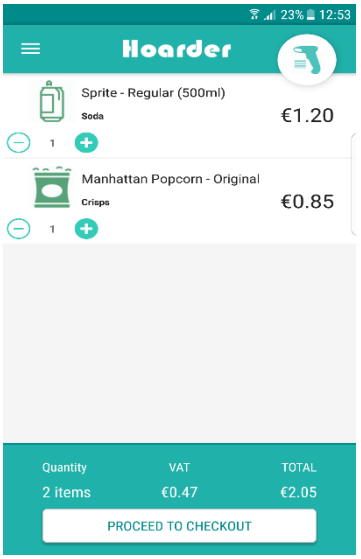




CREATE AN ACCOUNT SCREEN

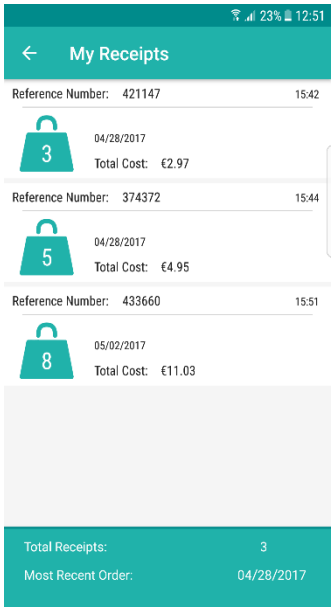
The create an account screen works similarly to the login page in that it only sends one HTTP POST request which will send the user input from the fields in JSON format to the server and add that JSON to the database. The layout is almost identical as well apart from the number of fields. Once the application receives a 200 OK and there is no duplicate account in the database the user is redirected back to the login page where they will continue further into the application.

SCAN/MAIN SCREEN



The main screen for the hoarder application consists of the virtual shopping cart itself, the side navigation bar which accesses the other screens of the application, the scan button which activates the scan screen for the application and the footer which displays price data for your cart contents. The scan screen simply consists of a camera view with a barcode reticle, utilising the ZXing barcode scanning library that will recognise a barcode and collect the content from it.

Once an item is scanned the data from that barcode is sent to the server in the form of a HTTP POST request where it will compare that barcode data to any that exist in the database and return product data only if the barcode has been registered as a product in that store. If the item exists, the user will see it in the cart list view contained in the centre of the screen which, similar to the other list views of the application, will utilise a custom row for each of the items in the list.

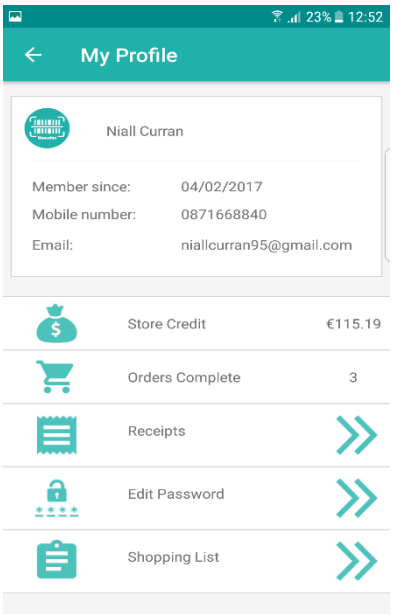


RECEIPT SCREEN

The receipt screen contains all of the order details for that user and could be used as proof of purchase and a way of the user monitoring how much money they are spending on items. The list View for this will be similar to the list View in the Main Page and the shopping list page as it will also use a separate row layout to define the look of each receipt in the list. Here, the application makes a call to the server to grab all the data from the receipt collection and populates an Array List which will be used to populate the list view.

The app bar at the top of the screen is the same as before allowing the user to go back to the main page. The rows themselves will contain the number of items, cost of order, time and date of transaction as well as a reference number for that receipt. The footer is the final element of this screen and simply contains the amount of receipts you have and the date of your most recent transaction.

PROFILE SCREEN

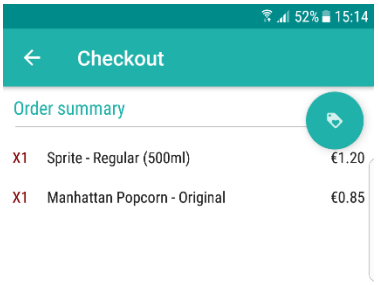


The profile screen contains all of the information that the user supplied on creating an account with Hoarder. The screen consists of a header and footer with the same top bar as before that allows the user to navigate back to the main screen. The header holds the user information such as name, date of account creation, mobile number and email address. The footer holds both store credit and order count information and links to other screens of the application namely the receipts screen, shopping list screen and the edit password dialog box.

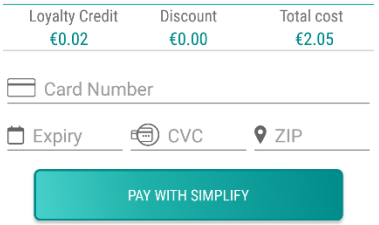
The edit password box will allow the user to input both their current password and their requested ‘new password’ and once submitted, the server will decide whether the current password matches the one in the database for that user and if so will approve of the password change.

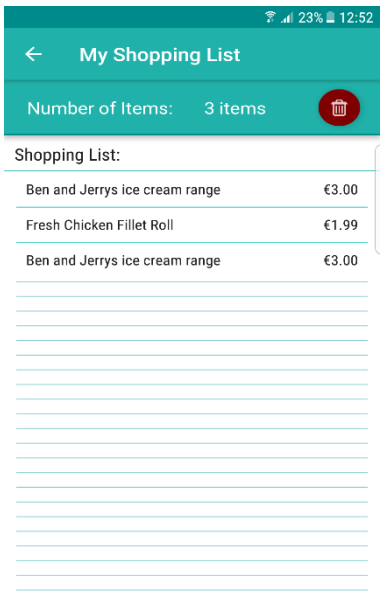
PAYMENT SCREEN

The checkout/payment screen of the application is the final step in completing the transaction. Here, the user can decide whether to apply a discount to the transaction with their loyalty credit using the loyalty credit button in the top right of the page. This screen consists of a list view containing the products the user is wishing to buy and a footer which contains the details of the order including the total cost, loyalty credit the user will gain from this transaction and any discount the user applied using their loyalty credit.



Again, the list view contains custom row layouts which in this case consists of a counter for the number of items you have of that particular name, the item itself and the cost for that item.





SHOPPING LIST SCREEN

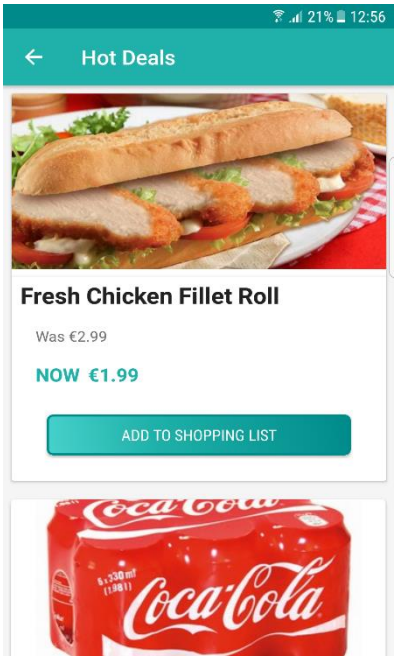
The shopping list screen consists of a list view containing items that the user had decided to save for a later time or items that they frequently purchase. The user can interact with their shopping list by clicking items that have been scanned and clicking the ‘Add to shopping list’ button contained in the dialog box or by visiting the ‘Hot Deals’ page and clicking the ‘Add to shopping list’ button under each deal item.

The header of this screen will show the user the number of items in the list as well as giving the user the option to clear the list. By clicking each item in the list, similar to the main screen, you will have the option to remove that specific item from the list.

HOT DEALS SCREEN

This screen will act as the advertising for the shop where users can see specific price reductions or deals that are currently happening in the store. The screen consists of multiple card views containing each deal and a button for each which will add the deal item to the users shopping cart.

The top bar for this page is the same as most of the pages where it consists of a link to the main page of the application.

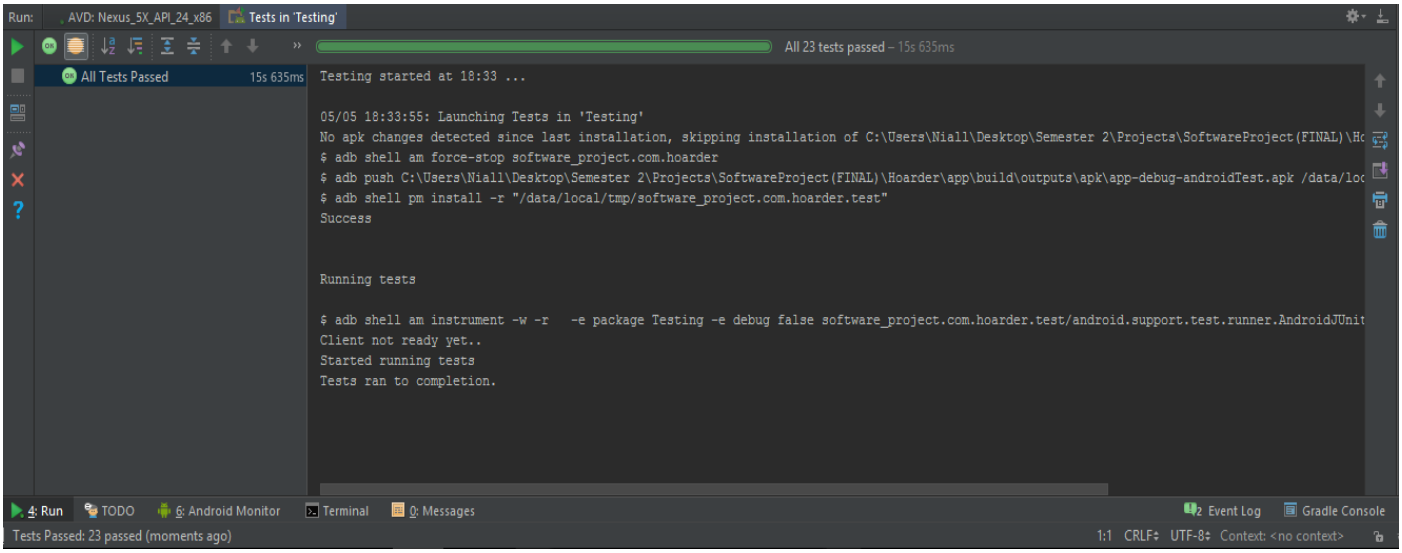


TESTING

Testing for the Hoarder application has been carried out in 3 phases starting with the standard unit testing which took place during the development process and ending with usability testing where I will be looking to gather people that may wish to use the application, other people in the software industry and retail workers for stores that Hoarder would be targeting. Below, I have outlined these phases and explained each of them in detail.

UNIT TESTING

When developing the Android application, I wrote unit tests within Android Studio to test activities and views that I was developing at the time. Android studio makes it easy to test all of the features of your application and there are many frameworks that can be easily implemented into the unit tests that will help with the mocking of information such as Mockito.



My unit testing was performed entirely in Android Studio and was quite successful. The setup and destroy functions were created to launch each activity depending on the test class file and once launched, I was able to test to ensure whether the correct xml elements have loaded in the UI and whether certain functions worked to the standard expected or had the correct error responses. The initial testing was completed early on in the development process where I had tested major functionality for the application and as development went on, these tests needed to be changed to ensure new UI elements were being checked for on launch.

The above image is from recent testing where I tested some of the main activities for my application and as you can see, all tests passed in good time considering this had to launch multiple activities and check for xml layout content on launch. This testing helped along the way as it was used to determine the stability of the application as I was in the development phase. However, I quickly changed my testing method towards the end of the development process and decided to pursue a more user facing approach to testing where I distributed APK files for testing purposes to members of the public in various fields. I will explain this in greater detail in the following section ‘Usability Testing’

PERFORMANCE TESTING

Performance testing was used to determine the backend quality of the application such as the responsiveness and the stability under various amounts of workload which could be caused by multiple users contacting the server/database at one time. This was used when testing the quality of the system in areas like scalability, reliability and data/resource usage. To complete this form of testing I had written a python script that sends requests to the server URL repeatedly every few seconds and would stop once the URL could not be reached anymore. I increased the interval in seconds between each request as the testing continued. When I didn't use this script, I sent requests through Postman where I could visualise each response clearly and still send requests very easily. The Postman testing benefited me the most as I used it for most of the endpoints whereas the python script only accessed the main URL of the server (Test URL). The test URL I used was https://hoarder-app.herokuapp.com/ where the response simply greeted the user.

USABILITY TESTING

The purpose of Usability Testing is to determine whether the application meets the overall project requirements set out at the beginning of the development process. When development process had ended and I felt the system was stable and commercially ready, I generated APK files and distributed them within a group of technical people (Software & Hardware) and regular every-day users. Each user received an APK file for the application itself, a list of barcodes that can be used for testing and a survey asking important questions relating to the speed, responsiveness, look and overall functionality of the application as well as the backend Node.js server. Out of all the people I tested the application with, all of them said that they would be inclined to use the app if it were to be available for their device and over 80 percent of them mentioned that they would be very likely to recommend the application to others.

In terms of speed and responsiveness, both received an average rating of very fast and very responsive which I was very pleased with given that they were the highest rating possible and I had set out in the beginning to primarily develop an application that was to the standard to be expected from a commercial application which includes good responsiveness, user friendly UI and speed of functionality. The comments that were supplied helped me to determine what further modifications to the application I could possibly make in the future including small changes like saving the users card information for future transactions or large changes like adding entirely new functionality. In the appendix, I have supplied the surveys I received from these testers.

RESULTS FROM TESTING

The initial unit testing, as discussed above, was successful during the development process and helped me to ensure any additional functionality I was adding to the pre-existing activities would not cause the application to fail. These tests would launch each activity individually and test before I generate any APK files to be added to my android device. Performance testing was then carried out on the server to determine the state of the server under stress and/or load. Although these tests were beneficial, I felt that the form of testing that gave me the most interesting results was the usability testing that I carried out with manual testers. The feedback I gathered from this form of testing was all positive in the sections for speed, navigation, responsiveness and overall application quality. Over 90% of the people that I had surveyed had also selected either likely or very likely to the "How likely are you to recommend this application to others?" and the "How likely would you be to use this application if it was available on your device?" questions. Overall, the results gathered from the testing of my application meant that I had succeeded in developing a commercially ready application that focused on speed, reliability and user experience.

CONCLUSION

The design and development of the Hoarder barcode shopping application was quite a challenging project but was ultimately a very rewarding experience for me. I decided to use languages that I did not have experience using and some technologies that were new to the industry in general. I feel if I had used different technologies that were more well-known or that I had more experience with, I may have had an easier time developing the application overall. However, at the same time it was a much more beneficial and interesting endeavour for me to work with the technologies I have chosen such as the relatively new Node.js server side language or the MongoDB NoSQL database.

My goal at the beginning of the development process was to develop an app that closely resembled the apps that are currently being used by people in their day to day lives. I have used apps like Amazon, Just Eat and Deliveroo to determine what is to be expected from the UI of a commercial application as well as determining any extra functionalities that my application should have, to put it in the same class as these applications such as customer support and a loyalty system. Additionally, I also set out to have the speed, reliability and responsiveness mimic one of these apps and, given the results of my surveys and my own personal testing, I can gladly say that I have achieved what I set out to do at the beginning of the development process in all of these areas.

A development from working with these technologies during this project was my new-found interest in Android development and mobile development in general. Ever since developing the initial prototype for the Mid-Point presentation, my interest in this area has grown to a point where I have researched IOS development also and am actively pursuing careers in the mobile development area. Given this new-found interest in mobile development, I feel that the development of this application as my software project for this course has equipped me with some of the skills I may require to be successful in the role of a mobile app developer in the future.

BIBLIOGRAPHY

OFFICIAL ZXING ("ZEBRA CROSSING") PROJECT HOME

Barcode scanning library | *GitHub*

[online] Available at: https://github.com/zxing/zxing [Accessed 2016].

Node.js documentation

Nodejs | documentation

[online] Available at: https://nodejs.org/en/docs/ [Accessed 2016].

ROBOMONGO — NATIVE MONGODB MANAGEMENT TOOL (ADMIN UI)

Robomongo.org | *Robomongo — native MongoDB management tool (Admin UI)*.

[online] Available at: https://robomongo.org/ [Accessed 2016].

TESTS

Android tests | *Building Local Unit Tests* | *Android Developers*

[online] Available at: https://developer.android.com/training/testing/unit-testing/local-unit-tests [Accessed 2017].

SPLASH SCREENS THE RIGHT WAY

Big Nerd Ranch | *Splash Screens the Right Way*

[online] Available at: https://www.bignerdranch.com/blog/splash-screens-the-right-way/ [Accessed 2016].

GOOGLE/VOLLEY

GitHub | google/volley

[online] Available at: https://github.com/google/volley [Accessed 2016].

HEROKU DEV CENTER

Devcenter.heroku.com | *Heroku Dev Center*

[online] Available at: https://devcenter.heroku.com/ [Accessed 2016].

Simplify Documentation

Simplify Commerce | *Docs*

[online] Available at: https://www.simplify.com/commerce/docs [Accessed 2016].

APPENDIX

APPENDIX A: PROJECT PROPOSAL

Hoarder

Android shopping app using barcode scanner

Student Name:	Niall Curran
Student Number:	x13440572
Student Email:	niall.curran@student.ncirl.ie
Degree Programme:	BSc. In Computing
Specialisation:	Internet of Things(IOT)
Date of Submission:	21/10/2016

_____

Signature of student and date

Objectives

My primary objective for this project is to develop a functional shopping app with barcode scanning to grab item data and allow the user to pay for products quick and easily without requiring going to a checkout desk. I have broken the project down into the following objectives.

Barcode scanning using phone camera

My first objective will be to experiment with libraries and methods of allowing the users phones camera to read a barcode and throw back the content and format of the barcode. The prototype will need to grab the data and add it to an array List for the cart display. Once I have created a prototype I can begin working on grabbing the item data from a database and displaying It in an android studio List View.

Node.js server for back end:

Before I can finish working on the database I need to build a server side application which will process the payments and make contact with the database for this I could have gone with PHP or Ruby but I have decided to go with Node.js as I am more familiar with it from my internship and find it enjoyable to use. This server will receive http requests from the app which will include processing a payment, creating a user and getting product data. This will be the backbone for my project.

MySQL database & Hosting:

Before I can fully test the application, I will need to have both the Node.js server and a database deployed to some form of web server. At the moment, I am weighing my options on what to use but I am going to start with a free hosting service called Heroku which will allow me to deploy the node.js application and create a PostgreSQL database alongside it to store user information and the catalogue of items. I can then edit the database and deploy through GitHub if I choose from the Heroku dashboard which will be helpful once I get it set up.

Payment through MasterCard's simplify platform/API:

The final step of my project will be the payment functionality which instead of using the conventional PayPal I will use MasterCard's simplify API which does not require the user to create an account and will simply take in their credit card information, generate a token for that transaction and once it is used to process the payment it cannot be used again allowing for a safe, stable and easy to use payment system for my application.

Background

My idea came from the decade old SuperScan that SuperQuinn had started using in their shops and then which transferred to SuperValu after their acquisition. These are handheld devices that allow the user to scan items and then pay at the end when they reach a clerk and for every 10-people exiting the shop their receipt would be reviewed and their bag is checked to insure the amount paid matches the value of the products in their bag.

This system was clunky and did not do away with having to wait in a cue to pay for your items. It received a lot of praise at first but slowly died out with the need to register for the use of the device and the need to carry the device around the shop with you. This has therefore caused a steady decline through the years of people using this device to a point where there is only a small minority of people using it at all.

This is where my idea comes into play. Why would someone need to carry around one of these devices when the majority of people have a smart phone with a camera and the same capability in their pockets at all times? My application will use a phones camera as the barcode reader and will grab data from an external database where the catalogue of items relating to each barcode lie.

Another advantageous aspect of the application is that it could be used in multiple shops so the user can finish shopping in one and move to the other while staying within the same application. All we need to do on the development side is reach out to the shop and get the database connection info that holds the product data and attach the payment system to the shops bank account. The shops will also not have to spend money on the scanning devices so the likes of SuperValu who are currently using these devices will save a lot of money by using the application

A use case for this application would be like this:

'A user wants to get his/her shopping as quickly as possible over their lunch break. The user enters the shop and immediately gets a trolley to carry his items. They have five things to purchase milk, eggs, bread, crisps and washing powder. They open their app, log in and begin scanning items and putting them in their trolley. Once they have all of their items scanned and in their trolley, they check out by inserting their card details or just checking out with card details on file and walk towards the exit where they show their receipt with verification code and that day's date to a member of staff at the door and exit the shop'

As you can see most of the thought has been put into user experience rather than security and theft prevention however these are precautions the shops can put in place but the overall app will not require the shop making any changes to the overall look of the shop only the users checkout experience.

Technical Approach

Before developing the final application, I should continue to build prototypes that demonstrate the functionality I require and when I reach a point where I am happy with the look and design of the prototype I will add it to the final application. However, I feel that until my requirements document is finished I should not continue development and should instead focus my efforts on getting the documentation of the application finished and then and only then begin working on the final applications functionality to a large extent.

In terms of documentation I will be continuously reading articles on the android developer website as well as browsing other sites and sources such as videos on Udacity and books on android from the library. This will give me a greater understanding of how to create an industry worthy android application. I will also research specific libraries in order to get the best out of my application and implement them accordingly if required.

To build the application I will be using java as my primary language while using a node.js backend and MySQL database. For the early prototypes, I will use SQLite to determine the tables I need and for testing purposes but at a later stage I will implement a MySQL hosted database with extensive security. I will host the node.js and database backend through Heroku and will have the android app send http requests to the node.js server to complete the signup/login, grabbing item data and processing the payments features of the application.

The barcode scanning will be done by use of the phones camera and the ZXING library which recognises the barcode and grabs the content and format of the barcode. The application will then reach out to the database and grab the data for the item that barcode corresponds to. The product data will then be added to an array List which will then be displayed to the user in the form of a List View in android.

Project Plan

Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
	Software Project	**159 days**	**Tue 27/09/16**	**Fri 05/05/17**		
	Scope	**6.02 days**	**Tue 27/09/16**	**Wed 05/10/16**		
✓	Determine project Idea and	1 day	Tue 27/09/16	Tue 27/09/16		
	Prepare pitch slides	1 day	Wed 28/09/16	Wed 28/09/16	2	
	Determine all possible ways of improving the idea	1 day	Thu 29/09/16	Thu 29/09/16	3	
	Pitch Idea to Board of Lecturers	10 mins	Wed 05/10/16	Wed 05/10/16	4	
	Scope complete	0 days	Wed 05/10/16	Wed 05/10/16	5	

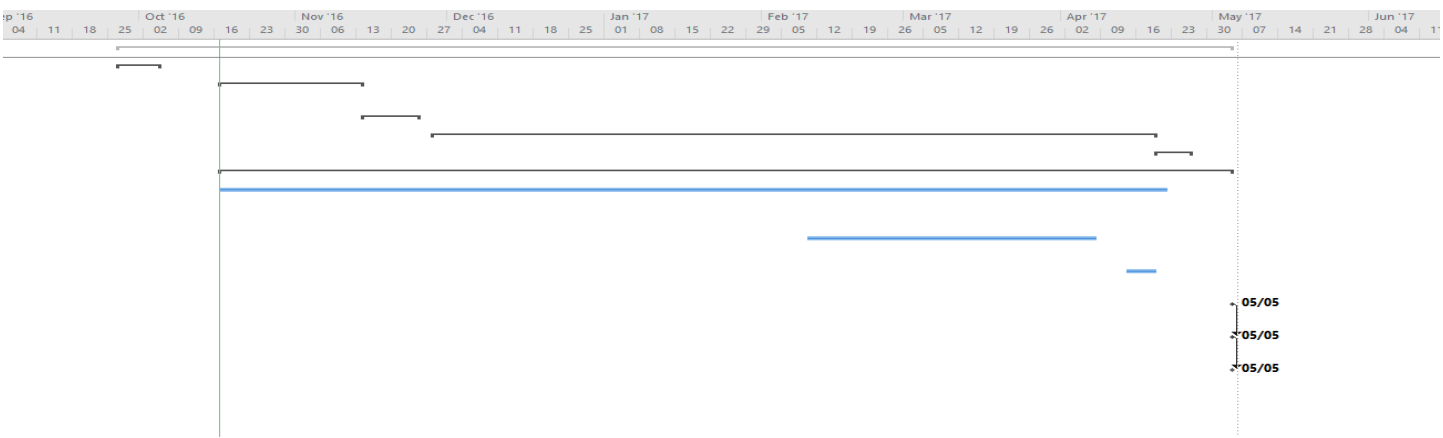
Development	**104 days**	**Mon 28/11/16**	**Wed 19/04/17**	
Review functional requirements specification document	1 day	Mon 28/11/16	Mon 28/11/16	21
Develop early prototype	4 days	Tue 29/11/16	Fri 02/12/16	23
Early prototype Submission	0 days	Fri 02/12/16	Fri 02/12/16	24
Develop application side features	53 days	Thu 08/12/16	Fri 17/02/17	25
Mid-Point presentation	2 days	Fri 16/12/16	Sat 17/12/16	
Develop server side features	53 days	Thu 08/12/16	Fri 17/02/17	
developer testing	3 days	Mon 17/04/17	Wed 19/04/17	28
Development complete	0 days	Wed 19/04/17	Wed 19/04/17	29

Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Analysis/Software Requirements	**20.13 days**	**Mon 17/10/16**	**Mon 14/11/16**	
	Write up proposal	5 days	Mon 17/10/16	Fri 21/10/16	
	Project proposal Due date	0 days	Fri 21/10/16	Fri 21/10/16	8
	Determine Requirements	3 days	Mon 24/10/16	Wed 26/10/16	9
	Write requirements specification document	10 days	Fri 28/10/16	Thu 10/11/16	10
	Receive and analyse feedback on requirements specification from supervisor	1 hr	Fri 11/11/16	Fri 11/11/16	11
	Research required libraries and technologies	1 day	Fri 11/11/16	Mon 14/11/16	12
	Requirements Specification Due date	0 days	Mon 14/11/16	Mon 14/11/16	13
	Analysis complete	0 days	Mon 14/11/16	Mon 14/11/16	14

Documentation	**145 days**	**Mon 17/10/16**	**Fri 05/05/17**	
Create any necessary user documentation	136 days	Mon 17/10/16	Fri 21/04/17	
Write Project report	8.2 wks	Fri 10/02/17	Fri 07/04/17	
Review all user documentation	4 days	Fri 14/04/17	Wed 19/04/17	
Software and Doc upload	0 days	Fri 05/05/17	Fri 05/05/17	
Project presentation	0 days	Fri 05/05/17	Fri 05/05/17	38
Documentation complete	0 days	Fri 05/05/17	Fri 05/05/17	39

Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Design	**9 days**	**Mon 14/11/16**	**Fri 25/11/16**	
	Design wireframes and research UI designs	3 days	Mon 14/11/16	Thu 17/11/16	15
	Research barcode scanning libraries and methods	2 days	Thu 17/11/16	Mon 21/11/16	17
	Incorporate supervisor feedback into software design	2 days	Mon 21/11/16	Wed 23/11/16	18
	Review written software specifications	2 days	Wed 23/11/16	Fri 25/11/16	19
	Design complete	0 days	Fri 25/11/16	Fri 25/11/16	20

Testing	**5 days**	**Thu 20/04/17**	**Wed 26/04/17**	
Complete any integration/Junit testing on application	4 days	Thu 20/04/17	Tue 25/04/17	29
Test with public to insure it is easy to use	1 day	Wed 26/04/17	Wed 26/04/17	32



I had researched early on how to implement barcode scanning into android applications and had come to the conclusion that the ZXING library was the best for what I needed to achieve. My primary language is Java in android studio which will be reaching out to a node.js server doing all the work on the server side. For the prototype, I will be using SQLite until I get the app up and running and then I will use MySQL or PostgreSQL for the database which will be hosted on Heroku.

Heroku will allow me to connect to the database and node.js server free of charge which will be perfect for demonstration purposes. Heroku will need to be deployed each time I wish to use the app but If the application got support and was being used in retail shops I would utilise an independent server to deploy them but for demonstration purposes I will stay with Heroku.

Heroku also has PostgreSQL support which is something I want to get experience in for CV purposes however it does not differ greatly from what Oracles SQL databases act like.

On top of all of this I will implement the simplify payment API into the application which would process payments securely without needing the user to have an account and they simply input their credit card details which are encrypted during the payment process.

Evaluation

For the final stages of the application development process I will begin testing by writing some unit and integration tests to establish that the app will work when put in the Hands of the end user. I have also been researching automated testing techniques using Cucumber and JBehave as they are commonly used now in many large scale organisations.

Once this developer testing is complete I will begin giving APK's to friends of mine to insure the app is easy to use for the end user. I will ensure to give the APK to users that are not good with computers as in the end I intend to create the application for the everyday user not just the technically inclined.

APPENDIX B: USABILITY TESTING FEEDBACK

Date: 03/05/2017

Age Range

☐ 17 or younger ☐ 18 - 29 ☐ 30 - 39 ☑ 40 - 49 ☐ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☐ IPhone ☐ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☐ Good ☐ Bad ☐ Mediocre ☑ Excellent

How easy is it to navigate the application?

☐ Easy ☐ Not at all easy ☐ Moderately easy ☑ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☑ Very fast ☐ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

Prices showed as sterling which is down to no option for English (Ireland) on my phone.

The time stamp on the receipts are out by an hour – Receipt Reference Number 507699 - time stamp of 13.08 but entered at 14.08.

What other features would you want in this application?

1. An option to remember your card details when going to pay

Date: 06/05/2017

Age Range

☐ 17 or younger ☑ 18 - 29 ☐ 30 - 39 ☐ 40 - 49 ☐ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☐ IPhone ☐ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☐ Good ☐ Bad ☐ Mediocre ☑ Excellent

How easy is it to navigate the application?

☐ Easy ☐ Not at all easy ☐ Moderately easy ☑ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☑ Very fast ☐ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☑ Very likely ☐ Likely ☐ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

No

What other features would you want in this application?

It would be nice to be able to email or text my shopping list to someone else in case someone else in my household was doing the shopping instead of me.

Down the line the app could hold multiple language support for people who don't speak English. The app could be set to the default language of the phone.

Date: 04/05/2017

Age Range

☐ 17 or younger ☐ 18 - 29 ☐ 30 - 39 ☑ 40 - 49 ☐ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☐ IPhone ☐ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☐ Good ☐ Bad ☐ Mediocre ☑ Excellent

How easy is it to navigate the application?

☐ Easy ☐ Not at all easy ☐ Moderately easy ☑ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☐ Very fast ☑ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☐ Very likely ☐ Likely ☑ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

Initial use using "Create Hoarder Account" gave me a message 'Email has already been used'.

Login worked afterwards.

What other features would you want in this application?

Date: 05/05/2017

Age Range

☐ 17 or younger ☐ 18 - 29 ☐ 30 - 39 ☑ 40 - 49 ☐ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☑ IPhone ☑ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☐ Good ☐ Bad ☐ Mediocre ☑ Excellent

How easy is it to navigate the application?

☐ Easy ☐ Not at all easy ☐ Moderately easy ☑ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☐ Very fast ☑ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☐ Very likely ☑ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

No

What other features would you want in this application?

Screen Rotation Support would be nice!

Date: 08/05/2017

Age Range

☐ 17 or younger ☐ 18 - 29 ☐ 30 - 39 ☐ 40 - 49 ☑ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☑ IPhone ☐ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☑ Good ☐ Bad ☐ Mediocre ☐ Excellent

How easy is it to navigate the application?

☑ Easy ☐ Not at all easy ☐ Moderately easy ☐ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☐ Very fast ☑ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☑ Very likely ☐ Likely ☐ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☑ Very likely ☐ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

Took me a while to get used to using Android again as I have been using IOS for the last year, but once I got used to it the application worked very well.

What other features would you want in this application?

An IOS version would be great!

Date: 08/05/2017

Age Range

☐ 17 or younger ☐ 18 - 29 ☐ 30 - 39 ☐ 40 - 49 ☑ 50 - 59 ☐ 60 or older

Which of the following devices do you own?

☐ IPhone ☐ IPad ☑ Android Phone ☐ Windows Phone

How did you find the application overall?

☑ Good ☐ Bad ☐ Mediocre ☐ Excellent

How easy is it to navigate the application?

☑ Easy ☐ Not at all easy ☐ Moderately easy ☐ Very easy

How responsive is the application in your opinion?

☑ Very responsive ☐ Not at all responsive ☐ Responsive ☐ Moderately responsive

How fast is it to complete a transaction (From scanning an item to payment) using the application?

☐ Not at all fast ☐ Moderately fast ☐ Very fast ☑ Fast

How likely would you be to use this application if it was available on your device?

☐ Not likely ☑ Very likely ☐ Likely ☐ Moderately likely

How likely are you to recommend this application to others?

☐ Not likely ☑ Very likely ☐ Likely ☐ Moderately likely

Were there any issues you encountered with the application?

No.

What other features would you want in this application?

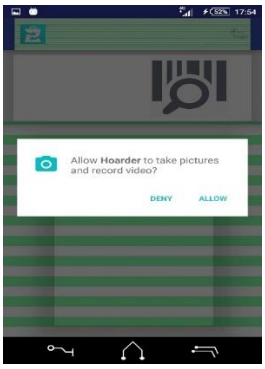
Ability to remember the last user name to log in

APPENDIX C: MONTHLY REPORTS

SEPTEMBER

MY ACHIEVEMENTS

This month I worked hard to make a solid start on my project and assure myself that what I was planning on doing was achievable for the time frame we are all given. I spent the first week getting to grips with Android studio and understanding how to code my activities and style the layout. I had previously done some work with Android studio however, since we had never done it in college until this year I was far from being an expert with the IDE but with the help of online tutorials from sites like android-coffee.com and developer.android.com I was able to develop a very early prototype of my barcode scanner to evaluate the difficulty involved. My first step was to create a splash screen for the application along with a basic page for the barcode scanning button and content text fields. I worked with the ZXING library to implement this and found it to be a very nice library to work with overall. The design of the app overall needs to be severely improved upon and I will need to restructure the activities but I feel overall this month was quite successful in terms of getting work done.



At the end of September, I had a working barcode reader that would scan using the phones camera and display the barcode data (barcode content and format) in two text fields in the main activity.

MY REFLECTION

Overall, this month was very productive and I have achieved a lot. Firstly, I managed to get a working barcode scanner implemented into my application along with a splash screen and basic UI that I will expand on in the coming months.

Although I completed a vital component in the overall application I feel the UI was not to the standard I wanted it to be by the end of the month. I was hoping to have a defined theme for the entire app however I have not finished styling the app bar and other layouts in the application.

INTENDED CHANGES

Next month I will style the app and work on developing a cart system to replace the standard two text fields for the barcode scanner. I will also look at mongo dB or MySQL in creating a catalogue of items linked to various barcode numbers that I can use for demonstration purposes for my presentation in December.

SUPERVISOR MEETINGS

Date of Meeting: 05/10/2016

Items discussed: Project pitch

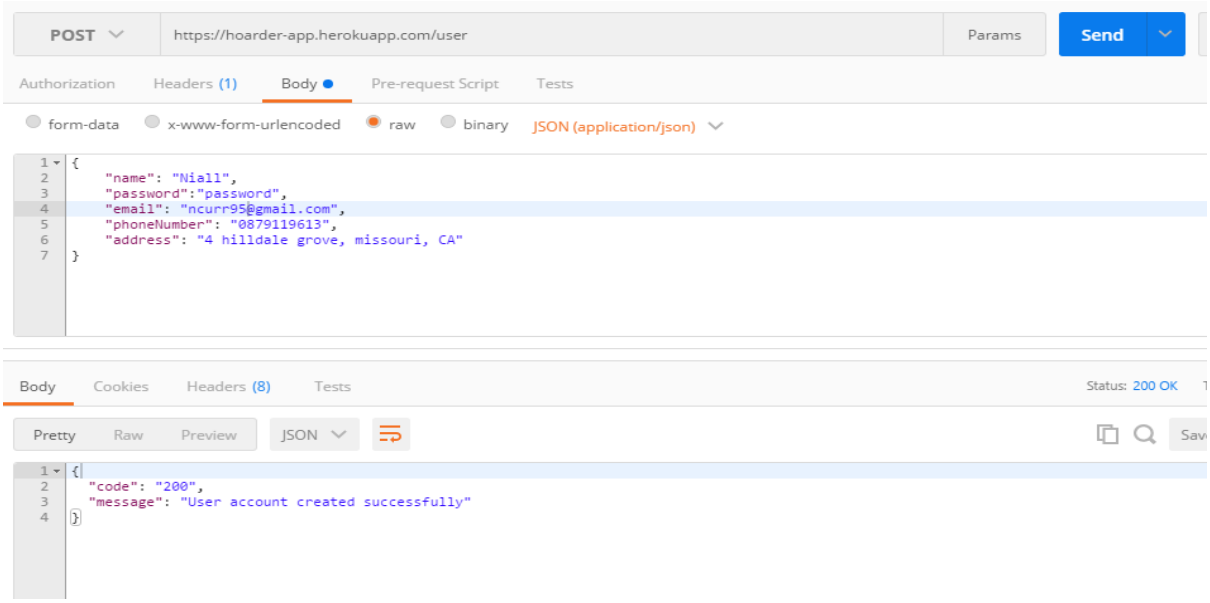
Action Items: Do away with the original NFC functionality and make the application specifically barcode related.

OCTOBER

MY ACHIEVEMENTS

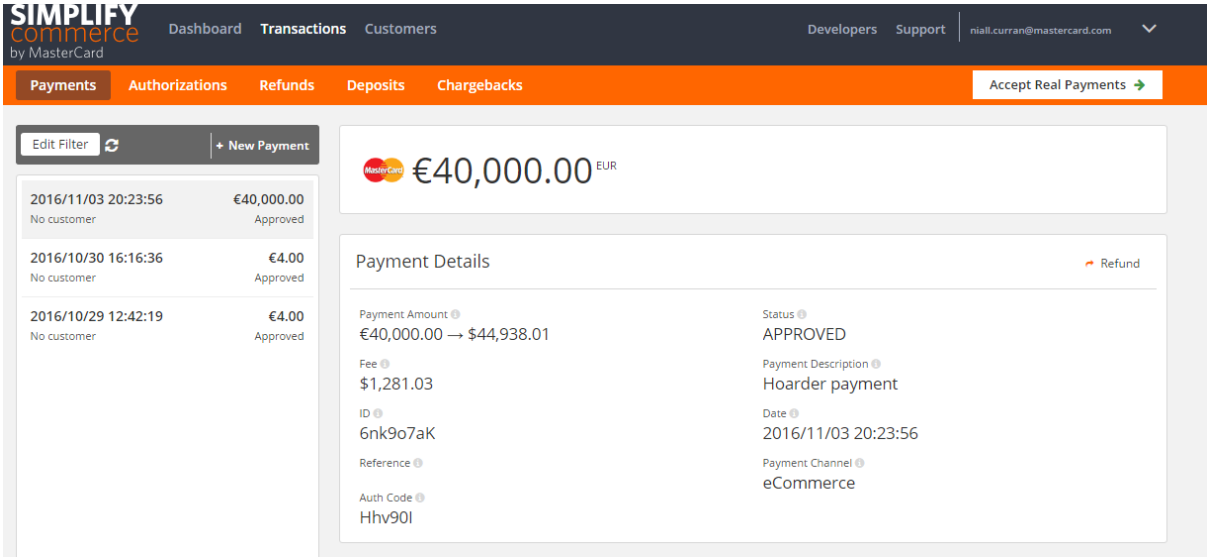
I spent the entirety of October working on getting my project proposal finished and submitted, making a start on developing the server side for my application and starting on my requirements specification document.

The proposal for my project only took a few days to complete and submit so the rest of the month was spent making good progress with my requirements specification and making a start on the server side for my project. The server side was to comprise of a Node.js server with a mongoDB backend database all of which were hosted on Heroku. The code itself was not difficult to implement as I had prior experience in Node.js from my internship in third year. The server was built to accept http requests from the client which I tested using a google application called Postman.



The response given from the server is in JSON format so my next step will be to receive this response on the client side with my android application and parse the JSON to receive the user data.

Along with standard user creation and login I also added functionality for adding an item to the catalog, read item data based on its barcode and payment through MasterCard's simplify payment API. I had configured a developer sandbox for simplify during my internship which will accept payments but with fake account data so actual money will never be actually sent across the system. However, If I decide to bring this application to the market, the API can easily be configured to accept real payments.



As you can see from the above image I can easily monitor any payments made through the app using the simplify dashboard. Simplify also encrypts all credit card information when transmitted and generates a token for that specific transaction which can only be used once. It can also be used for any credit card including visa, MasterCard and American express.

This month I also worked on my requirements gathering and made a start on the requirements document also. For this, I defined some use cases based on my interaction with people I know and who may use the application. From this I will find it easier to define the specific specifications for the application before making a start on the actual android side of the application.

MY REFLECTION

I found this month to be very productive primary due to my entire server side, except for some changes I might end up making towards the end of the year, being complete. Between this month and last month, I managed to get a working barcode scanner implemented into a basic android application, designed some mockups for what I believe the application should look like, finished and uploaded my project proposal document, developed my server side using Node.js and MongoDB and made a start on my requirements specification.

INTENDED CHANGES

Next month I will be primarily working on my requirements specification document as well as defining the style of the application as I believe the UI will play a large role in my application. Once I complete my requirements specification I will also begin modifying my basic android application into one that can send and receive http requests to my server side and parse the JSON it receives into viewable UI data.

SUPERVISOR MEETINGS

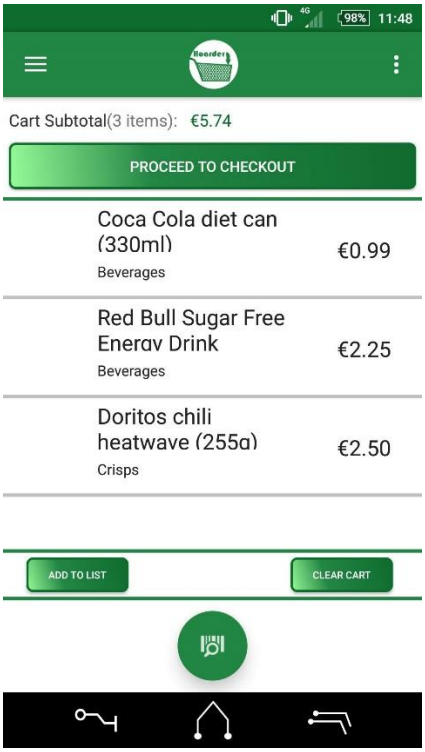
Date of Meeting:	10/11/2016
Items discussed:	First Meeting with supervisor organised
Action Items:	TBD

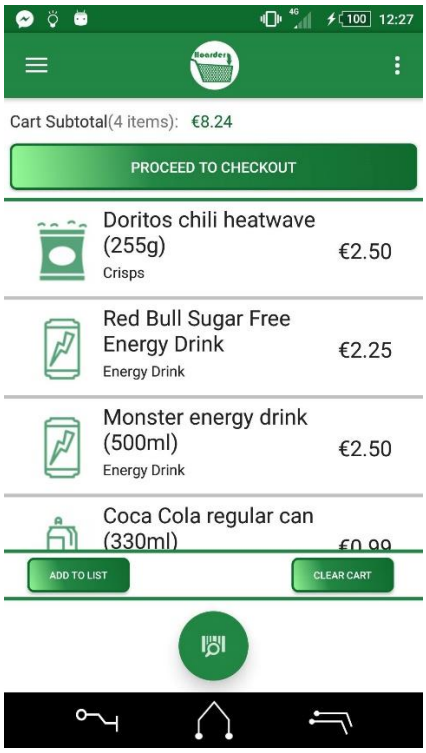
NOVEMBER

MY ACHIEVEMENTS

I spent the entirety of November working solely on developing my prototype and cleaning up some of the server side code for my project. Towards the end of the month I was working to get my technical report complete along with my many other projects not related to this project and felt I made a good start on it but I will definitely need to make improvements on it throughout all of next semester.

Given that my server side is mostly complete, besides making the code neater and adding in a little more functionality, I made a lot of progress on my app to mimic this. My application can now scan items, receive the scanned item data and add it to the user's cart.





From the above pictures, you can see that a number of items have been added to the cart using the scanning functionality and the user can see each of the items displayed as well as the number of items and the cost of everything in their cart. For the Midpoint presentation, I will look at having the "Proceed to checkout" button functional in sending the cart data to the checkout page while also prompting the user to input their card details before sending the data to my server.

MY REFLECTION

I spent most of my time this month working on the application side and I have gotten a lot of the base functionality completed and am now making progress on cleaning up the server side as well as allowing a payment to be made on the application side.

INTENDED CHANGES

Next month I will be primarily working on getting my technical report up to date as well as continuing to develop my prototype so that I have as much functionality as possible on the day to present. My goal is to have the application working to a point where the user can scan a barcode, receive the item data for that barcode and add it to his cart where they can see the number of items in their cart, the total cost and have the option to proceed to checkout which will bring them to a payment form. Once I have this all done I will need to create a PowerPoint presentation which will help me convey what I want to achieve with this project and the progress that I have made for the Midpoint presentation.

SUPERVISOR MEETINGS

Date of Meeting:	10/11/2016
Items discussed:	First Meeting with supervisor
Action Items:	It was decided that I should look at incorporating my original idea of using NFC as a part of the application once I have the base functionality in place. This would add a larger IoT element to my project and set my project apart from many others. The next meeting was set up for the 8th of December where we would discuss the Midpoint presentation and our working prototypes.

DECEMBER

MY ACHIEVEMENTS

December was a rather stressful time with multiple projects having to be uploaded so I did not achieve what I had hoped to achieve with the project this month which was getting the payment functionality to work on the application side. However, I was still able to demonstrate a successful payment using Postman using the server side endpoints in the mid-point presentation which I was rather happy about.

I spent a lot of December trying to get the prototype up and running (Which I had mostly finished in November) and compiling a list of slides which could help me explain my idea for the presentation.

MY REFLECTION

Overall, this semester has worked out rather well and I have gotten a lot of the core functionality for my application and server completed however, I recognise that there is still a lot of work left to do and I will be focusing a lot of my spare time over the Christmas holidays to working on getting the payment system working on the application side as well as implementing the rest of the required functionality.

INTENDED CHANGES

Now that the Mid Point presentation is finished I will continue to work on the remaining features for both the server side and the application side to have the project complete for the final presentation. In the presentation, I received a lot of good feedback about what I was doing and had a few further modifications to the project pitched by my supervisor which could be interesting. I was given the idea to one individual shop as opposed to multiple shops and simply create a new instance of the application for each individual shop. The loyalty system will also be altered to store a monetary discount as opposed to a point system which will be a percent of the total cost of an order will be added to a user's wallet which would vary depending on the store itself.

SUPERVISOR MEETINGS

Date of Meeting:	20/12/2016
Items discussed:	Mid-Point presentation
Action Items:	With the successful completion of my prototype which showed the core functionality of the android and server-side application and the submission of my Technical report I was able to give a very in depth and

JANUARY

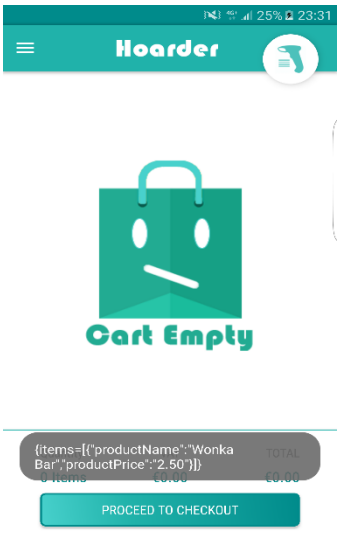
MY ACHIEVEMENTS & REFLECTION

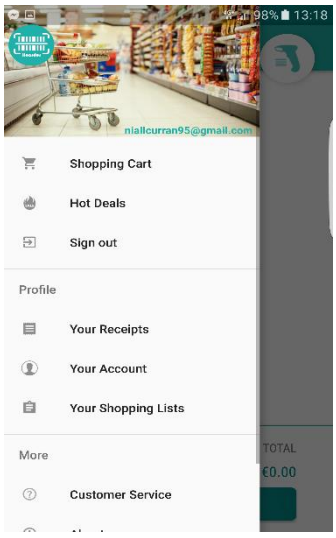
After a very well received mid-point presentation, I felt I was heading in the right direction with my application. Once the exams had concluded in early January I began extensive development of my application side in android studio while also making some good modifications to my server side to allow for the receipt, shopping list and payment functionality of the application.

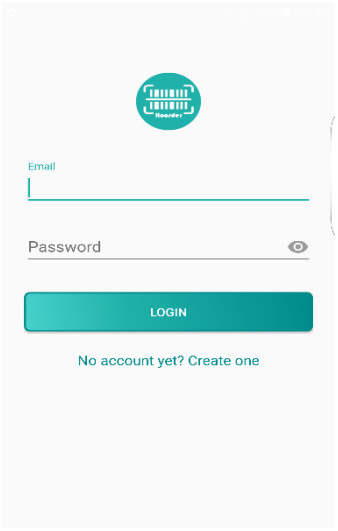
I started by making UI modifications as I felt the colour scheme as well as the appearance of the home/shopping cart screen seemed relatively outdated and required a more modern look to draw the users eye and make it look like a far more commercially viable application. I implemented additional UI features such as images for when the users cart is empty or if he has no current orders at that store for the receipt screen.

My first priority with implementing the remaining functionality was to ensure the standard path of the application (Being able to scan items, checkout, make the payment and have a receipt generated for that payment) was completed to my original specification. Once this was complete I worked on the profile screen which displays the user information, the receipt screen which displays the users receipts for every purchase at that store and the shopping list screen which allows the user to add specific items to a list which is associated with their account on the server side. All of these changes required extensive development of the server side which spanned over the entire month so I still have to update my documentation as well as complete testing on the application by implementing unit tests into the app as well as generating APK files for people to test the application and report to me their experience.

Authentication for the application on both the server side and application side has also been completed with the exception of encryption which I will be using a node.js library with a custom cypher to achieve. When this is implemented in the next few weeks all of the passwords in the database will be encrypted before being transmitted to the database and decrypted only to compare the password to user input on login.







Overall, this month was the most productive as all of the ground work was already complete in the previous few months however, I feel I am in a very good position at the moment as a large amount of my project is complete and now I can focus more on the documentation and testing side to my application.

INTENDED CHANGES

Next month I will be focusing on encryption of the passwords in the database as well as some documentation and testing work. I will also look to finish my work on the receipts screen to allow the user to see all of the receipt details in the database as currently they can only view their receipts in a list with limited information displayed.

system which will be a percent of the total cost of an order will be added to a user's wallet which would vary depending on the store itself.

SUPERVISOR MEETINGS

Date of Meeting:	03/02/2017
Items discussed:	Mid-point presentation marks breakdown and analysis of where I may have lost marks.
Action Items:	Focus on testing

FEBRUARY

MY ACHIEVEMENTS & REFLECTION

This month was spent researching encryption techniques for Node.js to ensure passwords and sensitive user data is encrypted whilst resting in the database. My research was very successful as I am now confident enough with this method of encryption to be able to implement it into my project server next month. On top of this I made minor aesthetic changes to pages in the application as well as making good progress in adding functionality to the remaining pages that I have not added functionality to and extending on the receipt page which I had started last month. The application is now coming together and I feel it will be completed and ready for testing towards the end of next month where I will begin generating APK files to distribute to multiple people to get feedback as a method of usability testing.

INTENDED CHANGES

Next month I will be focusing on implementing the encryption, finishing the application to its entirety if possible and beginning testing the application (Usability testing) and ensuring I have researched and implemented some Unit testing into both the application and server side.

SUPERVISOR MEETINGS

Date of Meeting:	09/03/2017
Items discussed:	Discussed final presentation and progress report on the project
Action Items:	None

MARCH

MY ACHIEVEMENTS

This month I worked primarily on the security, testing and UI of my application while also tidying up my server side node.js code base. I used a node.js encryption/decryption library that will allow the app to securely store the details of the user in transit between the server and the database. I have also worked on the testing of my application by experimenting with android unit tests and now I feel I can implement more tests to cover the entire application. While writing unit tests I have also started distributing APK files to different people that I believe would be able to give me constructive feedback on the quality of my application. Towards the end of the month I started adding the remaining functionality to some of the smaller pages such as the Locations page and the deals page and adding additional functionality to the receipts page.

MY REFLECTION

This month may not have been as productive as previous months but I feel that I have completed a vast majority of the application thankfully so for April and May I will be finishing testing of the application and finalising the look of all the screens.

INTENDED CHANGES

Next month I will finish my styling of the application, cleaning up of the server code base, testing (Both unit and usability testing) and ensuring my report is up to date and ready to be submitted in May.

SUPERVISOR MEETINGS

Date of Meeting:	TBD
Items to be discussed:	Full application demonstration
Action Items:	TBD