Mid Term Delivery

Enterprise Computing Team Project

CA472

Name: Jamie Behan
Student Number: 18406986
Email: Jamie.Behan6@mail.dcu.ie

Name: Bernard McWeeney
Student Number: 18384466
Email: Bernard.McWeeney2@mail.dcu.ie

Name: Shaun Kee
Student Number: 18308546
Email: Shaun.Kee2@mail.dcu.ie

Submission: 14/01/2022
Programme: EC4

Table of Content

1. Introduction

1.1. Overview

SCAN-N-GO is an application whereby a consumer can enter a retail store, scan a barcode of any product(s) and pay utilising their phone's built-in payment system (Apple Pay, Google Pay etc.). This will allow consumers of fast paced retail stores to bypass traditional queuing systems in-store hence expediting their time in-store. SCAN-N-GO will operate within a two-sided business model. We have identified two primary customer segments - owners of fast paced retail stores and customers of fast paced retail stores. For each of these customer segments we have defined two different value propositions that we intend to deliver. Our solution for owners of fast paced retail stores aims to increase store profitability by reducing staffing costs by up to 50% and increasing store revenue by up to 10%. Following on from this, our solution for consumers of fast paced retail stores aims to decrease in-store queuing time by up to 80%.

For consumers of fast paced retail stores we will provide a web application that can be accessed via their mobile device. This application will be the catalyst for scanning barcoded goods and making quick purchases of goods in-store. This application will allow the consumer to pay for one or many goods simultaneously.

For owners of fast paced retail stores we will provide an admin interface which will be setup within the retail store to enable in-store staff to monitor the activity of the SCAN-N-GO users that are currently in-store. This admin interface will display information such as how many users are in-store, the names of users in-store, what goods each user has scanned in-store as well as a notification of when purchases are made. This information will act as an increased security measure for the retail store.

1.2. Business Context

Due to the nature of SCAN-N-GO's business model both customer segments (owners of fast paced retail stores & consumers of fast paced retail stores) can be viewed and discussed in isolation as well as in tandem within a business context. For example, revenue streams as well as the method of delivery for each of their value propositions differ greatly between both customer segments. There is however overlap in the target market size.

Our total addressable market is the global retail market which is estimated to be approximately 20 trillion USD in size. Initially, we will have a core focus on the Irish market for grocery / convenience stores which was estimated to grow to €19.8 million in 2021. We can also expect a compound annual growth rate of 4.5% due to the maturity of the market. (See Appendix 1)

Consumers of fast paced retail stores

The SCAN-N-GO application will be available for free to consumers of retail stores. It will be made accessible via any web browser. The only requirement for consumers is that they must make an account in order to utilise the service. We aim to monetise this particular customer segment through the use of In-App advertisements. We believe that allowing consumers to use our app free-of-charge will lead to higher rates of adoption and usage. This increased volume will lead to a higher amount of advertisement revenue.

Aforementioned, we aim to decrease the in-store queuing times for consumers by up to 80%. Through our primary research we believe this is a metric we can deliver upon. We intend to achieve this by two primary methods. We remove the need to queue in-store to interact with a cashier or self-service till due to our application allowing customers to pay using only their phone. We also achieve our value proposition by ensuring that the in-app payment system is as quick and seamless as possible. (See Appendix 2)

Owners of fast paced retail stores

Our two primary revenue streams will be targeted towards this particular customer segment as there will both be a transaction fee of 1% on every purchase completed in-store charged to the owner of the respective retail store as well as an initial integration cost for businesses that wish to use our SCAN-N-GO service. By having a 1% transaction fee on all transactions it allows us to greatly benefit from scaling and onboarding more customers. The initial integration cost for businesses will be nominal fee initially as we aim to encourage early adopters to utilise our service. This initial setup fee will scale as SCAN-N-GO becomes a more widely used service.

As previously mentioned, we aim to increase store profitability by decreasing staffing costs by up to 50% and increasing store revenue by 10%. We will achieve this through 4 primary actions.

- **Less Staffing:** SCAN-N-Go enables the consumer to become the cashier hence drastically reducing the amount of staff required in order to operate additional tills in-store.
- **Less Congestion In-Store:** SCAN-N-GO allows customers to more quickly and easily make purchases. This will lead to more efficient customer turnover hence leading to less congestion in the store.
- **Customer Retention:** As customers continue to utilise our service they may favour stores which use our service in turn increasing the customer retention rate.
- **Increased customer turnover speed:** SCAN-N-GO quick and seamless payments will allow customers to more quickly enter and exit the store.

2. General Description

2.1. System/Product Function

Listed below is our preliminary system / product functions, with a brief description of the function behaviour. We go into more detail about each function's characteristics in section 3 of this document. The functional requirements of our Product / System, specifies what the system is expected to do.

Our User Functionality

Registering
The user is required to sign up for an account to gain access to our WebApp. Signup information collected: User Name, Password, Email, Address, Credit Card details and age. We use age to verify if the user can purchase age restricted products.

Login
To gain access to the WebApp the user needs to login with their login credentials to scan and purchase a product.

Scan In / Out from Store
The user needs to “Scan-In” to a store to load the store's products and “Scan-Out” to confirm the user is no longer in the store.

Product Barcode Scanning
For the user to purchase an item, the user will need to scan the product barcode using the mobile app.

Add to Basket / Remove from Basket
The user can update their product basket by removing an unwanted item from their basket or by adding additional product quantities.

Checkout Basket
For the user to be able to check out their basket, they can pay by using their smartphones' built in payment system like Apple Pay / Google Pay.

View Orders
The user can view their current and previous shop orders, it will also include a digital receipt for proof of purchase.

Admin Dashboard - Active Users
The Admin User can view currently active users in their store. This will help Admins monitor shop security.

Admin Dashboard - Users Orders
The Admin User can view users orders from the admin dashboard to help verify customers purchases.

Our System Functionality

Web Application Design
We aim to develop a user friendly Web Application for both admins and general users, which is simple and intuitive to navigate and use.

Processing Search Results from Barcode Scan
We require the user to scan the barcode of their desired product and we require a connection with the store's point of sales system to provide the user with an accurate price and product information.

Basket Checkout & Order Creations
We require the user to provide access to their smartphone built-in payment system or their credit card information to complete the basket checkout & order.

Database Connectivity
We have to connect our database with the shop's Point of sales system to fetch product information and we have to connect our web application to our database to update and retrieve data for both our admin dashboard application and our general user application.

2.2. User Characteristics and Objectives

Consumer Web Application
Our SCAN-N-GO web application is designed for consumers of retail stores to use quickly and efficiently when making their in-store purchases. With minimal setup requirements i.e Account registration. It will enable consumers who are in a rush or have little time to make purchases in-store quickly regardless of the current congestion status of the store. The application once finalised will operate in a way that allows the customer to very quickly scan a product and checkout with minimal distractions in-app.

Retail Store Interface
Our SCAN-N-GO retail store interface is designed to allow staff within the retail store to easily digest the on-screen information at a glance catering to a fast paced retail environment's expected workflow. Access to legible information at a glance will allow a member of staff to monitor in-store activity of SCAN-N-GO users as well as tend to their other duties in-store.

2.3. Operational Scenarios

General Consumer
For a customer to use our SCAN-N-GO web application, they must register or login with their details to gain access to the app. The Register / Login page is the first page displayed. If they select Register, they will be required to fill out a form asking for their Name, Password, Age, Email, Address, Credit Card details.
If they select Login, they will be required to enter their username and password. If their credentials are correct, they will proceed to the apps homepage, otherwise they will receive an error message asking them to try logging in again.
The homepage will consist of their phone's camera view and a navigation menu. Once the user is logged in, they will be able to scan into a participating store to “check in” using their phones built in camera. Once checked in, the user will be able to scan a shop's product to add to their basket, again using their phone's built-in camera.
The user can then view their basket and update / remove items that they scanned in. If the user is happy to proceed with the purchase of their basket, they can do so

by checking out by confirming their order and paying using their phones built-in payment integration such as Apple Pay.
When the user has completed their transaction, they can provide proof of purchase by viewing their Orders tab. Finally they can exit the shop, by “checking out” of the store by scanning the checkout store code.

Shop Administrator
For a Shop Administrator to use our SCAN-N-GO web application, they must register or login with their details to gain access to the app. The Register / Login page is the first page displayed. If they select Register, they will be required to fill out a form asking for their Name, Password and Email.
If they select Login, they will be asked to enter their credentials. If the credentials are correct they will be able to see the admin dashboard.
The homepage of the admin dashboard will consist of a list of active SCAN-N-GO store users. If the admin needs to verify an accurate purchase, the admin can select an option to view an active store user's shopping baskets items from the dashboard.
The admin dashboard also has a settings page for connecting to the store's POS API and for updating the admin users account details.

2.4. Constraints

Time
Our SCAN-N-GO application is an ambitious solution to both retail stores and consumers of retail stores. We have a strict time cut-off due to our project deadline.

People
Whilst our intention is to make our service as user friendly as possible. There may be a learning curve involved with using our service in particular on the retail store side as they must adapt to the interface.

Scalability
Our current objective is to establish a service which works seamlessly with a small face paced retail store, our long term goal would be to expand to larger and different types of retail stores. This could introduce a scalability issue however we do not anticipate this to be an issue for the foreseeable future.

Real time connectivity between systems
We intend to have real-time connectivity and communication between our web application and retail store / admin interface. We anticipate this particular aspect of our service to be a challenge to get fully functional.

3. Functional Requirements

3.1. Register & Login

Description:
This functional requirement requires the platform to allow a new user to register for SCAN-N-GO and also allow a known user to login if they already have an account. Our platform will require a login from a known user to be accessed. The registration form requires an Email Address, Username, Password, Date of Birth and Home Address.

Criticality:
A user must be logged-in to use the platform as we need this user information for security reasons and also to provide saved data in-app.

Technical:
The registration and login forms will be created using HTML and the data will be sent to the backend to verify once the form is filled out and submitted. This data will be sent via REST API through a Javascript function. The Backend will verify logins and respond with an access token, this will then be stored locally on the customers device and allow them to access the platform. For new users, the data will be validated on the front end and sent to the backend to be processed. Once the registration data is sent to the backend and successfully processed the new user can login with the details they provided.

Dependencies:
This function requires that the backend system is live and running to process the data sent from the frontend.

3.2. Store "Scan-In"

Description:
Once a user is logged in, they must declare to the application what enrolled store they are currently in. To do this, the user must scan the store's unique SCAN-N-GO store barcode which is located primarily on the front door of the store. Once a user scans the store's barcode, they will be able to proceed with the purchasing process.

Criticality:
A user must declare what store they are shopping in so that the platform knows which store to retrieve stock data from and register the purchase with.

Technical:
Once the user scans the store barcode the user will be associated with that store in the database until they checkout with a purchase or simply time out. When a user is associated with a store, all in-app actions will be associated with that store's product Database. There is the potential in the future to automatically detect which store they are in using geo-location.

Dependencies:
This function depends on the user being logged-in to the platform.

3.3. Barcode Scanning

Description:
On the homepage, the user will be able to scan the barcode of a product within the store to display its product information (Price, Product Name etc.). This action will take the user to the product view (See Functional Requirement 3.4).

Criticality:
This is the most essential feature of the application as it is our company's Unique Selling Point. Without this feature the application must use a more manual user input to select the product they are looking for. Allowing the user to retrieve product info from simply scanning a product barcode is extremely efficient.

Technical:
We will use the HTML5 hardware API to access the customer's camera on their smartphone. From there will use Javascript to decode the product barcode into a string of numbers. Once the barcode is decoded it will be sent to the backend server and the user will be redirected to the product view.

Dependencies:
This function requires the user to have a working camera on their smartphone and also be logged into both the platform as a known user and also be scanned into the store they are shopping in.

3.4. Product View

Description:
The product view simply displays to the user the product information, and allows the user to choose a quantity and add the product(s) to their basket.

Criticality:
This function is critical as it is the only way a user can add a product to their basket.

Technical:
A user will be redirected to the product view after scanning a barcode. The url redirection will simply contain the product ID as the query parameter which will indicate to the frontend what product to load onto the product view page.

Dependencies:
This function requires the user to be logged in.

3.5. Add To Basket / Remove From Basket

Description:
The Add & Remove from basket functionality will add or remove a product from the basket when the user chooses to add / remove a product using the front end.

Criticality:
This function is critical as the user may want to delete items from their basket while the Add-to-Basket function is obviously a standard piece of ecommerce functionality.

Technical:
When a user chooses to Add or Remove a product from their basket they will be calling a REST API at the backend server (either /add or /remove). This will take a product ID and quantity as a parameter. The backend server will then add or remove the quantity of that product from the basket and the frontend will be refreshed.

Dependencies:
The remove function requires that the product already be in the user's basket before being removed. The add function can only be used on products that are in stock in the store.

3.6. View Basket

Description:
The view basket function will display the current user's basket information to the user. This will include the products in the basket, the quantity of products, the prices of products and also provide an option to checkout.

Criticality:
This function is critical as it is a final verification to the user of what's in their basket before checkout and ultimately allows the user to proceed with the checkout.

Technical:
When the user goes to the /basket page, the frontend will call the backend for basket information using a REST API call. The only parameter that the request takes is the user's token to authenticate. The response from the backend is a JSON object containing all basket information.

Dependencies:
This function requires the user to be logged in and currently have items in their basket (If no items are in the basket then an empty basket page is shown).

3.7. Checkout Basket

Description:
The checkout function takes all the basket information and converts it to an order. This step requires payment details from the user to complete.

Criticality:
This function is critical as the checkout function is ultimately the function that allows the user to pay for their order and complete the buying process.

Technical:
Once the /checkout API is called from the Backend server the checkout information is automatically converted into an order.

Dependencies:
This requires the user to be logged in and also have products in their basket. The card transaction must be successful to successfully checkout.

3.8. View Orders

Description:
A user can view all their previous orders by viewing the /orders page. This will display all previous orders to the user with all relevant order info (in chronological order). The most recent order placed will be the first order immediately viewable to the user. This will also act as a digital receipt for proof of payment, if the customer is asked to provide proof of purchase before leaving the store.

Criticality:
This is an essential piece of functionality as it can be used from a security perspective to provide the customer with a proof of purchase, which can be shown to a member of staff / security before leaving the store.

Technical:
When a user goes to the /orders page an API call is made to the backend server to retrieve all order information. The order information is sent as a response from the backend server and displayed on the frontend using Javascript.

Dependencies:
This requires the user to be logged in and have previous orders associated with their account.

3.9. Admin Dashboard

Description:
The Admin dashboard provides admin users (Store Owners & Employees) with an admin system that allows for customising their SCAN-N-GO store configuration and also allows viewing of live SCAN-N-GO store data.

From the Admin Dashboard an Admin User can:
- See how many users are currently in their store and using SCAN-N-GO.
- See live basket information for users currently using SCAN-N-GO in store.
- Update stock information.
- Update Admin User details.
- Update POS integration settings.

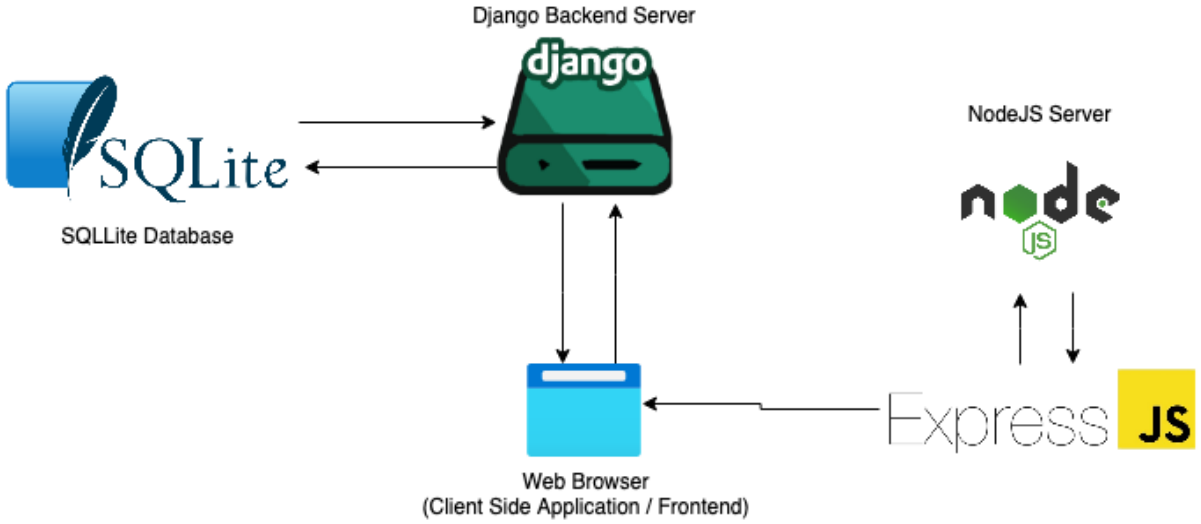
Criticality:
The dashboard is critical for security as it gives the Admin Users an overview of all SCAN-N-GO users currently in store at a glance while also providing live basket information for these users. The dashboard is also critical to develop as it is the only place where Store Owners or Employees can update stock info, it also gives the business the ability to update and edit their current POS integration settings.

Technical:
The admin dashboard will call the backend server via API call to retrieve all users currently active in the store and their basket information. This will then be displayed on the front end dashboard using Javascript and HTML / CSS.

Dependencies:
This requires the store user to be logged in using the Admin credentials. The store must be enrolled for SCAN-N-GO.

4. Software Architecture

4.1. Architectural Diagram



4.2. Components Of Architecture

SQLite Database

Our SQLite database will be our primary database. It will hold all the tables shown in the Logical Data Model. It will be hosted on the same machines as the Django Server.

Django Backend Server (Running Django REST Framework)

Our backend server will be the backbone of our web app. It will be the component that will compute essentially all the logic. The server will retrieve requests from the front end and return a response from the SQLite Database.

Client Side Web App (The Client Web Browser)

The Client's web browser is where they will physically interact with our platform. The client's browser will render the pages provided by the ExpressJS framework and will call on the backend API through a modern UI.

NodeJS

The NodeJS server will serve web pages to the user.

ExpressJS

ExpressJS allows for the dynamic filling of web content on the pages that nodeJS will serve to the clients browser.

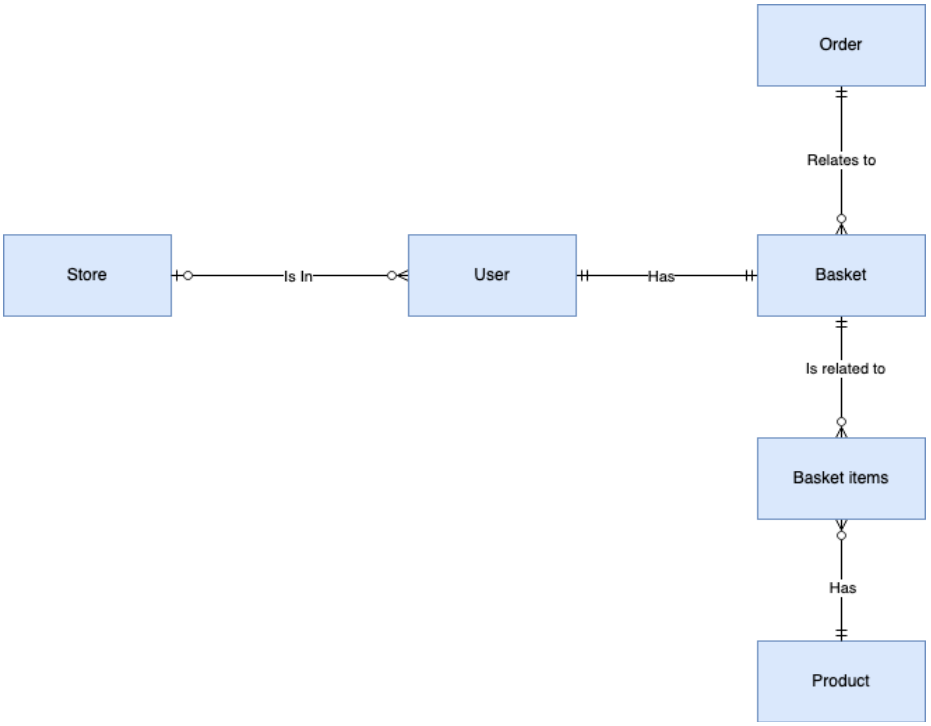
Docker Containerisation

Docker will containerise the frontend and backend of SCAN-N-GO. This will allow us to package up our application and all its dependencies in the two docker containers (one for Frontend and one for Backend). Our application will be hosted on Microsoft Azure.

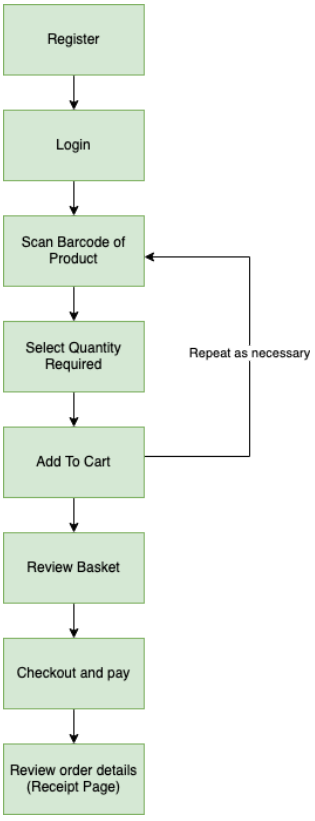
5. High-Level Design

5.1. High Level Design Diagram

Logic Data Model



Functional Layout



6. Prototype Delivery

For our prototype we will develop **SCAN-N-GO Mobile** (The End-User Application) & **SCAN-N-Go Admin** (The Admin Application). Both these applications are completely different in both functionality and purpose but will both run on the same servers as each other and incorporate an identical architecture (See 4.1 Architectural Diagram)

Our **SCAN-N-GO Mobile** prototype will allow an End-User to:

- Login / Register for SCAN-N-GO
    - Non logged in users will not be able to access the platform
- Scan a product barcode to display product information
- Add a product to the basket
- Remove an item from the basket
- View the basket
- Checkout
- View digital receipts / previous orders
- Logout

Our **SCAN-N-GO Admin** prototype will allow an Admin User to:

- See how many users are currently in their store and using SCAN-N-GO
- See live basket information for users that are currently using SCAN-N-GO in store
- Update stock information
- Update Admin User details

6.1. Software Development Challenges

Barcode Scanning

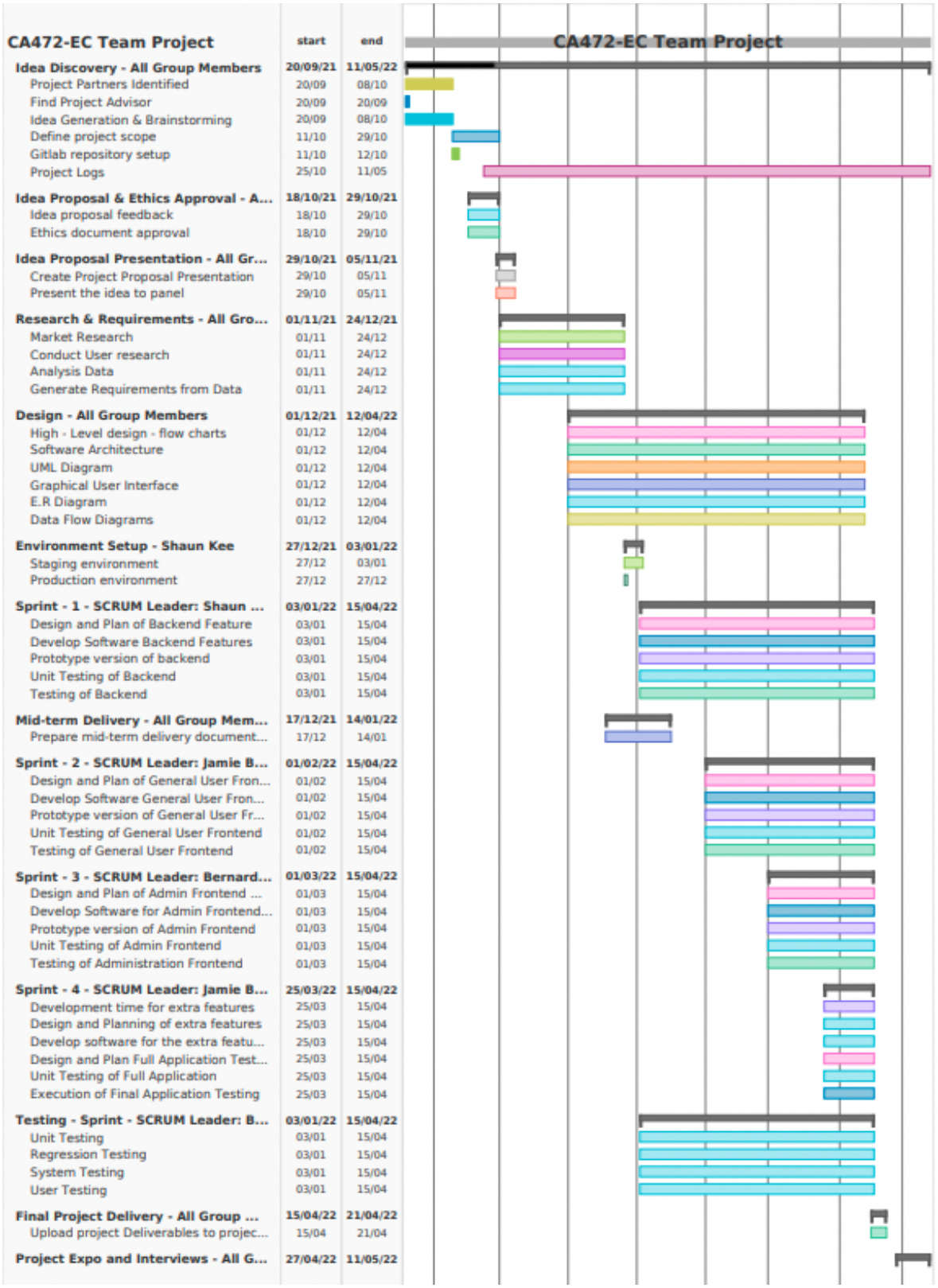
One of the biggest development challenges that we will inevitably face on this project is trying to enable barcode scanning on the homepage of the web app. Seeing as barcode scanning is a fundamental part of our web app, we will have to get this feature integrated and working for our prototype. First we will explore integrating a pre-built JavaScript library into our frontend but if that fails we will try to create our own barcode scanning library.

Live User / Basket Information

We believe that another challenge we will face during development will be trying to display ***live*** user information and basket information in the admin section of the SCAN-N-GO system. We believe that optimising API response times and well written javascript code will allow us to overcome this challenge.

7. Preliminary Schedule

7.1. Gantt Chart (Fig 7.1)



- Bernard McWeeney - Technical Leader - Administration Application
- Shaun Kee - Technical Leader - End-User Application
- Jamie Behan - Commercial Leader - Front-End Developer

8. Glossary

8.1. WebApp
A WebApp is simply an abbreviation of web application. A web application is application software that runs on a web server.

8.2. HTML
The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser.

8.3. JavaScript
JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

8.4. Django
Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.

8.5. NodeJS
Node.js is an open-source, cross-platform, back-end JavaScript runtime environment and executes JavaScript code outside a web browser.

8.6. ExpressJS
Express.js, or simply Express, is a back end web application framework for Node.js.

8.7. Docker
Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

8.8. SQLite
SQLite is a relational database management system contained in a C library.

8.9. REST Framework
Django REST framework is a powerful and flexible toolkit for building Web APIs.

8.10. JSON
JSON is a text-based way of representing JavaScript object literals, arrays, and scalar data.

8.11. API
An application programming interface (API) is a connection between computers or between computer programs.

8.12. POS
A point of sales (POS) system is a connection between multiple checkout terminals, barcode scanners, weighing scales and payment terminals. These devices are all connected to a host computer providing instant and accurate product data.

9. Appendices

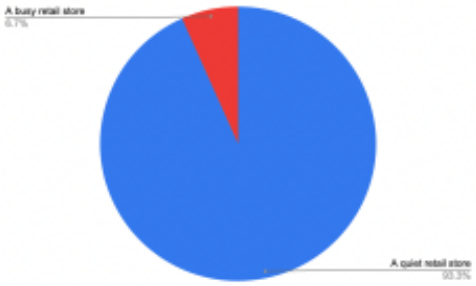
9.1. Appendix 1

https://www.ibisworld.com/default.aspx

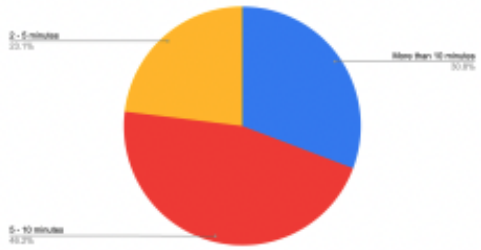
9.2. Appendix 2

Primary Research - Mobile App Users

For our primary research we surveyed potential users of our mobile application and interviewed owners of retail stores.

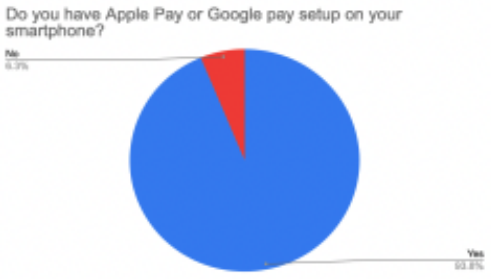


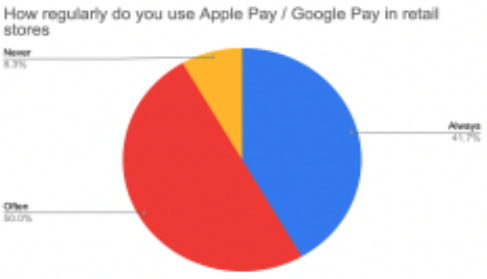
When we asked the potential users of our mobile application to show their preference in shopping in either a busy retail store or quiet retail store, the choice was clearly leaning in the direction of a busy retail store as over 90% of people chose a quiet retail store over a busy retail store. This gives us a fantastic indication to the preferences that people have and backs up the fact that having a quieter retail store will ultimately lead to a more appealing shopping choice and thus, an increase in sales.



We also surveyed these potential users with how long they think they spend in queues on average in retail stores on a daily basis. The results were shocking, nearly 50% of people said they think they spend 5-10 minutes queuing on a daily basis. Through some secondary research we learned that people spend on average up to 47 days queuing over the course of their lifetime.***("British people spend 47 days queueing over their lifetime, poll claims," 2019)***. This clearly shows a suitable place in the market for our solution!

We also asked potential users if they have Apple Pay or Google Pay setup on their smartphone and if they use these payment methods in retail stores. The results show that clearly a very high percentage of people have these installed on their devices and they actively use them in retail stores





Finally, we asked potential customers the all important question, would they rather use self service or a cashier in a retail store. The following chart shows the results, whereby there is a distinct preference for self service retail when compared against interacting with a cashier

10. References

- IBISWorld - Industry Market Research, Reports, and Statistics [WWW Document], n.d. URL https://www.ibisworld.com/default.aspx (accessed 12.1.22).
- Django Introduction - [WWW Document], n.d. URL https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (accessed 13.1.22).
- Django REST Framework - [WWW Document], n.d. URL https://www.django-rest-framework.org/ (accessed 13.1.22).
- API - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/API (accessed 13.1.22).
- JSON - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/JSON (accessed 13.1.22).
- SQLite - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/SQLite (accessed 13.1.22).
- Docker (Software) - [WWW Document], n.d. URLhttps://en.wikipedia.org/wiki/Docker_(software) (accessed 13.1.22).
- ExpressJS - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/Express.js (accessed 13.1.22).
- NodeJS - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/Node.js (accessed 13.1.22).
- Django (web_framework) - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/Django_(web_framework) (accessed 13.1.22).
- JavaScript - [WWW Document], n.d. URL https://en.wikipedia.org/wiki/JavaScript (accessed 13.1.22).
- HTML- [WWW Document], n.d. URL https://en.wikipedia.org/wiki/HTML (accessed 13.1.22).
- Web_application - [WWW Document], n.d. URLhttps://en.wikipedia.org/wiki/Web_application (accessed 13.1.22).
- POS - [WWW Document], n.d. URL https://erply.com/pos-system/ (accessed 13.1.22).