# Proximity Marketing Application

Development Journey Document (Semester 2)

Shane Walsh

20079607

Supervisor: David Drohan
2nd Reader: Catherine Fitzpatrick

BSc (HONS) in Software Systems Development

## Acknowledgements

I would like to take this opportunity to thank my supervisor David Drohan for all his advice and support throughout the year. Without his insightful feedback and creative suggestions this project would not be possible.

Along with my fellow students, I would also like to thank my SSD lecturers and WIT staff for helping me along the way and making the journey as smooth as possible.

I would also like to gives special thanks to my girlfriend Aoife who always steered me in the right direction with positivity and kindness. And also my mother Josephine and sister Jen who were always there when I needed help.

# Proximity Marketing App:
## Development Journey

17.04.2018

Shane Walsh
20079607
Software Systems Development
Final Year Project - WIT

# Table of Contents

# 1. Introduction

The purpose of this document is to present the development journey of the Proximity Marketing Application (PMA). Progressive in nature, the PMA is a web application that allows in-store customers to access product reviews, a product price comparison and digital discount vouchers. This project is a requirement of the WIT Software Systems Development Final Year programme.

It is hoped this report will clearly document how the PMA was developed by outlining the development process and methodology, the technologies used and the challenges faced and overcome. Beginning with a project overview, the development journey will be explained under the following headings:

- **Project:** Overview, Scope, Target audience
- **Technologies:** Hardware, Software, Methodology
- **System Architecture:** Overview, User states, Flowchart, Architectural Design
- **Environment Setup:** React, Firebase, Deployment, Authentication
- **Component Development:** BLE, Reviews, Compare, Vouchers
- **Application Data:** Data Tables, Entity Relationship, Firestore
- **UX Development:** Product Description, Reviews, Vouchers & Sign-in Screens
- **Conclusion:** Challenges, Solutions, Future Development
- **References**
- **Appendices**

# 2. Project

This section gives an overview of the PMA and outlines the scope of the project, along with the intended target audience.

## 2.1 Overview

Using Bluetooth Low Energy (BLE) beacon technology, it was the intention of this project to design and develop a progressive web application that instantly provides shoppers with relevant information about products they're physically looking at - straight to their mobile device. This app allows users to review and compare products and take advantage of special offers through digital discount vouchers.

Combining the in-store experience, where you can pick up, touch and get the feel of a product, with the online experience where reviews and product insight are at your fingertips - the PMA brings the benefits of both worlds together.

82% of smartphone users say they consult their phones on purchases they're about to make in-store, making it vital for companies to identify and engage the consumer at the right time and in the right way.[1]

[1] https://www.unacast.com/post/proximity-marketing-what-how-why

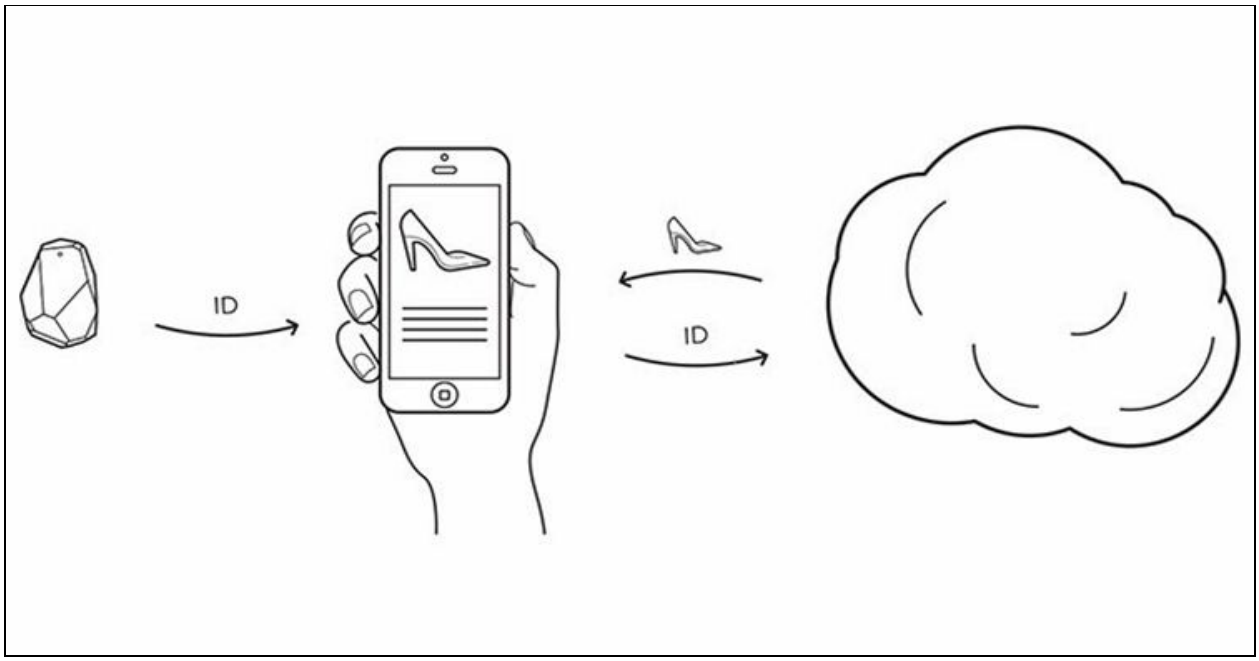


Fig 1. Beacon system overview[2]

Take this typical scenario: A customer walks into a store and heads directly to a pair of shoes. Standing in front of the shoes she receives a browser notification on her smartphone. She clicks the link, upon which the PMA provides her with product info, reviews, special offers and a price comparison with similar products.

Customers increasingly expect to be pushed the right content at the right time based on the context, compared to the tedious process of diving into the phone to pull content out manually.[3] The PMA intends to deliver information that is timely, relevant and personal.

[2] http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/
[3] https://www.unacast.com/post/proximity-marketing-what-how-why

## 2.2 Scope

The following table outlines the feature set for the PMA.

**Feature Set Table**

| REF | Name | Description |
|---|---|---|
| FS-001 | Product Review | View online reviews for the product |
| FS-002 | Price Compare | Offers a price comparison with similar products online |
| FS-003 | Vouchers | Send discounts and special offers to customers |

Fig 2. Feature Set Table

A discussion on how these features are implemented can be found in section 6.

## 2.3 Target Audience

The target audience for this application are retail customers that like to make an informed decision using their smartphone before purchasing an in-store product. Location-based marketing is fast changing the way customers interact with brands around them. A 2014 consumer study conducted by beacon technology platform Swirl found that over 70% of shoppers who received beacon-triggered content and offers on their smartphone said it increased their likelihood to make a purchase during a store visit. More than 60% of respondents said they'd do more shopping at brick-and-mortar stores that delivered mobile content and offers while they shopped, and 61% of people said they'd simply visit a store more often if they offered beacon marketing campaigns.[4] Figures like these would suggest customers are very receptive to the idea of brand owners pushing them content as long as it's timely and relevant. In other words, modern shoppers enjoy customized experiences and personalized recommendations and fit into the target audience for this application.

[4] https://www.shopify.com/retail/the-ultimate-guide-to-using-beacon-technology-for-retail-stores

# 3. Technologies

This section outlines the technology and methodology used to develop the Proximity Marketing App and a justification for their usage.

## 3.1 Hardware

**Bluetooth Beacons**

Fig 3. Kontakt beacon[5]

This project will utilize three Kontakt Bluetooth Low Energy (BLE) beacons. BLE beacons are essentially hardware transmitters that broadcast unique identifiers to nearby portable electronic devices. This technology enables smartphones, tablets and other devices to perform actions when in close proximity to a beacon.

Why choose Kontakt beacons?

BLE vendors are many and varied, however Kontakt support both iBeacon and Eddystone protocols, offer extended API functionality and an easy to use SDK for

[5] https://developer.kontakt.io

integrating functionality into existing apps. Kontakt also offer reasonable pricing for their beacons along with great customer service.

A more detailed description of Bluetooth beacons and how they will work in the context of the PMA will be given later in the Component Development section.

## 3.2 Software

**React.js**

React is an open-source JavaScript library used for handling the view layer of web and mobile apps. It allows for the creation of reusable UI components.

Why use React?

The main purpose of React is to be fast and scalable. As a student developer, it is my intention to broaden my horizons regarding front-end development frameworks. Having experience using Angular I wish to expand that knowledge base and use React for this project in order to add another 'bow to my string' so to speak.

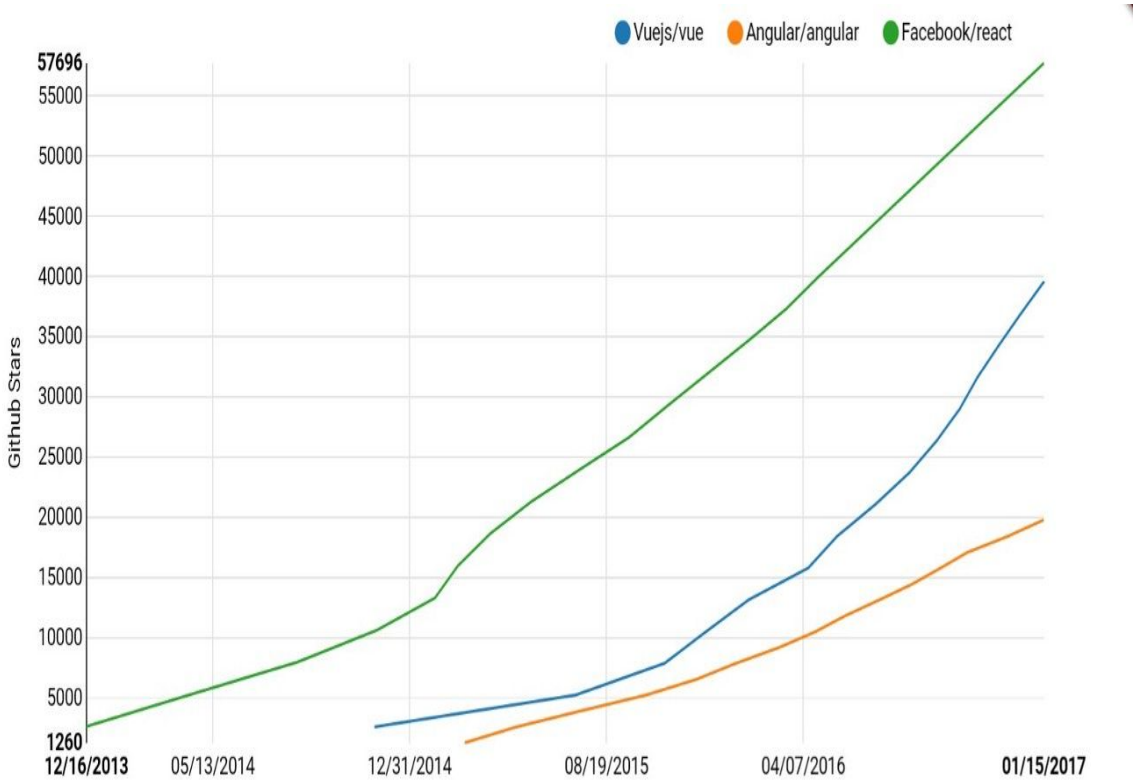


Fig. 4 Github stats: front-end frameworks

React certainly is a favourite with big business dealing with data heavy server-side rendering. Facebook, Instagram, Netflix and Yahoo all use React components. The figure above shows React's popularity on Github in relation to other prominent front-end frameworks.[6] Vue is another popular progressive framework for building user interfaces. It is similar in ways to React, both use virtual DOM's and both are solutions to the same problem. However, React fully embraces JavaScript as Vue does not. Using Vue templates creates scoping issues, whereas with React all the standard JavaScript scoping rules apply.[7]

Benefits of using React include:

- Components - UI is made up of reusable components.
- One-Way Data Flow - allows for better scalability and debugging.
- Virtual DOM - more efficient UI update.

[6] https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176
[7] https://medium.com/@CalinLeafshade/why-i-chose-react-over-vue-3dd9a230b507

- Ecosystem - great support community

Stack-overflow illustrates the rapid growth and decline of other web frameworks over the last few years in the graph below.

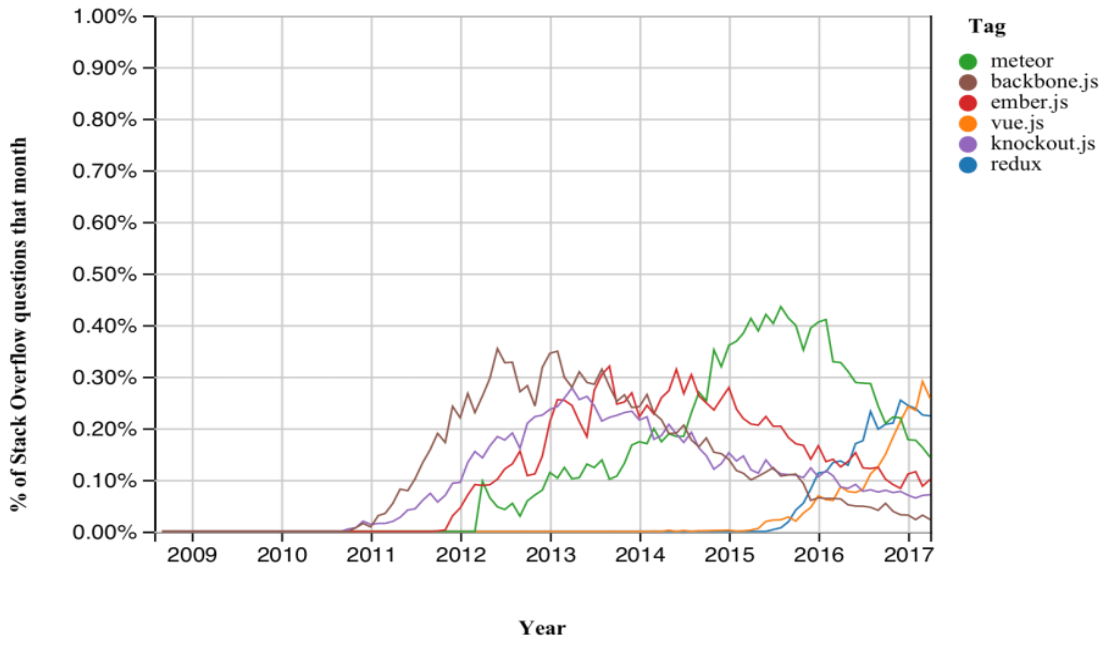


Fig. 5 Web UI Frameworks[8]

## Firebase

Firebase is a web application development platform supported by Google. It will form the backbone of the PMA as it will be used to provide server-side functionality, real-time data storage and deployment. A more in-depth analysis of Firebase will be covered later in the Environment Setup section.

Why use Firebase?

- Firestore database
- Built-in authentication process
- Large data storage capacity
- Notifications

[8] http://www.insights.stackoverflow.com/trends

- Free to use

Other platforms such as Amazon Web Services were considered to provide server-side support for the PMA. However, seen as several components of the app use Google technologies (Eddystone, NearBy) it was decided that Firebase, also a Google technology, would be the most suited and help minimise the risk of conflict and data communication problems.

### Google NearBy Notifications API

The NearBy API enables a web site or web app to be associated with a BLE beacon. Essentially, NearBy facilitates a notification to be pushed to a web browser on a mobile device. A URL opens in the web browser when the user clicks the notification.

Why use NearBy?

- No app install required
- Supports Eddystone
- Addition tags for analytics

### Webhose API

Webhose offers two API's that are critical to the feature set of this application.

1. Reviews API - returns online reviews. It allows the PMA to get comprehensive, up-to-the-minute access to millions of online reviews from thousands of sources. This API will be used to implement the product review component of the PMA feature set.
2. eCommerce Product Data API - get access to a database of millions of products from thousands of online retailers and e-commerce sites.[9] This API

[9] https://docs.webhose.io/docs/ecommerce-product-data-api

was used to implement the price comparison component of the PMA feature set.

Why use Webhose?

- All data returned from this API is in the JSON format, which is ideal seen as this app will be written entirely in JavaScript.
- Free to use RESTful API token.
- After investigating similar alternative technologies such as Import.io and Parsehub, it was clear Webhose offered a more extensive service with better functionality and more in depth documentation and support.

**Voucherify (voucherify.js)**

Voucherify is a cloud-based API that helps integrate loyalty and discount functionality into web applications. This platform gives total flexibility in the way coupon and loyalty systems are designed, implemented and maintained.

Why use Voucherify?

- The Voucherify RESTful API allows a voucher redemption system to be plugged into any web application, in this case the PMA.
- Extensive support documentation.
- API integration.

## 3.3 Methodology - Agile SCRUM

For this project an Agile SCRUM approach was applied towards development and design. Scrum is a subset of Agile and is generally geared towards development teams with no less than 5 or 6 members. As this is an individual project, it is the intention of the developer to extract the most relevant and appropriate components of SCRUM and apply them to the development process. After considering other methodologies such as the Waterfall model, it was decided that agile development methods are best suited to this project.

# 4. System Architecture

This section gives a high level overview of the PMA system and its interconnected components. Also discussed are the user states and the architectural design.

## 4.1 Systems Overview Diagram

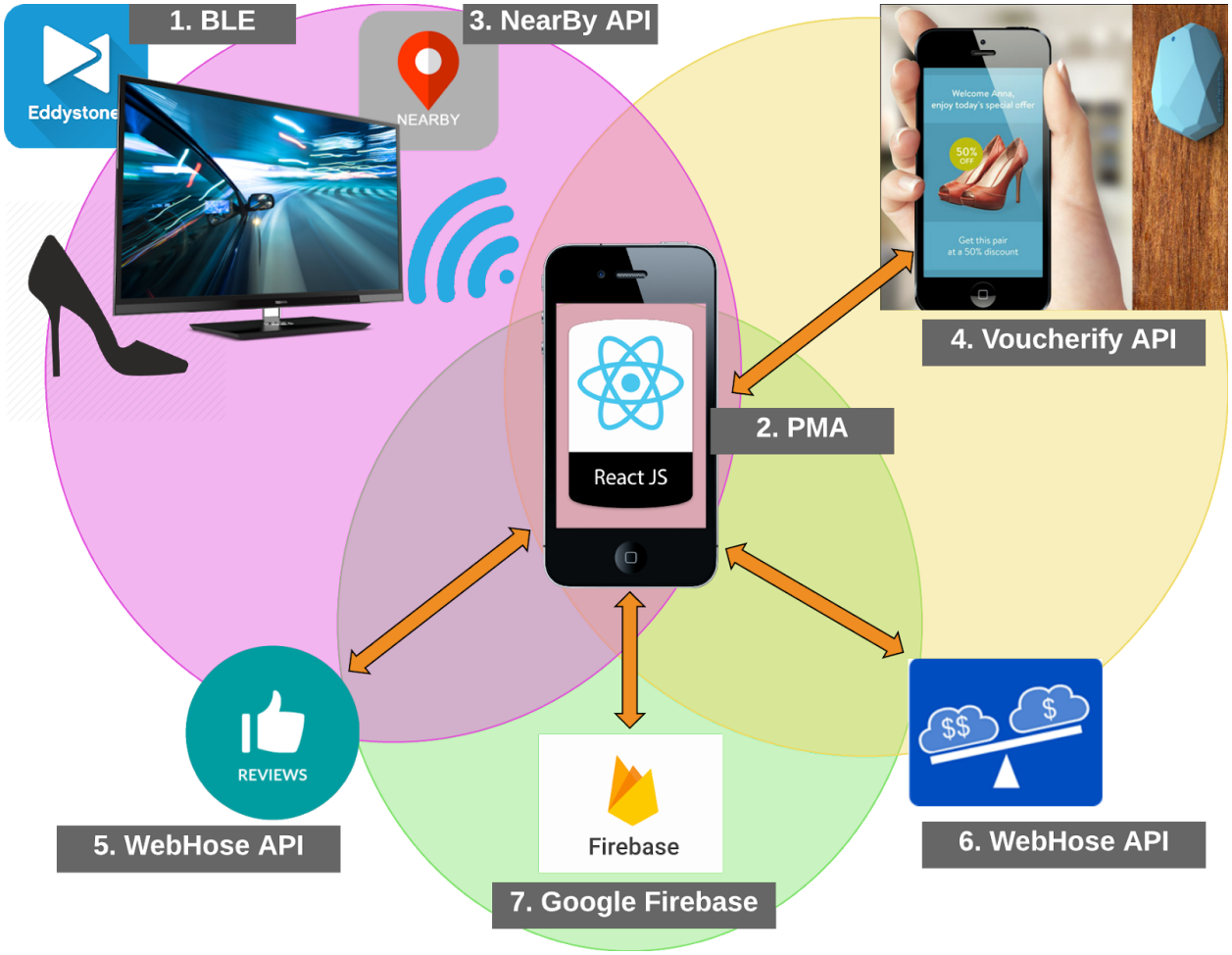


Fig. 6 Systems Overview Diagram

**(1)** BLE - Eddystone formatted Bluetooth low energy beacon sends a push notification to mobile device. User clicks on notification to be taken to the PMA.

**(2)** PMA - this web app was developed to be as progressive as possible, meaning:

- Works for all users regardless of browser choice
- Fully responsive - adapts to desktop, mobile, tablet
- App-shell model to provide app-style navigation and interactions.
- Secure - served via HTTPS to enhance security

**(3)** NearBy allows nearby Bluetooth devices to be detected.

**(4)** The Voucherify API allows users to redeem digital vouchers.

**(5) (6)** Webhose API gives access to online product reviews along with product price comparisons.

**(7)** The app is deployed and persistence maintained using Google's Firebase platform.

**PLEASE NOTE:** The integration and implementation of the above technologies (1-7) will be discussed at much greater length in the Component Development section.

## 4.2 User States

Two user states exist for the PMA

1. Registered
2. Unregistered

Registered

A registered user is granted access to the complete functionality of the PMA, which includes:

- Product Review
- Price Comparison
- Digital Vouchers

Unregistered

Unregistered users have limited access to PMA functionality, including:

- Product Review
- Price Comparison

Below a flow chart illustrates the steps a user takes using the PMA.

## 4.3 Flow Chart

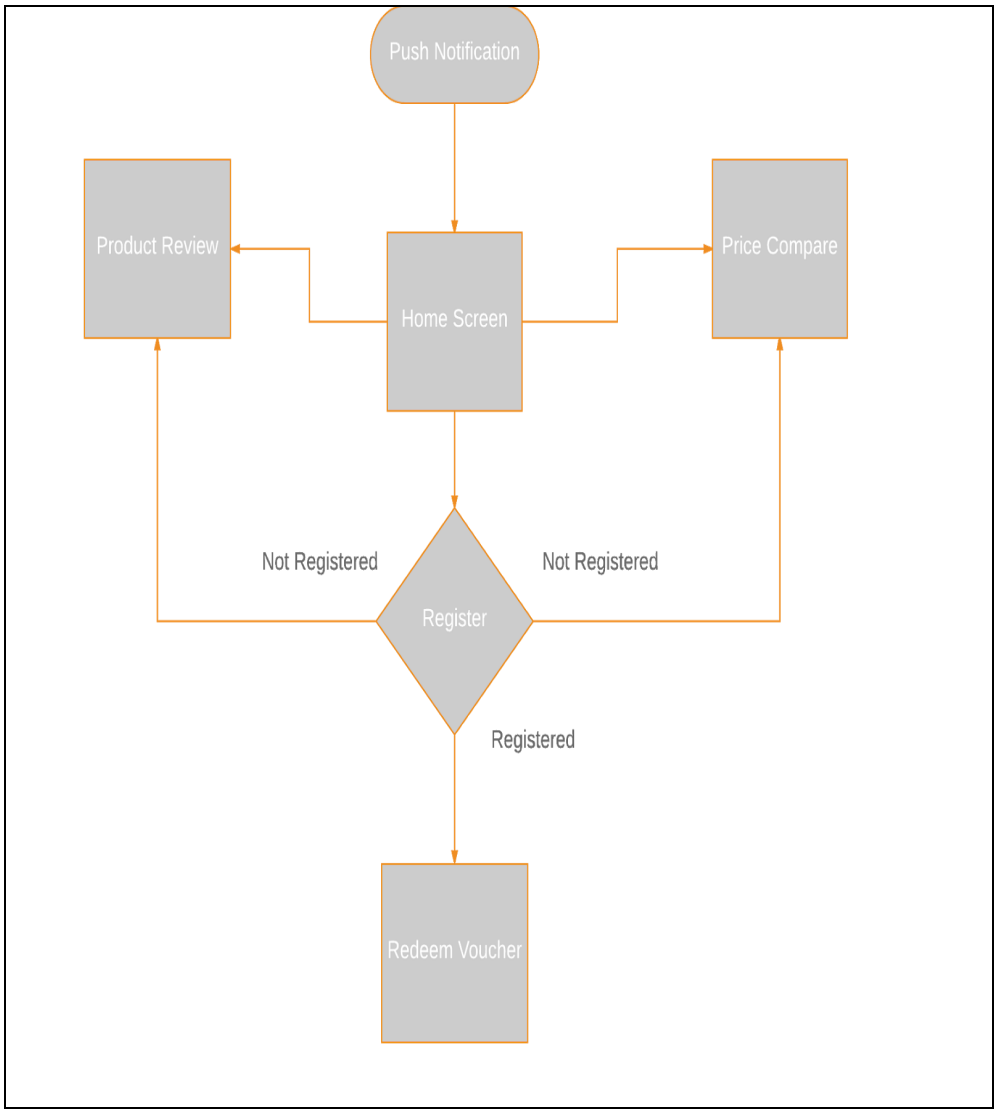


Fig. 7 User Flow Chart

## 4.4 Architectural Design

Flux

Flux is an architecture developed by Facebook as a solution to problems encountered using the traditional MVC design pattern. As the PMA uses React.js to render the front-end presentation layer, Flux is employed to complement React by using a unidirectional data flow design pattern.

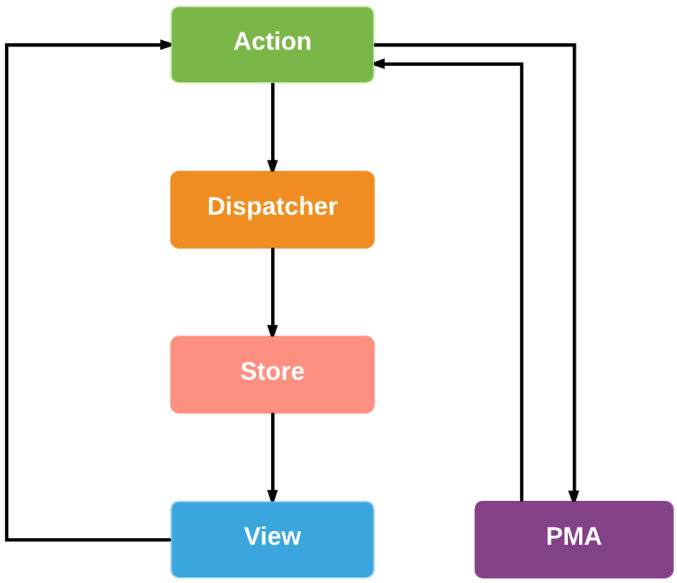


Fig. 8 Flux Design Pattern

- Actions - are methods that pass data to the Dispatcher
- Dispatcher - acts as a control centre receiving and broadcasting payloads to callbacks
- Stores - a collection of data storing the UI state to be used by the dispatcher
- Controller Views - UI components

# 5. Environment Setup

This section explains how the PMA was initialised and structured while looking closely at the integration of React and Firebase.

## 5.1 Initialising React

As the PMA is essentially a frontend React application driven primarily by JavaScript, it was necessary to have the following prerequisites installed for local development.

- NodeJS > v6

  https://nodejs.org/
- NPM - Node Package Manager (NPM comes packaged with NodeJS)

Firstly, it was necessary to create an application directory that will hold the proximity marketing app. In this instance the directory was simply named PMA.

After changing into the application directory - the following command was executed to create the React app.

```
$ npm install -g create-react-app
```

As we can see, npm creates the application and uses the -g flag to install the React app globally.

A boilerplate structure is created within the application directory as shown below:

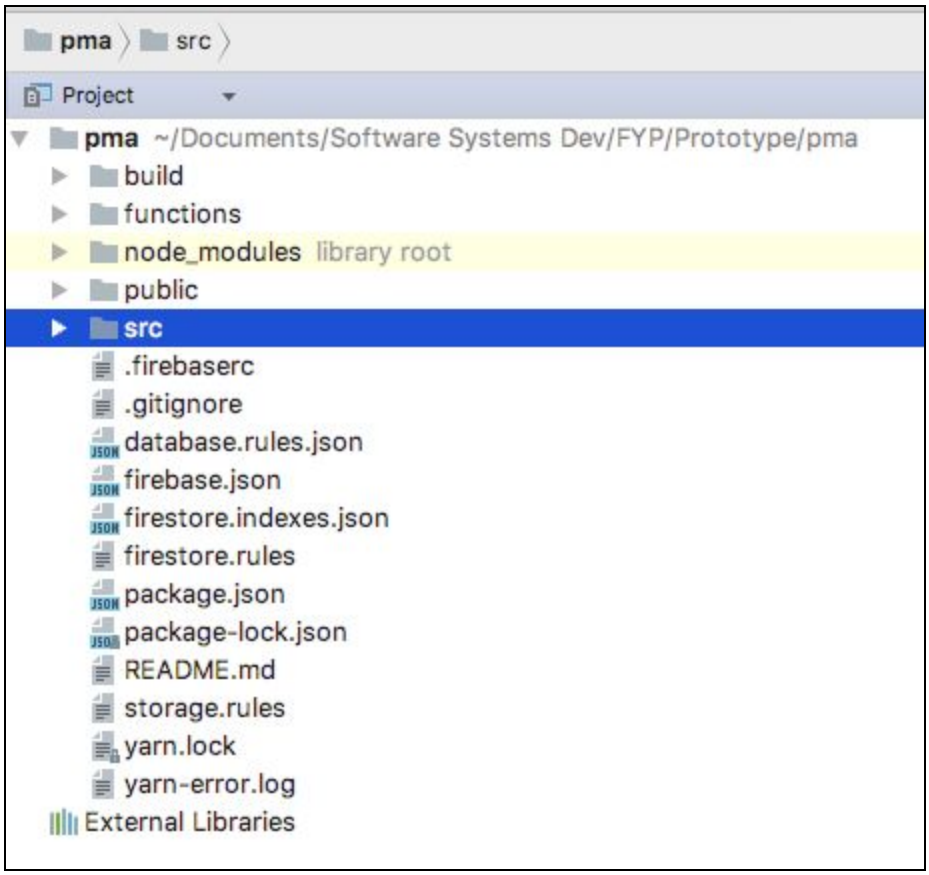


Fig. 9 React Directory Structure

- **build directory** - Application is built to this directory for production
- **functions directory** - Contains Firebase Cloud Functions. Explained later in Firebase section
- **node_modules directory -** Contains all the dependencies of the application
- **public directory -** Serves static files, i.e. No interaction with backend.
- **src directory** - Source directory that holds all the application code

In Fig. 9 can be seen a number of other files that mostly pertain to Firebase, which will be discussed later in this document.

## package.json

One of the most important files in the application directory is package.json. This file describes the application by name, version etc, but also lists all the application dependencies that are critical to functioning of the app and all the executable script commands.

```
{
  "name": "pma",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "firebase": "^4.8.1",
    "font-awesome": "^4.7.0",
    "material-ui": "^0.20.0",
    "react": "^16.2.0",
    "react-buttons": "0.0.11",
    "react-dom": "^16.2.0",
    "react-floating-button-menu": "^1.0.8",
    "react-google-button": "^0.5.1",
    "react-router-dom": "^4.2.2",
    "react-scripts": "1.0.17",
    "react-social-login-buttons": "^1.3.6",
    "react-star-rating-component": "^1.3.0",
    "semantic-ui-react": "^0.78.2",
    "voucherify": "^2.13.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

Fig. 10 package.json

Fig. 10 highlights all the application dependencies listed in the PMA package.json file. The name of the dependency is outlined along side the required version.

A description of each of the PMA dependencies is given below:

- **firebase v4.8.1** - Google's development platform. Explained is greater detail in the next section
- **font-awesome v4.7** - Font and character types
- **material-ui v0.20** - User interface framework
- **react v16.2** - Latest version of React JavaScript library
- **react-buttons v0.11** - Button component
- **react-dom v16.2** - React's virtual document object model
- **react-floating-button-menu v1.0.8** - Button component
- **react-google-button v0.5.1** - Google button component
- **react-router-dom v4.2.2** - Handle path routing
- **react-scripts v1.0.17** - Handle the scripts in package.json
- **react-social-login-buttons v1.3.6** - Button component
- **react-star-rating-component v1.3** - Five star rating component (Reviews)
- **semantic-ui-react v0.78.2** - User interface framework
- **voucherify v2.13.1** - Digital vouchers API component

**Scripts**

Beneath the dependencies in package.json are 4 react scripts:

- **start -** $npm run start → runs the application in local environment. Can be accessed in browser at localhost:3000
- **build -** $npm run build → builds the application to build folder for production
- **test -** $npm run test → runs application tests
- **eject** - $npm run eject → removes the boilerplate dependencies from the application

The primary function of React is to handle the frontend presentation of the application. The backend is primarily handled by Firebase which will be discussed next.

## 5.2 Adding Firebase

Firebase plays a key role for the PMA in several areas:

- **Cloud Firestore** - Allows the PMA to store user data on the Firestore platform.
- **Cloud Functions** - Cloud Functions automatically run backend code in response to event triggers and HTTPS requests. Although structured into the PMA project, Cloud Functions are not presently utilised, but hope to be in future development.
- **Hosting** - Firebase Hosting allows the PMA to be deployed live on Google servers. More details regarding hosting and deployment later in this section.
- **Authentication** - Using the industry standard OAuth 2.0, Firebase allows the PMA to authenticate its users through social providers such as Google, Facebook and Twitter.

Before adding Firebase to the PMA it is necessary to create a project within Google's Firebase console.

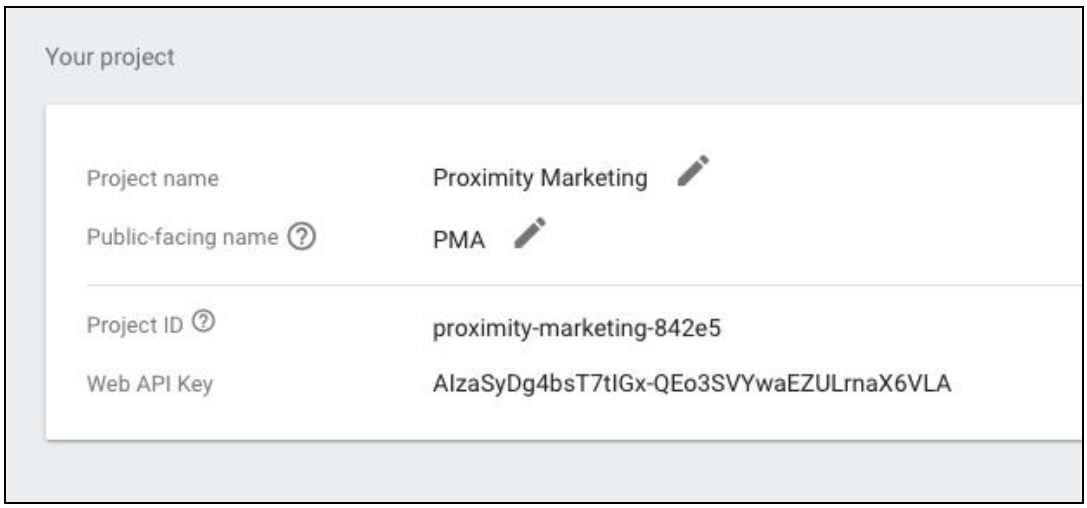


Fig. 11 PMA Project in Firebase Console

Once a project has been created as shown in Fig. 11, Firebase awards credentials that grants access to the platform's functionality. Such credentials can be seen in the Firebase configuration file below:

```
1   import * as firebase from 'firebase'
2
3   // Initialize Firebase
4   var config = {
5       apiKey: "AIzaSyDg4bsT7tIGx-QEo3SVYwaEZULrnaX6VLA",
6       authDomain: "proximity-marketing-842e5.firebaseapp.com",
7       databaseURL: "https://proximity-marketing-842e5.firebaseio.com",
8       projectId: "proximity-marketing-842e5",
9       storageBucket: "",
10      messagingSenderId: "549732289667"
11  };
12
13  firebase.initializeApp(config);
14
15  export default firebase;
```

Fig. 12 Firebase Config File

The next step is to install the Firebase Command Line Interface (CLI) which provides a variety of tools for managing, viewing, and deploying to Firebase projects.

```
npm install -g firebase-tools
```

The CLI is available at:

https://github.com/firebase/firebase-tools

Using the CLI - Firebase can be initialised within the PMA by the following command:

```
$ firebase init
```

Now the PMA is initialised as a React frontend application with Firebase functionality integrated and ready to use. For a discussion on firebase.json and firestore.rules please see Appendix A.

Firebase Hosting - Deployment

As mentioned earlier Firebase allows the PMA to be deployed by executing the following command within the project directory:

```
$ firebase deploy
```

The PMA application is then deployed to:


Firebase Hosting is SSL-only, meaning that content will only be served over HTTPS.[10]

As the above URL is very long and not aesthetically pleasant, a custom domain name was registered and connected to the application. Instead, the URL below points to the PMA application which is shorter and more relevant.


After confirming the domain name ownership it was a trivial task of connecting the URL to the application in the Firebase console.


Fig. 13 Connect Domain Name

[10] https://firebase.google.com/docs/hosting/deploying

Firebase Authentication

A Social Identity study conducted by Blue Research in 2012, gives great insight into consumer patterns regarding web registration and authentication.

- 86% of consumers are bothered by registering at a website[11]
- 77% think websites should offer social login, instead of requiring the creation of a new account[12]
- 54% may leave the site or not return if they had to fill out a form to login[13]

With this in mind it was decided the PMA would provide social login for the 3 main social media providers and omit the traditional email/password login.

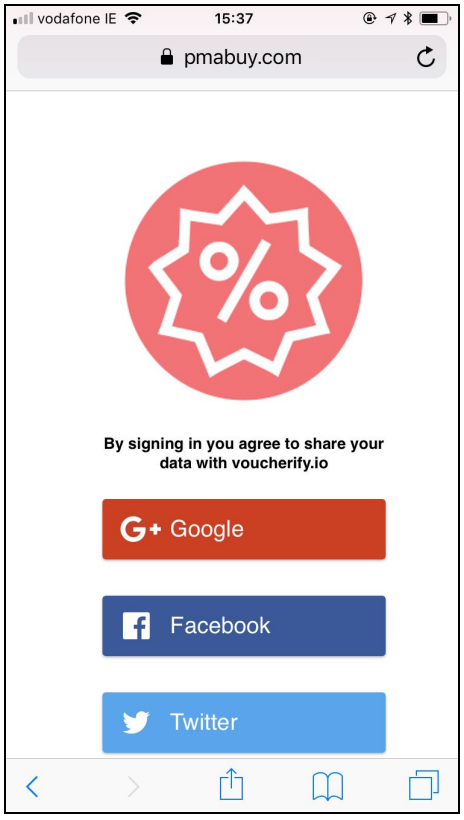


Fig. 14 Social Login Buttons

[11] http://www.prweb.com/releases/2012/1/prweb9086226.htm
[12] Ibid.
[13] Ibid.

Note the disclaimer informing users that by signing into the application, their user details will be shared with voucherify.io.

As shown in the user flow chart in Fig. 7, once the user is logged in, they then have access to product vouchers.

Firebase provides backend services to authenticate potential customers. Using the Google sign-in process, the next section looks at how authentication works. To explore how the Facebook and Twitter authentication process works please refer to Appendix B.

How Firebase Authentication Works

Firstly the user taps on the Google sign in button shown above in Fig. 14 which automatically redirects them to the Google sign-in screen - if they are not already logged in.

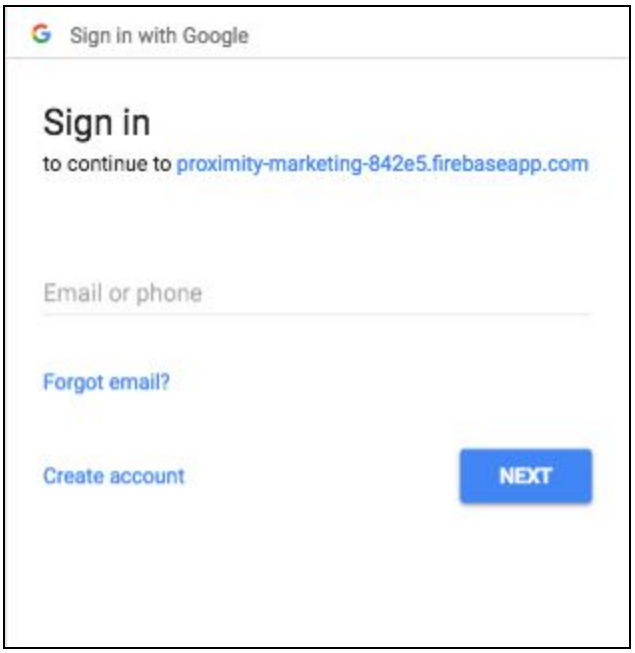


Fig. 15 Google Sign-in Screen

```
4
5   const provider = new firebase.auth.GoogleAuthProvider();
6
7
```

Fig. 16 Define New instance of GoogleAuthProvider

```
129
130         //Google social signin
131         firebase.auth().signInWithRedirect(provider);
132
```

Fig. 17 Redirect to Google Sign-in

Fig. 16 defines a new Firebase instance of GoogleAuthProvider, while Fig. 17 implements the redirect to the Google sign-in screen

The users credentials are passed to the Firebase Authentication SDK when they login to Google. The Firebase backend services then verify those credentials and if valid, the user is pushed back to the PMA voucher screen where they can continue to get their discount voucher.

After a successful sign-in, the PMA can access the user's basic profile information such as:

- Display Name
- Photo
- Email
- User ID

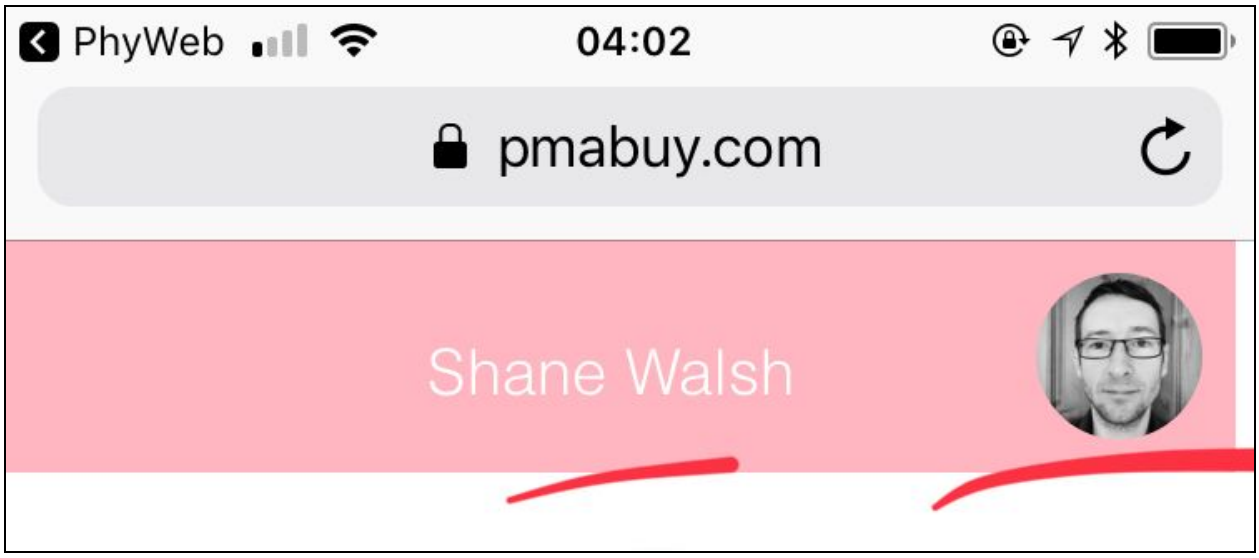


Fig. 18 Redirect to Google Sign-in

As shown above in Fig. 18, once logged in, the PMA has access to the user's name and photo.

Firebase Pricing

The PMA was developed using Google's free Spark Plan as illustrated below.

| Products | Spark Plan<br>Generous limits for hobbyists<br>Free | Flame Plan<br>Fixed pricing for growing apps<br>$25/month | Blaze Plan<br>Calculate pricing for apps at scale<br>Pay as you go<br>✓ Free usage from Spark plan included* |
|---|---|---|---|
| **Free Products**<br>Authentication (except Phone Auth), Analytics, Predictions, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging (FCM), Performance Monitoring, and Crashlytics. | ✓ Included | ✓ Included Free | ✓ Included Free |

Fig. 19 Firebase Pricing

# 6. Component Development

This section gives a detailed account of how all the technologies illustrated in the systems architecture work together to deliver a high value feature set. The component development journey will be discussed in the following order:

1. Beacon Setup
2. Product Reviews Component
3. Price Compare Component
4. Digital Vouchers Component

The diagram below gives an overview of the applications components.

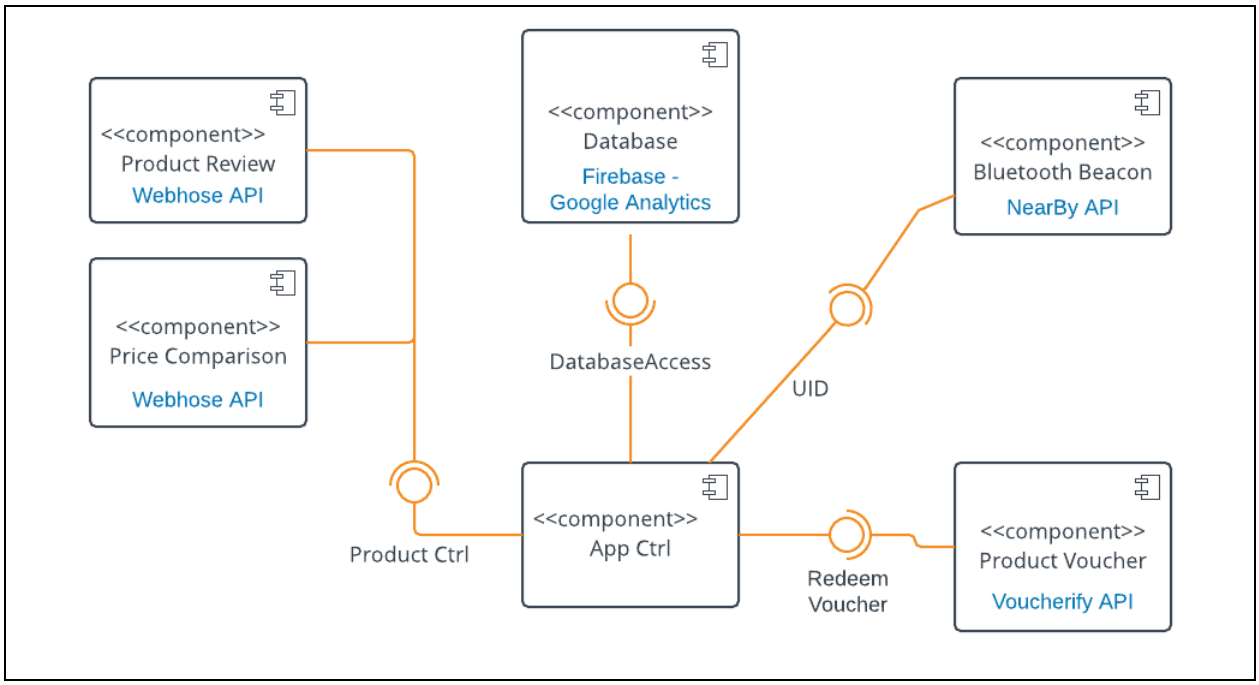


Fig. 20 Component Overview Diagram

## 6.1 Beacon Setup

How Bluetooth beacons work with the PMA?

Early in the development journey, it was paramount to establish a connection between the beacon hardware and the PMA software.

Each beacon is assigned to a single product. That beacon broadcasts or pushes a browser notification to all nearby mobile devices. Contained within that notification is a URL specific to that particular product. From this URL the user can access the application. The Bluetooth beacon uses the Eddystone communication protocol to transmit the URL. A more detailed explanation of this process is discussed below.

What is Eddystone?

Eddystone is communication protocol developed by Google that helps transmit information between two Bluetooth enabled devices. The Eddystone protocol can transmit 3 different frame-types:

- Eddystone-UID (Unique Identification)
- Eddystone-URL
- Eddystone-TML (Telemetry: data from sensors)

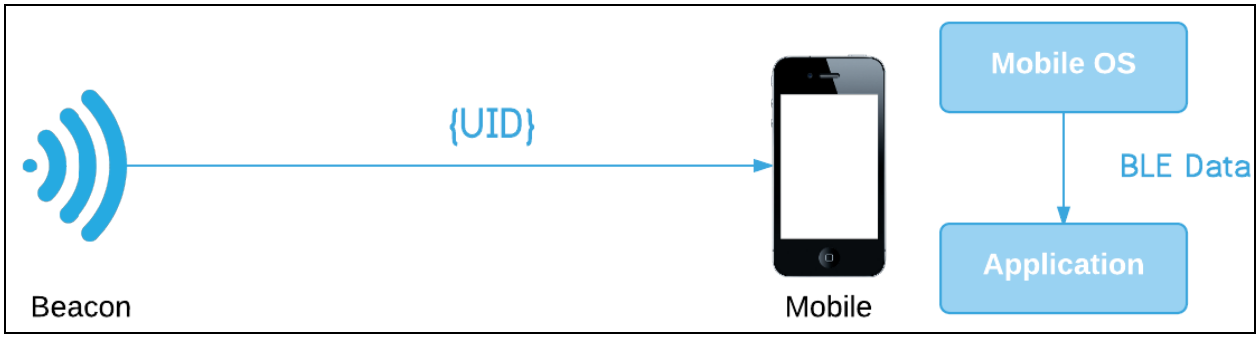


Fig. 21 Beacon - Mobile Communication

Important for this project is the Eddystone-URL which allows for a specific URL to be pushed to a user's mobile device. In this case our PMA URL, which can be seen below displayed on the user's notifications window:


Fig. 22 BLE Push Notification to PMA

Within that notification is a URL pointing to the PMA as seen in Fig. 22 above. The user clicks the notification to activate the web application which then displays the product description, product reviews and price comparison, with an option to sign-in to redeem digital vouchers.

Beacon Configuration

In order to implement the push notification, each beacon must be configured through the manufacturer's (Kontakt) web portal as shown in the following illustrations:

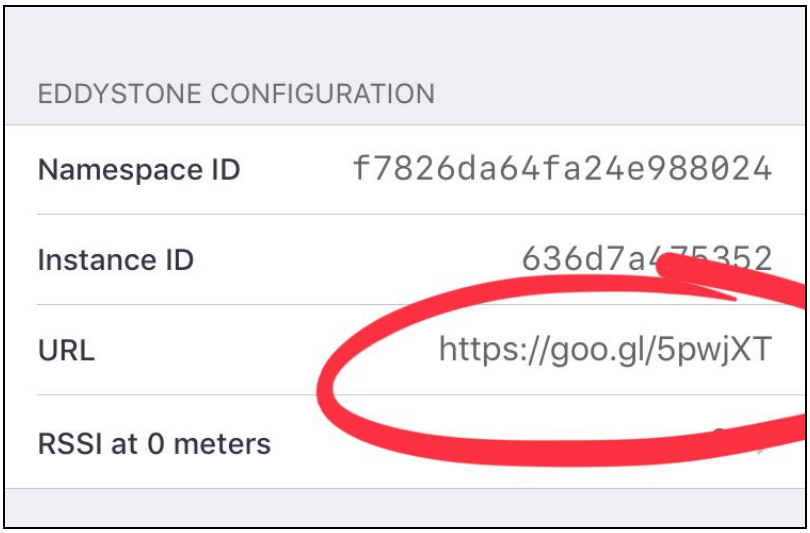


Fig. 23 Configure URL

Here we are setting the URL that the Bluetooth beacon will broadcast as a notification. Fig. 23 shows the URL that corresponds to the PMA. However, you may notice the URL marked in red above does not correspond to the URL in Fig. 22. This is because the URL above was shortened using Google's goo.gl utility, which was necessary because Kontakt does not allow URLs in excess of 20 characters. See Fig. 24 below.


Fig. 24 URL Redirect

The Kontakt web portal shown below gives an overview of all nearby beacon devices. Tapping on the icon takes you too configuration and set-up.

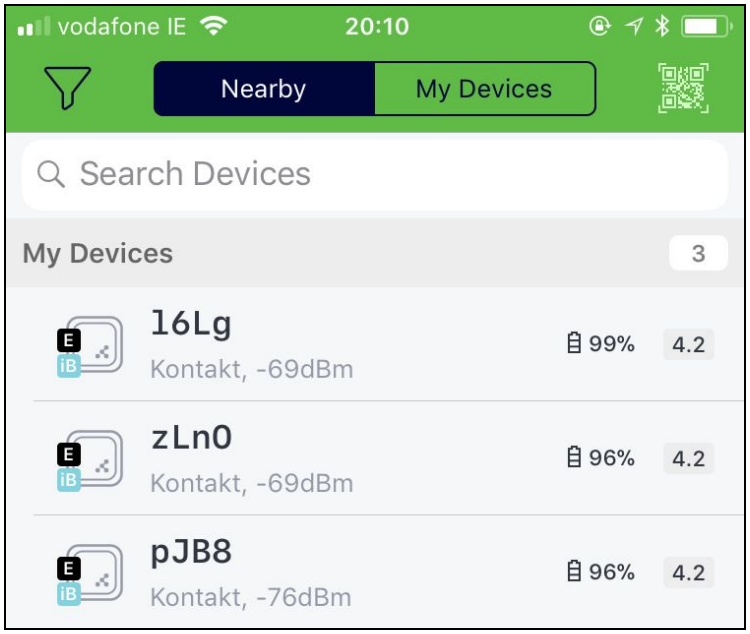


Fig. 25 Beacon Overview

Along with assigning the beacon URL, it is necessary to set the appropriate communication protocol as shown below.

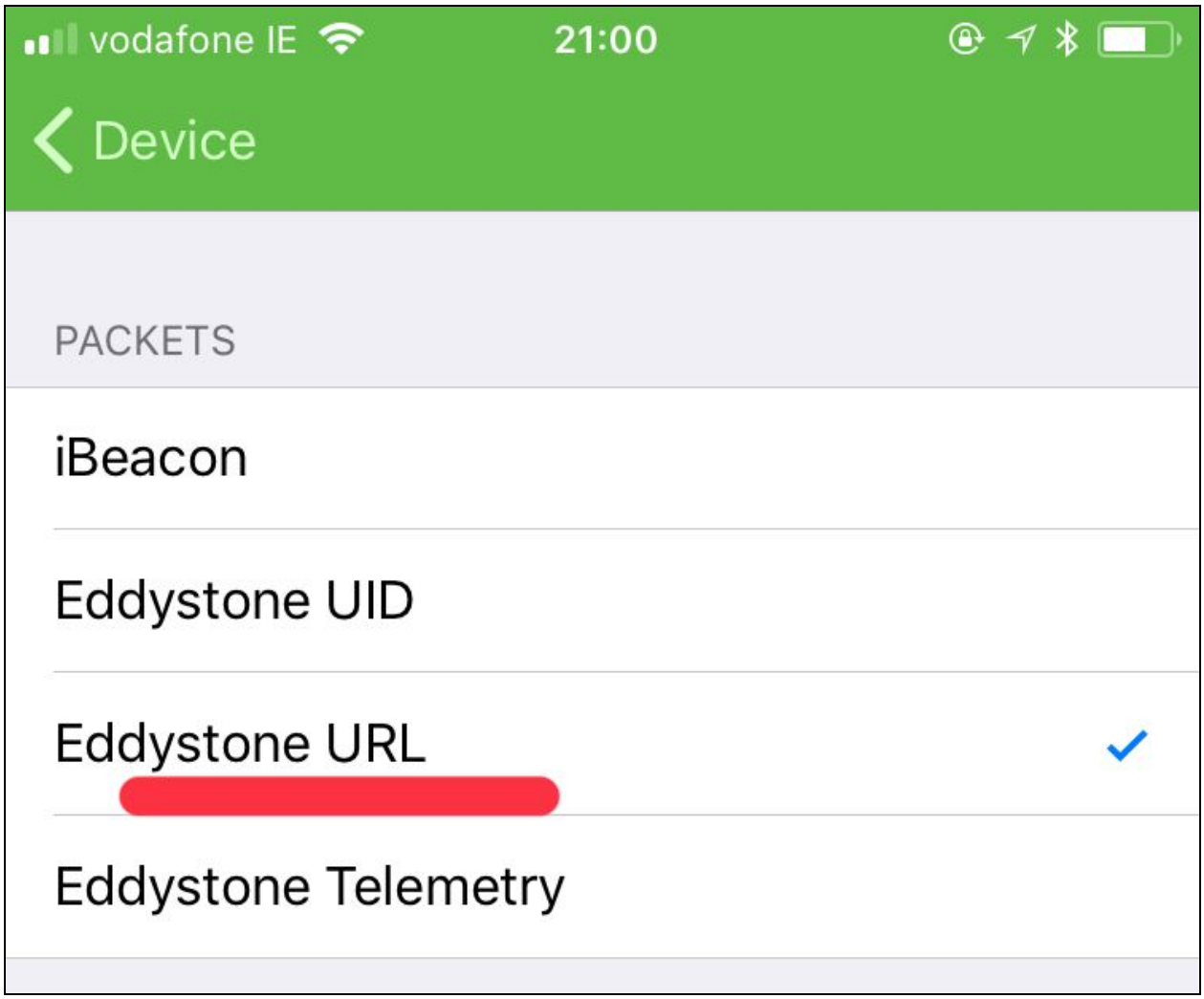


Fig. 26 Set Communication Protocol

Kontakt offer various different frame types or packets for their Bluetooth beacons. However, for the purposes of the PMA it is necessary to set our communication protocol to Eddystone and the packet to URL. This setting is of vital importance with regards to communications between the beacon hardware and the web-app software as it allows the push notification emitted by the beacon to point our users to the PMA web address.

Once the user taps the PMA URL in their notifications screen, they are brought directly to the PMA product screen as shown below.

PMA Product Screen

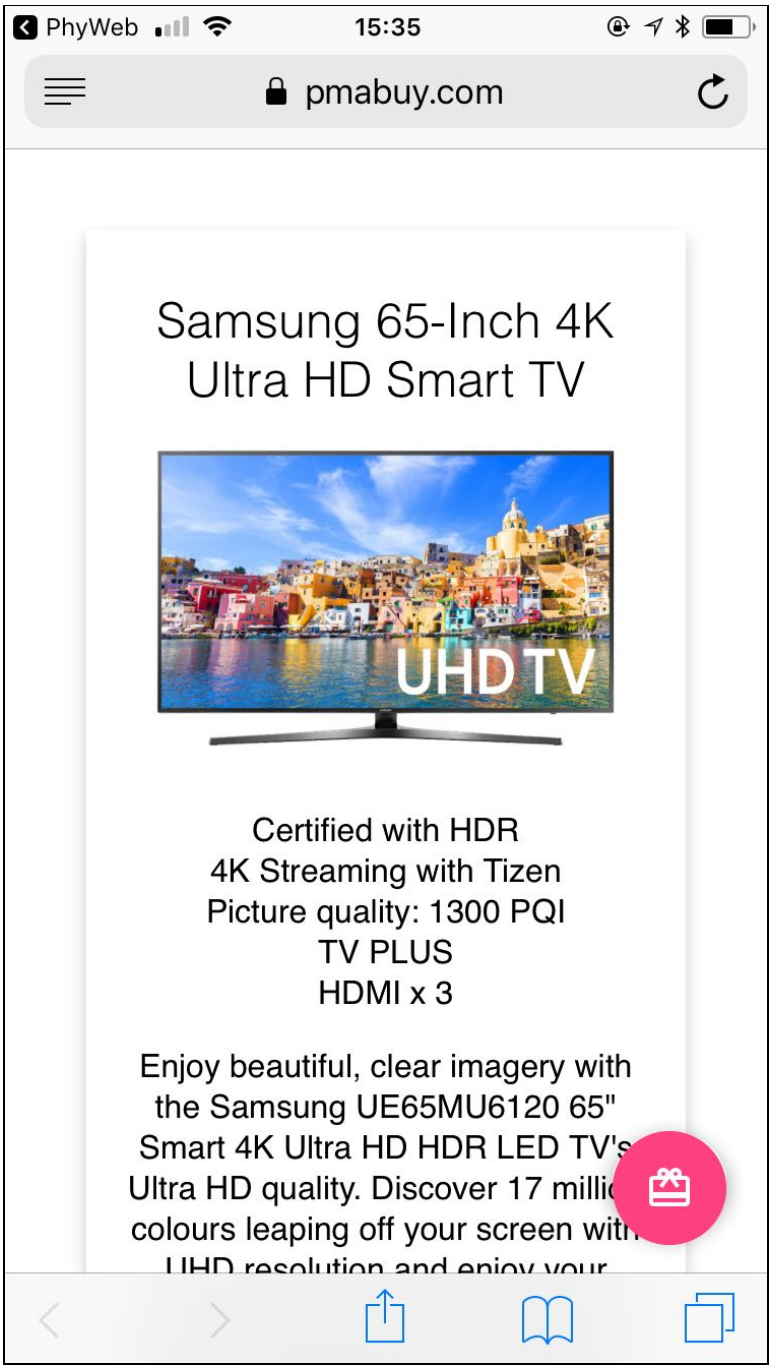


Fig. 27 PMA Product Screen

As can be seen in Fig. 27 above the user is presented with product information corresponding to the product they are focused on in-store.

The product screen is broken down into 3 sections:

- **Product Title**

  The title outlines the name and model of the product. It is the first thing the user sees. On occasion products in-store may not have the name or model listed, so it is reassuring for the user to have the exact name or model of the product on screen.

- **Product Image**

  Directly under the title the image of the product is displayed. Again, it is reassuring for the user to see an image of the product that is in focus.

- **Product Description**

  This section outlines the specifications and description of the product and offer the user a more in depth look at the product details.

## 6.2 FS-001 Product Reviews

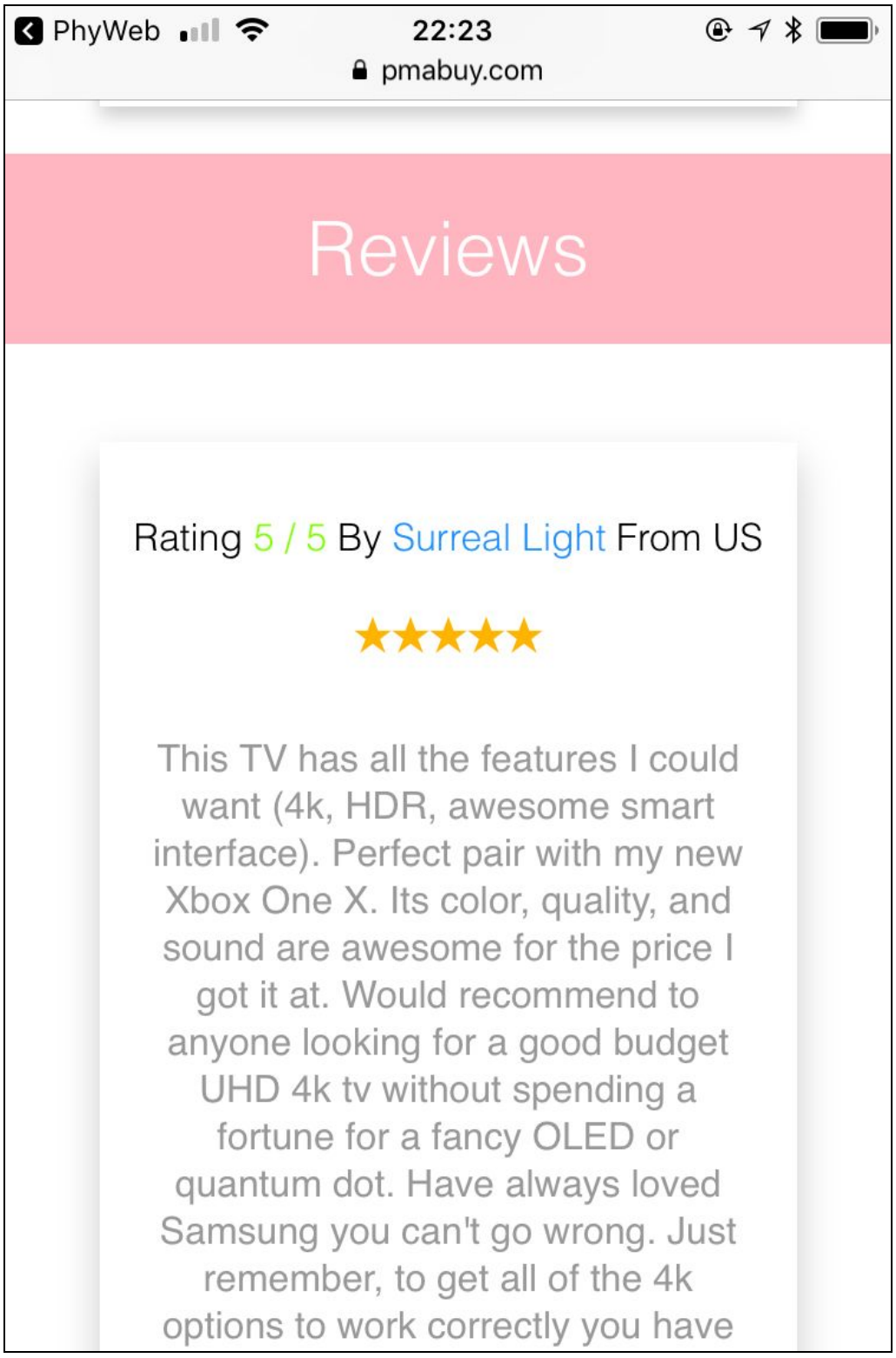


Fig. 28 Product Review

What is the Product Review Component?

Essentially the reviews component displays product reviews to help the user make a more informed decision on their potential purchase. The following figures suggest that positive reviews are a compelling way of convincing uncertain consumers to commit to a purchase.

- 70% of people consult reviews/ratings before purchasing[14]
- 71% agree that consumer reviews make them more comfortable that they are buying the right product/service[15]
- 63% of people are more likely to purchase a product from a site if it has product ratings and reviews[16]

Review are an effective way to create consumer trust in a product and with increased trust there is an increased likelihood of committing to a purchase. Similarly, the 5 gold star rating system is another way of creating a sense of consumer trust. As shown above in Fig. 28  the PMA displays a gold star rating which is dependent on the score given by the product reviewer. The reviewer's name and location are also displayed beside the rating to further enhance a sense of connection and trust.

Creating the Product Reviews Endpoint Query

As already mentioned in Section 3, the WebHose Reviews API was used to the implement the reviews component. It was necessary to register with WebHose.io in order to obtain an active API key as shown below:

[14] https://www.socialmediatoday.com/content/why-online-reviews-matter
[15] Ibid.
[16] Ibid.

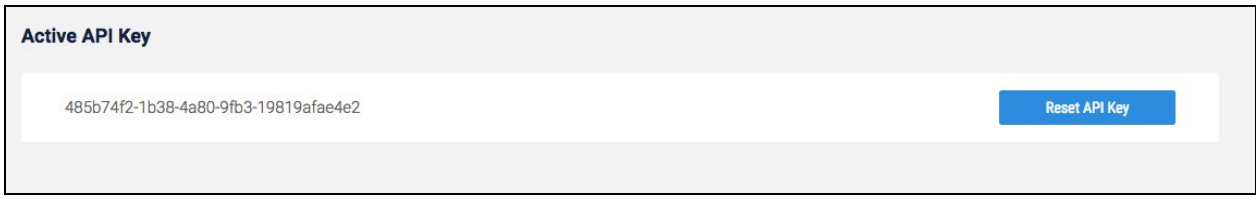


Fig. 29 WebHose Active API Key

In order to use the WebHose API it was necessary to call a WebHose endpoint using the active API key shown above in Fig. 29. Using a GET request, the endpoint returns structured web data, in this case product reviews, in JSON format that can be filtered to suit the needs of the PMA. For instance, the reviews need to be:

- Relevant
- Current
- Written in English

WebHose provides an online Reviews Query Builder to help create and test endpoint requests for product reviews. It is possible to tailor requests to suit specific needs. For instance, the following illustration shows a product reviews request for a Samsung 65" TV that are only written in the English language. In order to keep the reviews current and relevant a 30 day filter was imposed and can be seen in Fig. 30 below: "Crawled Since: 30 Days Ago"

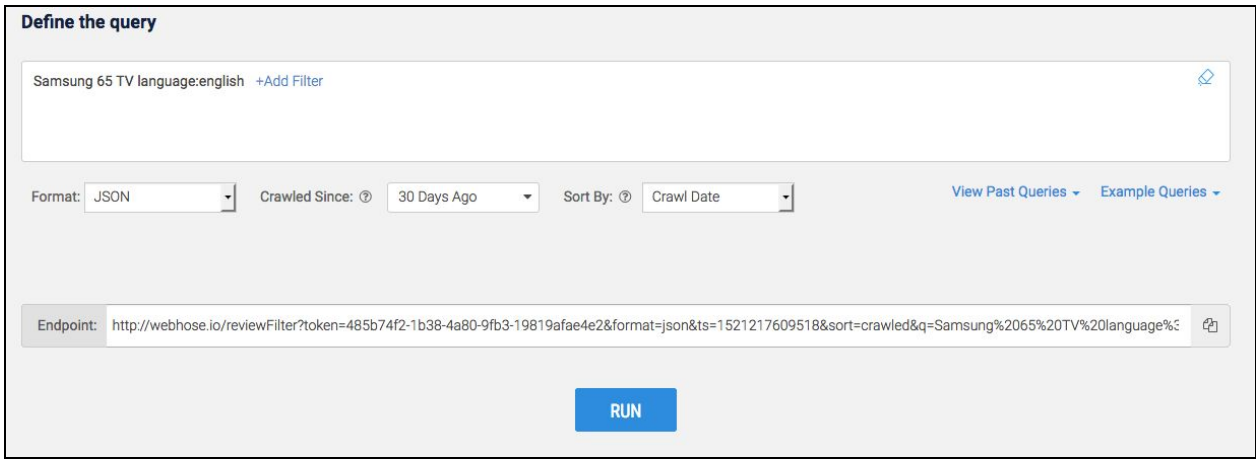


Fig. 30 WebHose Endpoint Query Builder

The endpoint query shown above in Fig. 30 returns JSON data in which the product review is contained. The next illustration presents the returned JSON and the product review side by side.

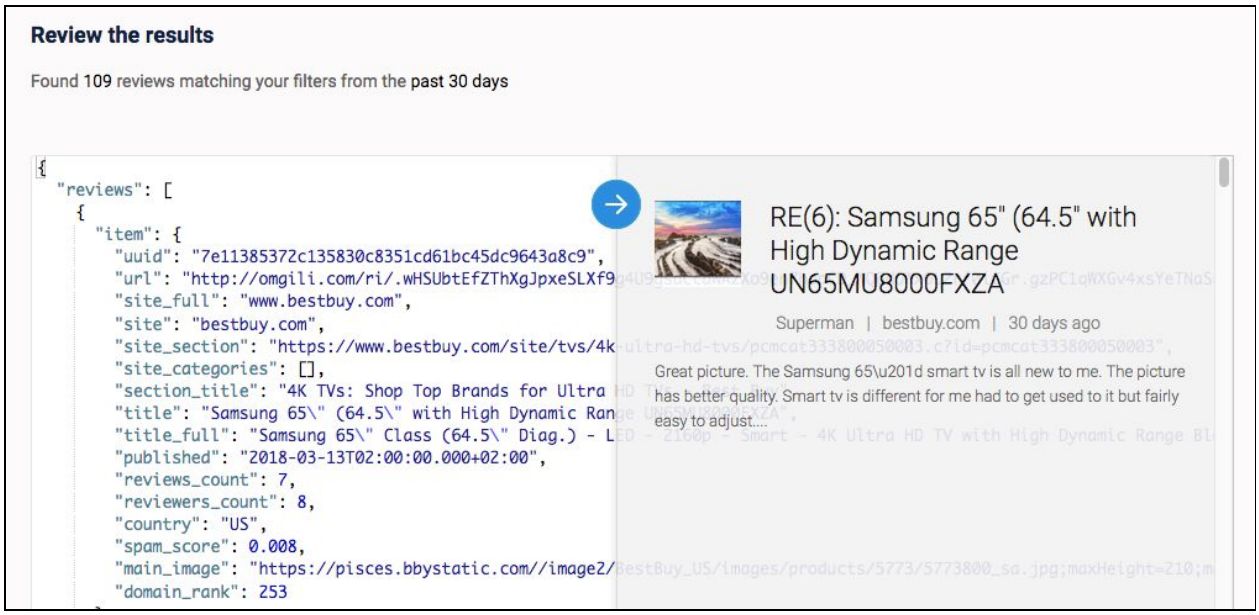


Fig. 31 Endpoint Query Returned JSON (WebHose Query Builder)

Each request may return up to hundreds or even thousands of matching responses, so for the purposes of the PMA the response limit is set to seven. The reason for limiting the responses are three fold:

1. Not to overwhelm the device with data
2. Not to overwhelm the user with reviews
3. User could get a good overall picture of a product from seven reviews

A sample JSON product review response is listed below. As the product reviews used by the PMA are dynamic in nature, only a sample can be given. Even though

the data is fluid and ever changing - the structure of the  JSON object remains the same.


```
{
    "reviews": [
        {
            "item": {
                "uuid": "690974929c9ae7d83f4249c439bf9c3bc83a0a30",
                "url": "http://omgili.com/ri
                    /.wHSUbtEfZThXgJpxeSLXf9g4U9gsaCcawAzXo9en7g6lWUF42SlZNFu5kOiHaYv_MDV0ryV04M_lTCkaU_vhLknk
                    .wrGm7rOGoGCNLllMu7u9yGD93jYxBhqXYsuVmnRSZr2hvJBA56B1Fm6Qt4ToFl7OGiovl3VdbCmmx2ScQZIvpn5epkSg--",
                "site_full": "www.bestbuy.com",
                "site": "bestbuy.com",
                "site_section": "https://www.bestbuy.com/site/tvs/40-inch-tvs/pcmcat1514909744071.c?id=pcmcat1514909744071"
                    ,
                "site_categories": [],
                "section_title": "40-Inch TVs: 40-Inch Flat-Screen Televisions - Best Buy",
                "title": "Samsung 40\" (39.5\" 2160p - Smart - 4K Ultra HD TV with High Dynamic Range Black UN40MU7000FXZA"
                    ,
                "title_full": "Samsung 40\" Class (39.5\" Diag.) - LED - 2160p - Smart - 4K Ultra HD TV with High Dynamic
                    Range Black UN40MU7000FXZA - Best Buy",
                "published": "2018-03-09T02:00:00.000+02:00",
                "reviews_count": 7,
                "reviewers_count": 8,
                "country": "US",
                "spam_score": 0.043,
                "main_image": "https://pisces.bbystatic.com//image2/BestBuy_US/images/products/5789/5789803_sa.jpg
                    ;maxHeight=210;maxWidth=210",
                "domain_rank": 253
            },

            "uuid": "267d13662a243b1cb1d2eb488880fe1ac57023db",
            "url": "http://omgili.com/ri
                /.wHSUbtEfZThXgJpxeSLXf9g4U9gsaCcawAzXo9en7g6lWUF42SlZNFu5kOiHaYv_MDV0ryV04M_lTCkaU_vhLknk
                .wrGm7rOGoGCNLllMu7u9yGD93jYxBhqXYsuVmnRSZr2hvJBA56B1Fm6Qt4ToFl7OGiovl3VdbCmmx2ScQ4
                .gk4M_H7vsaGGzN6vaywnEAT6XW8yEU5vYnpnLo1vPLijpLggtWf",
            "ord_in_thread": 4,
            "author": "CJ1272",
            "published": "2018-03-16T02:00:00.000+02:00",
            "title": "",
            "text": "Samsung 4K. This TV has many features and the smart remote is also very helpful. The remote has voice
                control very convenient. 3 HDMI ports and built in smart applications. TV guide for OTA channels is superb.
                If you have Samsung smart phone this TV integrates.",
            "highlightText": "",
            "highlightTitle": "",
            "language": "english",
            "external_links": [],
            "rating": 5,
            "crawled": "2018-03-17T02:35:28.015+02:00"
        },
```


Fig. 32 Product Review JSON Response

As can be seen above the product review JSON response returns a wealth of data regarding the review. However, the PMA is only concerned with several key elements within the JSON object as follows:

**Please Note**: The lines are numbered 1 - 35 in Fig. 32

- Line 25 - **author**

  Name of the reviewer

- Line 17 - **country**

  Country from which the review was published

- Line 28 - **text**

  The actual text body of the product review

- Line 33 - **rating**

  1 - 5 rating of the product. Used in correspondence with gold star rating system

The elements listed above are extracted from the JSON object by the PMA and presented to the user as shown in Fig. 28. The following section describes how this process was implemented.

## Product Review Implementation

After creating the GET product review request and checking the returned JSON response is correct and accurate, the endpoint URL is assigned to a constant within the Reviews component of the PMA.

```
const url = 'https://webhose.io/reviewFilter?token=485b74f2-1b38-4a80-9fb3-19819afae4e2&format=json
```

Fig. 33 Product Review Endpoint URL

A **fetch()** method uses the above URL as its argument and makes the asynchronous WebHose Reviews API call from the **componentDidMount()** lifecycle method as shown below:

```
componentDidMount() {

    fetch(url).then(reviews => {

            return reviews.json();
        }
    ).then(data => {

        let electronics = data.reviews.map((inst) => {

            return (
                <div className="card" key={inst.reviews}>
                    <h4 className="reviewTitle">Rating <span className="rating"> {inst.rating} / 5 </span>

                    <div>
                        <StarRatingComponent name="starRating" starCount={5} value={inst.rating} />
                    </div>

                    <div className="container">
                        <p className="reviewText">{inst.text}</p>
                        <hr/>
                    </div>
                </div>
            )

        })
        this.setState({electronics: electronics});
        console.log("This is Review State: ", this.state.electronics);
    })
}

render() {
```

Fig. 34 Query Product Review Endpoint using fetch()

The returning promise contains the response JSON object. When the data is fetched successfully, it is stored in local state within React's **constructor()** method:

```
10
11      constructor() {
12          super();
13          this.state = {
14
15              electronics: [],
16          }
17      }
18
```

Fig. 35 Define Local State in constructor()

The returned JSON is stored as an array of objects, in this case **electronics: []**

In order to extract the relevant elements from the returned JSON object, it is necessary to use the **map** function:

```
26
27          let electronics = data.reviews.map((inst) => {
28
```

Fig. 36 Mapping Elements of JSON Object

As mentioned earlier, the elements of particular value to the PMA are:

- **author**
- **country**
- **text**
- **rating**

The **electronics: []** array is then stored in local state:

```
45              this.setState({electronics: electronics});
46              console.log("This is Review State: ", this.state.electronics);
47          })
```

Fig. 37 Set electronics to Local State

After the **render()** method is triggered, instances of the review elements are rendered to screen.

```
49
50      render() {
51
52          return (
53
54              <div className="container">
55                  <div className="container">
56
57                      {this.state.electronics}
58
59                  </div>
60              </div>
61
62          )
63      }
64  }
```

Fig. 38 Render electronics

```
<span className="author">{inst.author} </span>
```

```
<div className="container">
    <p className="reviewText">{inst.text}</p>
    <hr/>
</div>
</div>
```

```
From {inst.item.country}</h4>
```

```
s}>
<span className="rating"> {inst.rating} / 5 </span>
```

Fig. 39 author|country|text|rating elements

All the above elements are displayed in Fig. 28 to give an idea of what they look like rendered to screen.

Five Star Rating

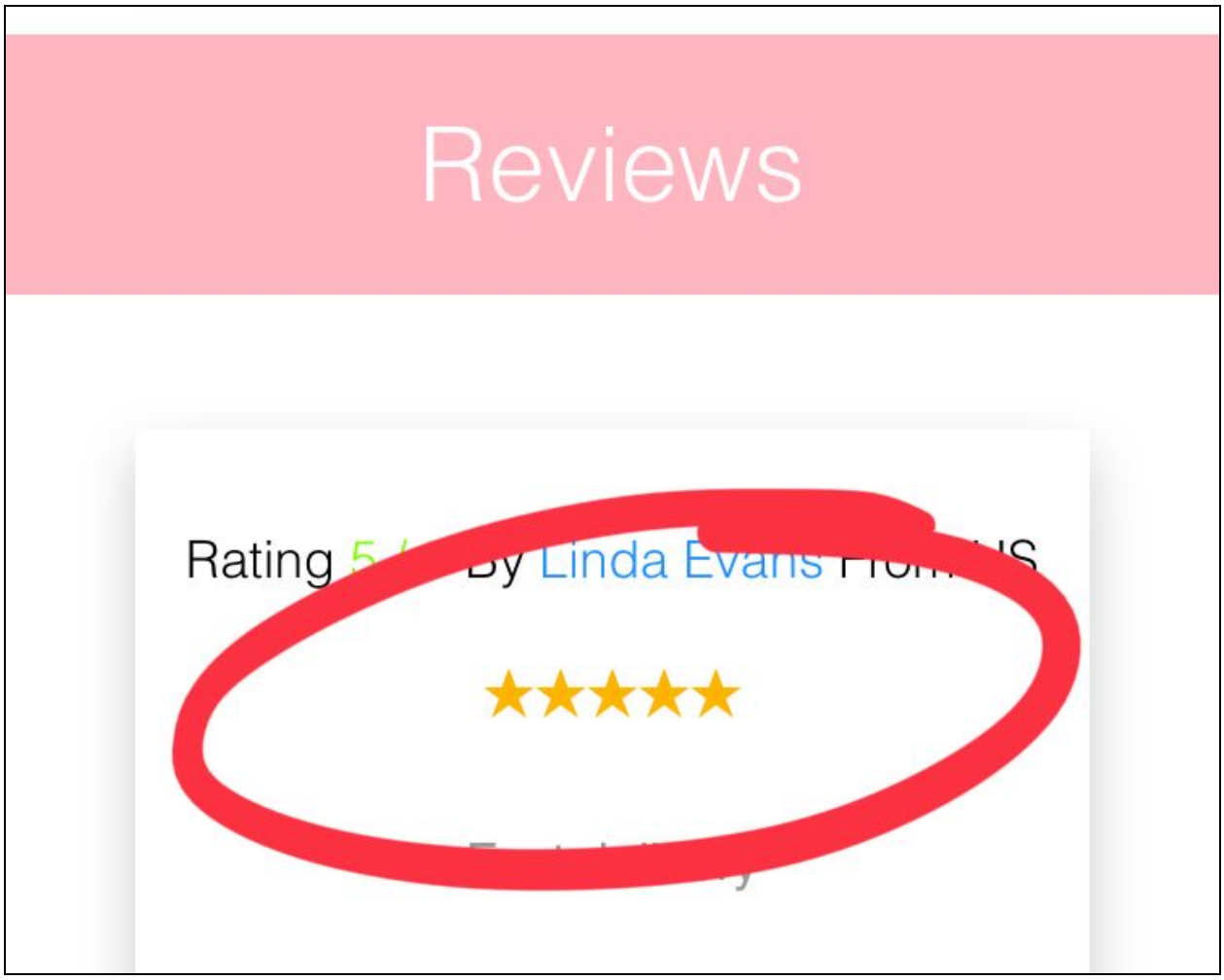


Fig. 40 Five Star Rating

The React-Star-Rating-Component was used to implement the five gold star rating system on the reviews screen. The component itself was downloaded as an npm package from:

https://www.npmjs.com/package/react-star-rating-component

The package was installed as follows:

```
npm install react-star-rating-component --save
```

It was then implemented within the PMA using the **<StarRatingComponent>**

```
32
33          <div>
34              <StarRatingComponent name="starRating" starCount={5} value={inst.rating} />
35          </div>
36
```

Fig. 41 Five Star Rating Component

The value of this component is derived from the **rating** element of the returned JSON object from the WebHose reviews API call. It can hold a value ranging from 1 - 5 which determines the amount of gold stars the product is awarded.

WebHose Pricing

The PMA is subscribed to WebHose on a free plan which allows for 1000 API calls per month. WebHose operates an incremental payment system for:

1. Web Data - Reviews, Blogs, Online Discussion
2. E-Commerce - Product Data

| API Calls(Web Data) | Price (per month) |
|---|---|
| 2000 | $50 |
| 20,000 | $200 |
| 1,000,000 | $4000 |

Fig. 42 WebHose Pricing

Similarly, E-Commerce API calls are free for the first 1000. However, after 1000 calls they are priced higher than Web Data.

## 6.3 FS-002 Price Compare

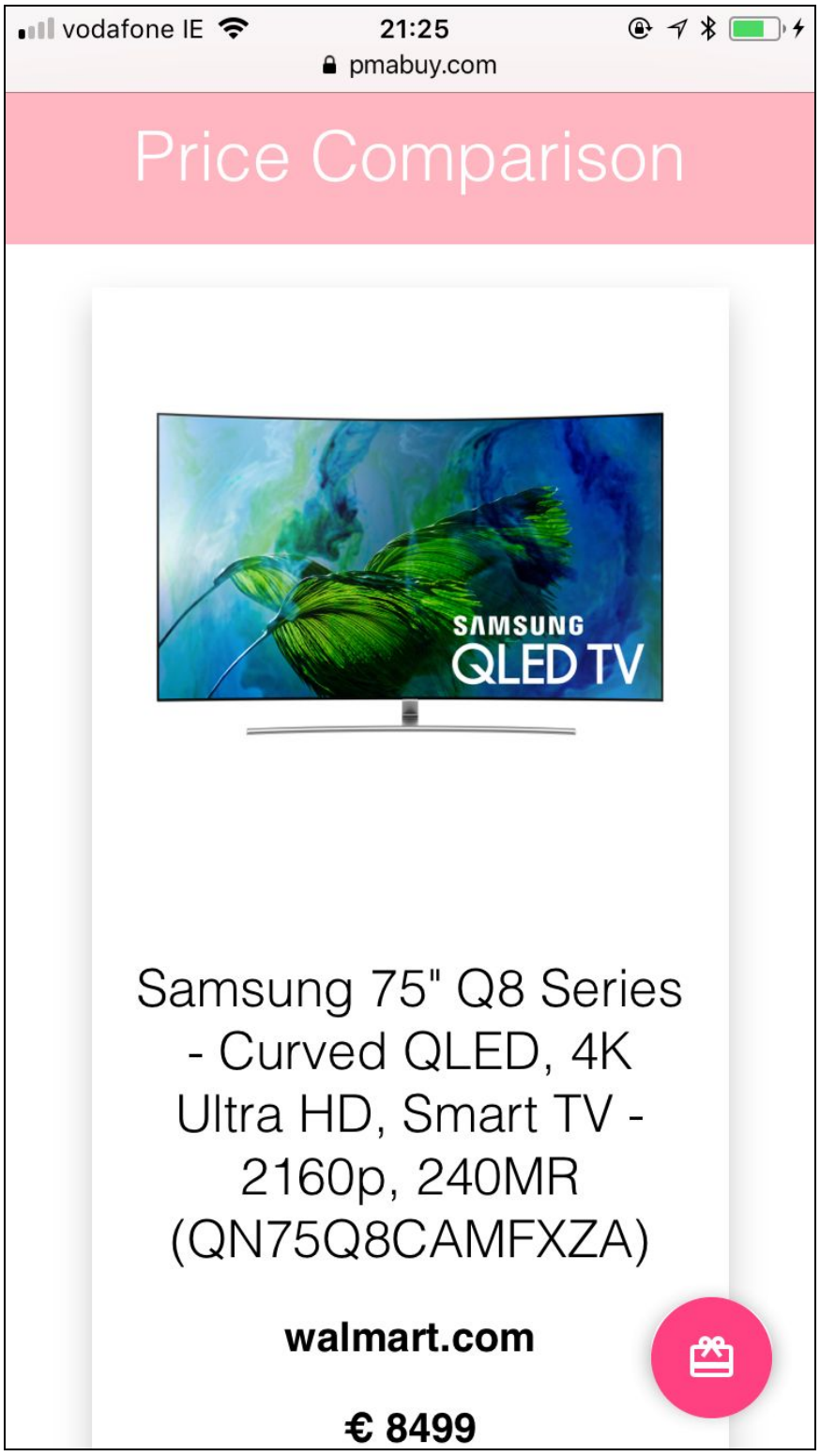


Fig. 43 Price Compare

What is the Price Compare Component?

The price compare component gives the user a price comparison with other similar products on the market from retail websites across the internet. It gives the user a sense of choice and value and reinforces the notion that they are getting a good deal.

Create Price Compare Component Endpoint Query

The WebHose E-Commerce API was used to obtain information regarding the price of similar products on retailer websites across the internet. Like the product reviews, the price comparison uses the WebHose active API key. Also like the product review component, the process of creating a GET request to query the WebHose API is exactly the same for price comparison. The only difference is the search criteria are defined using the WebHose E-Commerce Query Builder instead of the Reviews Query Builder. Refer back to Fig. 30 and Fig. 31 as the process and the UI are the exact same.

As mentioned, when creating the URL to query the E-Commerce API it was necessary to set search criteria and also set filters. In this case the PMA was looking for Samsung 4K Smart TVs with a filter of English language.

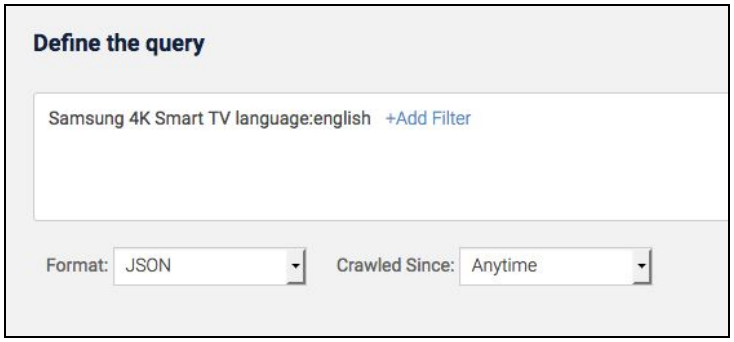


Fig. 44 Define Search Criteria for Price Comparison

A list of products is returned as an array of JSON objects shown below:


```
{
    "products": [
        {
            "url": "http://omgili.com/ri
                /.wHSUbtEfZTzBH6c9Z4cI15_sA3HthoyViwuCuY_SpwtuFgIdHqTjE0968gBabHc587wlb_H8H4WBKqBhAKOqqhadIlIDiTS483XPBnpM_
                .9Gw68xISSj20kAk6xRgWWnq1hK.wilp..1zZbhM0K0Sd7VieAa9B3FczrQyH3sTU-",
            "uuid": "9f45c5814b321fc3a107e845faee9ba52da1a2ca",
            "source": {
                "site_full": "www.walmart.com",
                "site": "walmart.com",
                "site_section": "https://www.walmart.com/browse/electronics/all-tvs/samsung/qled-tvs/3944_1060825_447913
                    /YnJhbmQ6U2Ftc3VuZ3x8dGVsZXZpc2lvbl90eXBlOlFMRUQgVFZz",
                "section_title": "Samsung QLED TVs All TVs - Walmart.com",
                "country": "US"
            },
            "name": "Samsung 75\" Q8 Series - Curved QLED, 4K Ultra HD, Smart TV - 2160p, 240MR (QN75Q8CAMFXZA)",
            "description": "Samsung 75\" Q8 Series - Curved QLED, 4K Ultra HD, Smart TV - 2160p, 240MR (QN75Q8CAMFXZA)",
            "brand": "samsung",
            "price": 8499,
            "currency": "$",
            "offer_price": 4997.99,
            "model": null,
            "manufacturer": null,
            "in_stock": true,
            "on_sale": true,
            "product_id": "55500573",
            "sku": null,
            "mpn": null,
            "colors": [],
            "aggregated_rating": 4.6429,
            "best_rating": null,
            "worst_rating": null,
            "rating_count": null,
            "reviews_count": 28,
            "categories": [
                "electronics",
                "home page",
                "shop tvs by type",
                "qled tvs",
                "tv & video"
            ],
            "width": null,
            "height": null,
            "weight": null,
            "depth": null,
            "images": [
                "https://i5.walmartimages.com/asr/cda49a5b-478f-4a28-aa99-06a86aa3329e_1.ef561fd49cf2def8d2b14e094bf4b494
                    .jpeg"
            ],
            "language": "english",
            "last_changed": "2017-08-03T06:56:45.003+03:00",
            "crawled": "2017-10-09T02:34:54.001+03:00",
            "product_history": "/productHistory?token=485b74f2-1b38-4a80-9fb3-19819afae4e2&productId=http%3A%2F%2Fomgili
                .com%2Fri%2F.wHSUbtEfZTzBH6c9Z4cI15_sA3HthoyViwuCuY_SpwtuFgIdHqTjE0968gBabHc587wlb_H8H4WBKqBhAKOqqhadIlIDiT
                S483XPBnpM_.9Gw68xISSj20kAk6xRgWWnq1hK.wilp..1zZbhM0K0Sd7VieAa9B3FczrQyH3sTU-"
        },
```


Fig. 45 Price Compare JSON Response

Like with the product reviews, in order to keep the price comparison current - a 30 day filter was imposed along with the format specification shown below.

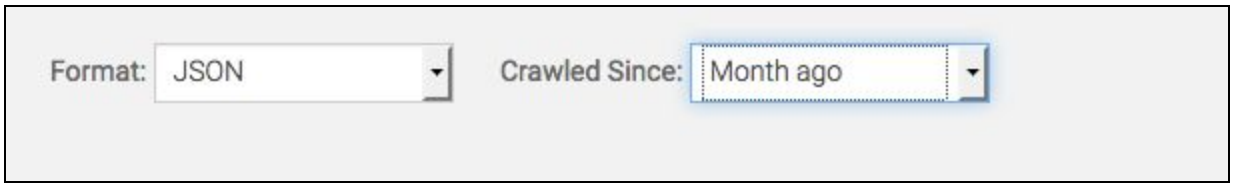


Fig. 46 Specify Format and Time Period for Price Compare Search

Again the PMA is only interested in certain data elements from the returned JSON object in Fig. 45.

- Line 13 - **name**

  Name of the product
- Line 16 - **price**

  Price of the product
- Line 8 - **site**

  Retailer website
- Line 43 - **images**

  Image of the product

The data above was extracted from the JSON object and presented to the user as shown earlier in Fig. 43. The following section describes how this process was implemented.

Product Price Compare Implementation

Similar to the reviews component, the price compare endpoint URL is assigned to a constant within the Compare component of the PMA.

```
const url = 'https://webhose.io/productFilter?token=485b74f2-1b38-4a80-9fb3-19819afae4e2&format=json&q=4K%20Smart%20TV&size=4';
```

Fig. 47 Price Compare Endpoint URL

Looking closely at the end of the URL, a size parameter is set to equal 4.

```
'&size=4';
```

Fig. 48 Size Parameter

Setting the size parameter to equal 4 indicates the PMA would only like 4 products returned in the JSON object. Otherwise hundreds of products matching the search criteria would be returned, adding an extra data burnden onto the device and overwhelming the user with information.

Using the URL in Fig. 47 as its argument, a **fetch()** method calls the WebHose E-Commerce API from the **componentDidMount()** lifecycle method.

```
componentDidMount() {
    fetch(url).then(products => {
        return products.json();
    }
    ).then(data => {
        let electronics = data.products.map((inst) => {
```

Fig. 49 Query E-Commerce Endpoint using fetch()

The asynchronous call returns a promise containing the response JSON object.

Much like the reviews component, the relevant elements are extracted from the object and rendered to screen.

```
let electronics = data.products.map((inst) => {

    return (
        <div className="card" key={inst.products}>
            <img src={inst.images} alt="Compare" className="imgComp"/>

            <div className="container">
                <h2 className="compTitle">{inst.name}</h2>
                <h3>{inst.source.site}</h3>
                <h3>€ {inst.price}</h3>
                <hr/>
            </div>
        </div>
    )

})
```

Fig. 50 Render JSON Element to Screen

The result can be seen in Fig. 43 in which the user is presented with the name of the product, the product image along with the price and retailer.

## WebHose E-Commerce Pricing

The cost of using the WebHose E-Commerce API was discussed in the previous section. Please refer to Fig. 42.

## 6.4 FS-003 Vouchers

What is the Product Voucher Component?

As well as product reviews and a price comparison, the PMA offers users the chance to avail of a product discount through the use of digital vouchers. From the Product Screen, the user can tap on the vouchers icon which will take them to the Vouchers Screen via a social sign-in process.

The user must be signed into the application to get a product voucher code.

Why Vouchers are Good for Business?

Digital vouchers are effective way to help sell products and promote brand loyalty. Giving customers a discount is an easy way of creating a positive shopping experience. The better the experience, the more likely a customer will return to the shop. Because the user must sign-in to avail of the product vouchers, the PMA stores the user data which can be used in future digital marketing campaigns.

Voucher Implementation

The first step in implementing the voucher system is to register with voucherify.io and obtain:

1. Application ID
2. Application key

Both of these credentials are needed in order to interact with the Voucherify API.

From the custom dashboard at voucerify.io it was possible to carry out the following actions in order to get the voucher system operational:

1. Create campaign
2. Create vouchers and add them to the campaign
3. Create products
4. Assign vouchers to products

## Voucher System Setup

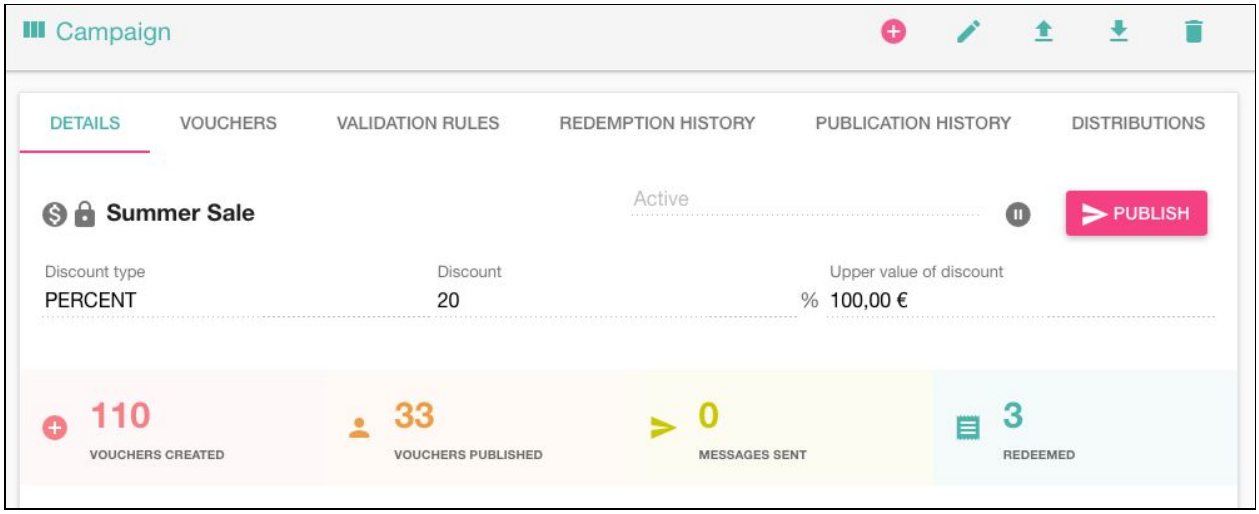


Fig. 51 Create Voucher Campaign

For the purposes of this project a campaign called 'Summer Sale' was initialised. It was then necessary to create product vouchers that the customer can use.

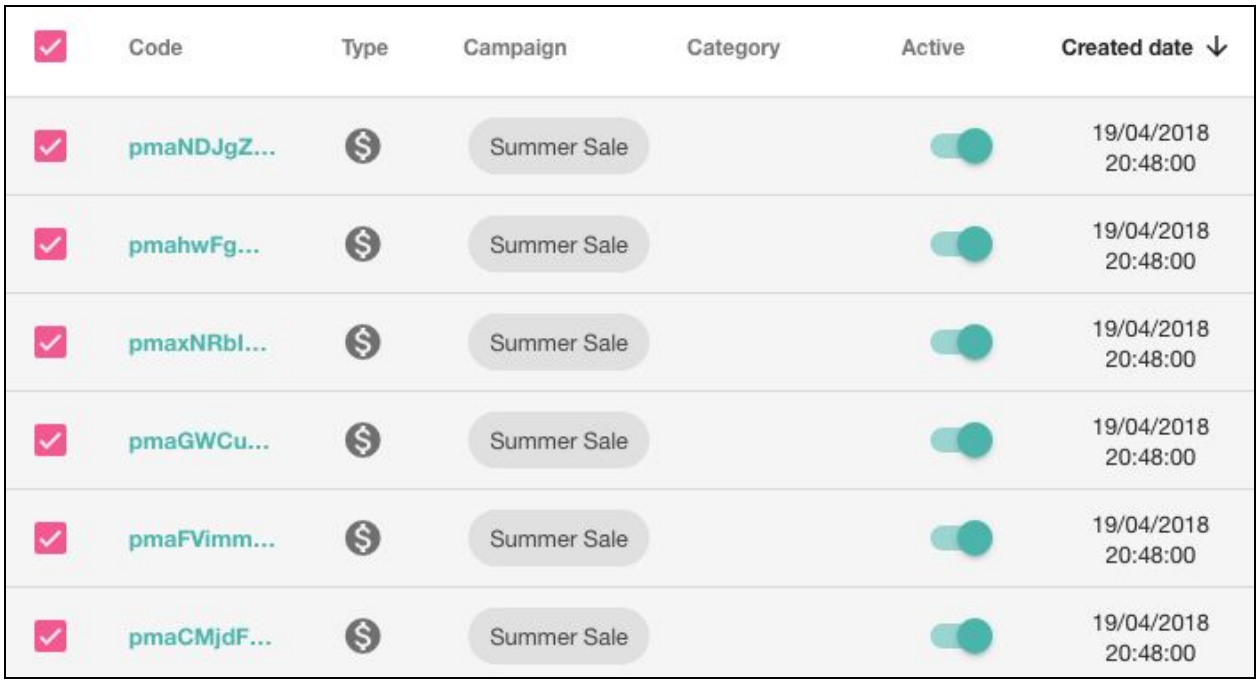


Fig. 52 Create Vouchers

Theses vouchers, shown Fig. 52 are then assigned to the 'Summer Sale' campaign which is shown in Fig. 51. The products that will be sold in-store are then added to the campaign. In this case the products are:

- Samsung Smart Ultra HD TV
- Asics Running Shoes


Fig. 53 Products in Voucherify Campaign

Finally, to complete the voucher system setup process, the vouchers are assigned to the products and a discount value is set as shown below.

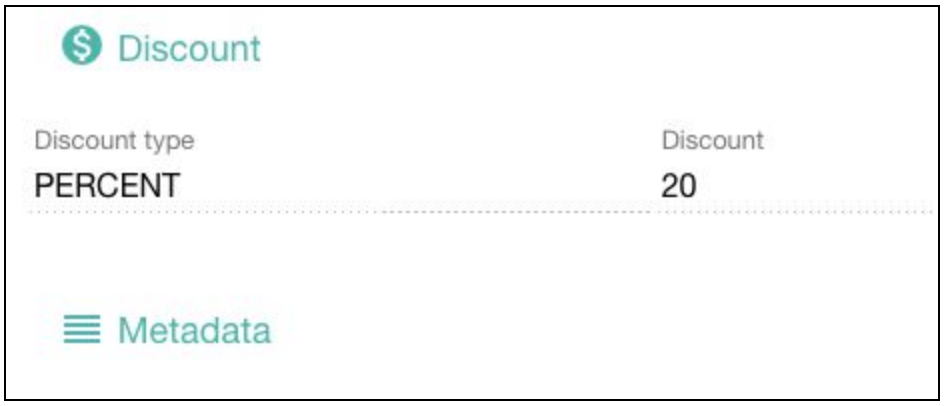


Fig. 54 Product Voucher Discount

In this case the value of the voucher is set to 20% discounted from the retail price.

Once the user completes the authentication process they are taken directly to the Voucher Screen which can be seen below.

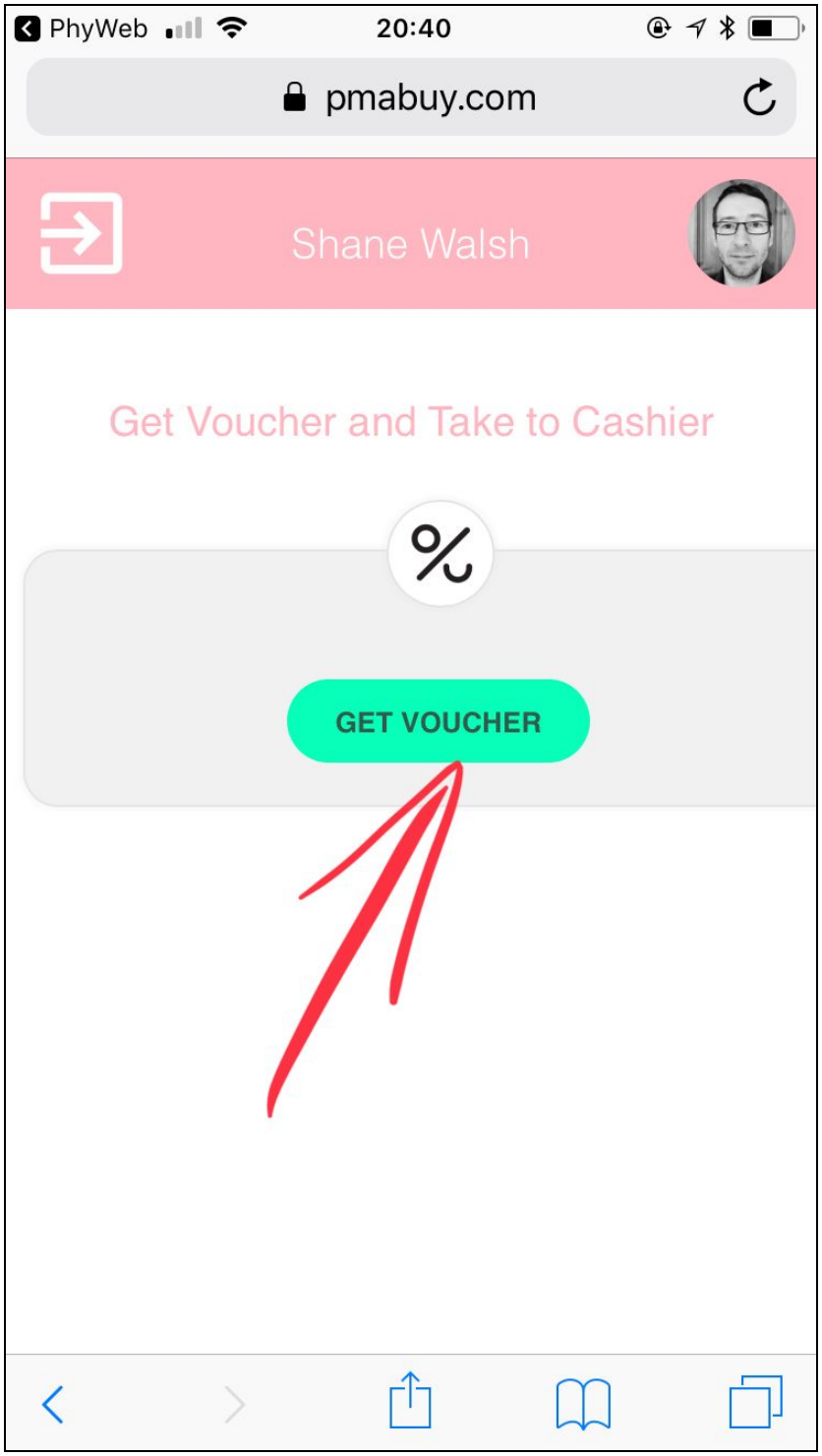


Fig. 55 Voucher Screen

The top banner contains the user's photo to the right, the user's name in the centre and a sign-out button on the left. Further down, the user is given the opportunity to tap the 'Get Voucher' button which automatically generates a voucher code that can be redeemed at the checkout.

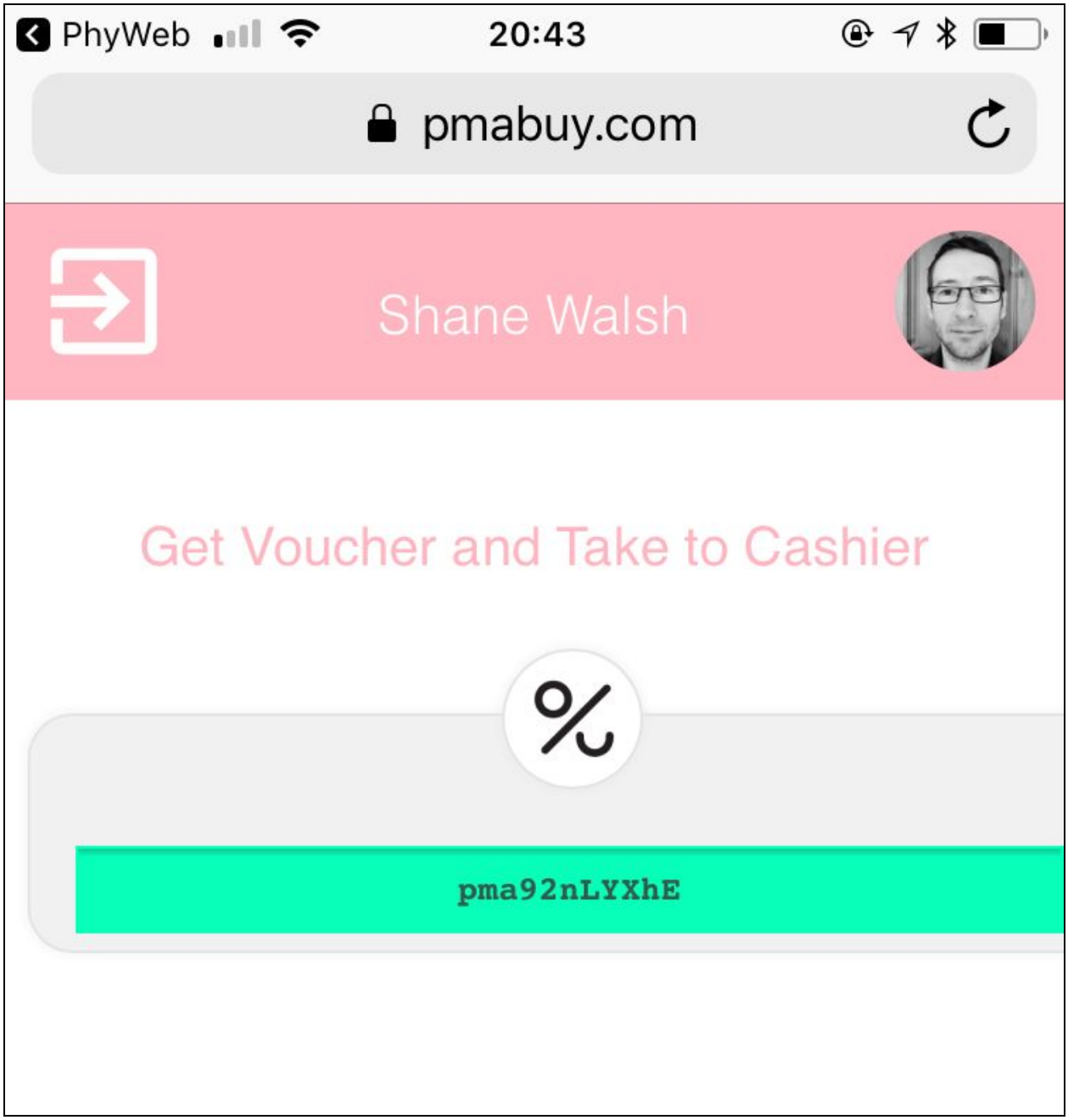


Fig. 56 Displayed Voucher Code

All voucher codes start with the first three letters ‘pma’ and then randomly generated characters after that.

You’ll notice the voucher code in Fig. 56 ‘pma92nLYXhE’ corresponds to a voucher in the ‘Summer Sale’ campaign - shown below 20% discount.

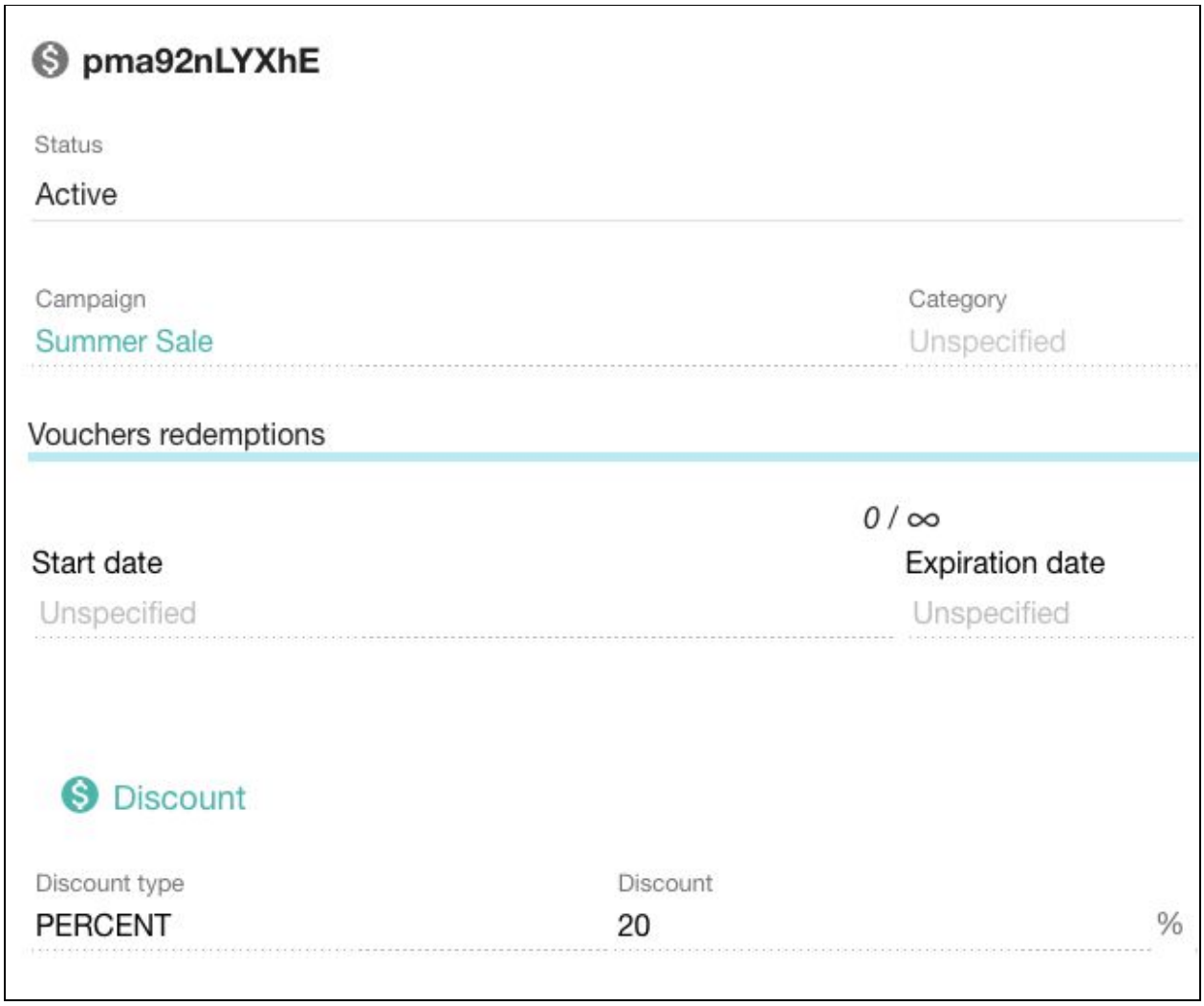


Fig. 57 Voucher pma92nLYXhE

The following points breakdown the PMA voucher process:

- Customer gets active voucher
- Customer is assigned to that voucher
- Customer redeems voucher

- Voucher is deactivated

Once the user takes the code to the cashier, the voucher redemption process is complete. Vouchers can only be used once.

Voucherify Pricing

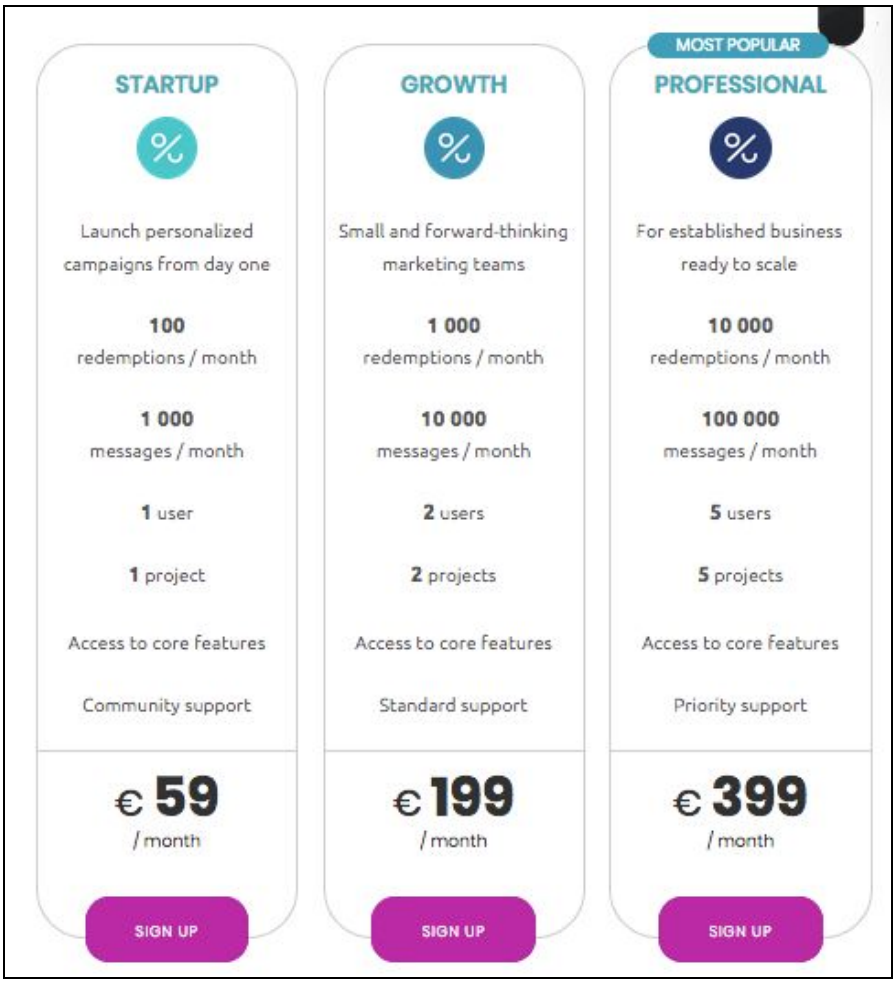


Fig. 58 Voucherify Pricing

A 3 month subscription to voucherify.io was deemed a very worthwhile investment as its integration broadened the functionality of the PMA and ultimately enhanced its usefulness for both customers and retailers alike. It must be noted that with special offers, the subscription price is significantly less than that shown in Fig. 58.

# 7. Application Data

This section describes all the data used by the PMA including:

- User data
- Product reviews data
- Product price comparison data
- Voucher data

## 7. 1 Data Type Tables

### User Data

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| displayName | String | Yes | User's name |
| uid | Number | Yes | Unique ID |
| email | String | Yes | User's email |
| avatar | String | Yes | URL to image |

Fig. 59 User Data Table

### Reviews Data

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| author | String | Yes | Reviewer's name |
| country | String | Yes | Reviewer's origin |
| rating | Number | Yes | 1-5 star rating |
| text | String | Yes | Body of review |

Fig. 60 Reviews Data Table

### Price Comparison Data

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| image | String | Yes | Product image |
| name | String | Yes | Product title |
| site | String | Yes | Retailer |
| price | Number | Yes | Product price |

Fig. 61 Price Comparison Data Table

### Voucher Data

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| voucherify_uid | String | Yes | Unique ID |
| Name | String | Yes | Product Name |
| Product_ID | Number | Yes | Product ID |
| Source_ID | Number | Yes | Source ID |
| price | Number | Yes | Product price |
| Created_date | Date | Yes | Date created |

Fig. 62 Voucher Data Table

## 7.2 Class Diagram and Entity Relationship

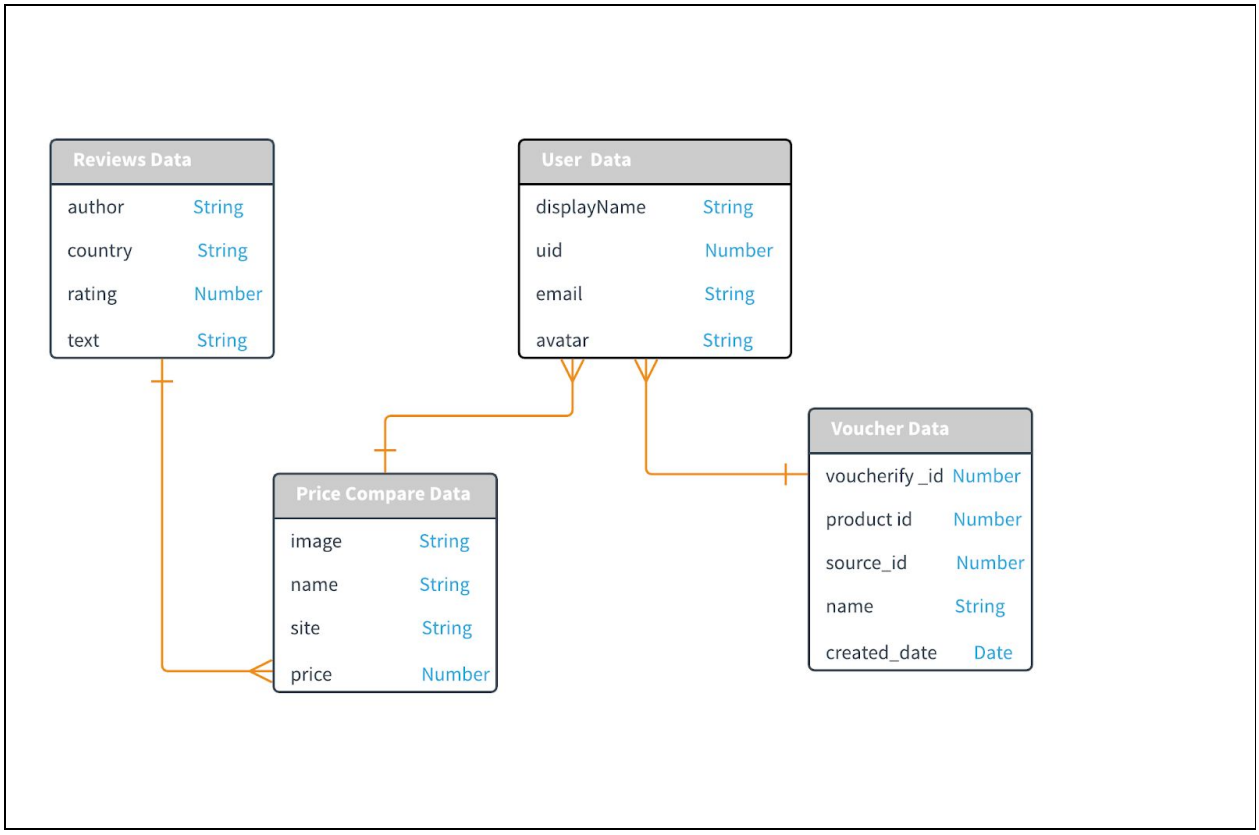


Fig. 63 PMA Data Relationship

## 7.3 Firestore

For data persistence the PMA uses Firestore, a NoSQL flexible and scalable database. It is a cloud based database that keeps data in sync across client apps through realtime listeners and offers offline support for mobile and web so that it is possible to build responsive and progressive apps that work regardless of network latency or Internet connectivity.[17] The following code snippet outlines how the PMA writes user data to Firestore:

[17] https://firebase.google.com/docs/firestore

```
36
37      handleUserFirestore(user){
38
39          // Initialize Cloud Firestore through Firebase
40          const db = firebase.firestore();
41
42          //add user_info to firestore
43          db.collection("user_info").add({
44              name: user.displayName,
45              email: user.email,
46              voucherify_uid: user.uid,
47              avatar: user.photoURL
48          })
49              .then(function(docRef) {
50                  console.log("Document added with ID: ", docRef.id);
51              })
52              .catch(function(error) {
53                  console.error("Error adding document: ", error);
54              });
55      }
56
```

Fig. 64 Writing User Data to Firestore

The result can be seen in Firestore as shown below:

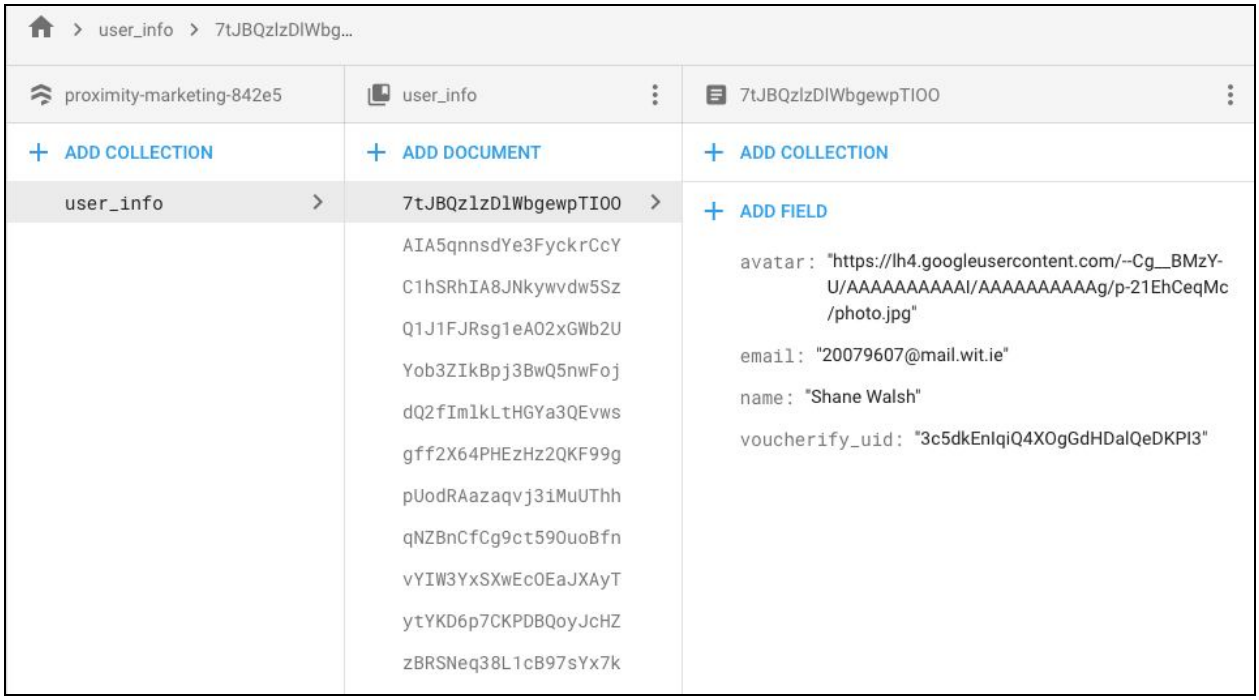


Fig. 65 User Data in Firestore

# 8. UX Development

This section explains the presentation of the PMA in terms of user experience. As the application is intended to be used by customers in a busy shopping environment it was decided to keep the UI minimal and as straightforward as possible. Therefore, the product description, the product reviews and the price comparison are presented to the user on a single screen. To access the reviews and price comparison, the user simply scrolls down. As shown in the package.json file, the PMA uses two UI frameworks:

- material UI
- semantic UI

Both of these frameworks offered React components that helped build and present the application UI in a fully responsive nature, adapting to different screen sizes effortlessly.

## 8.1 Notification Screen UI & Icon

Every smartphone notification screen is different and brand specific. However, this screen is the first point of contact for the user with the PMA. The URL is customised to reflect the product. For instance, Fig. 66 below shows two PMA URLs, one for a Samsung TV the other for a pair of Asics running shoes. A user can tell from just a quick glance which link is related to which product.

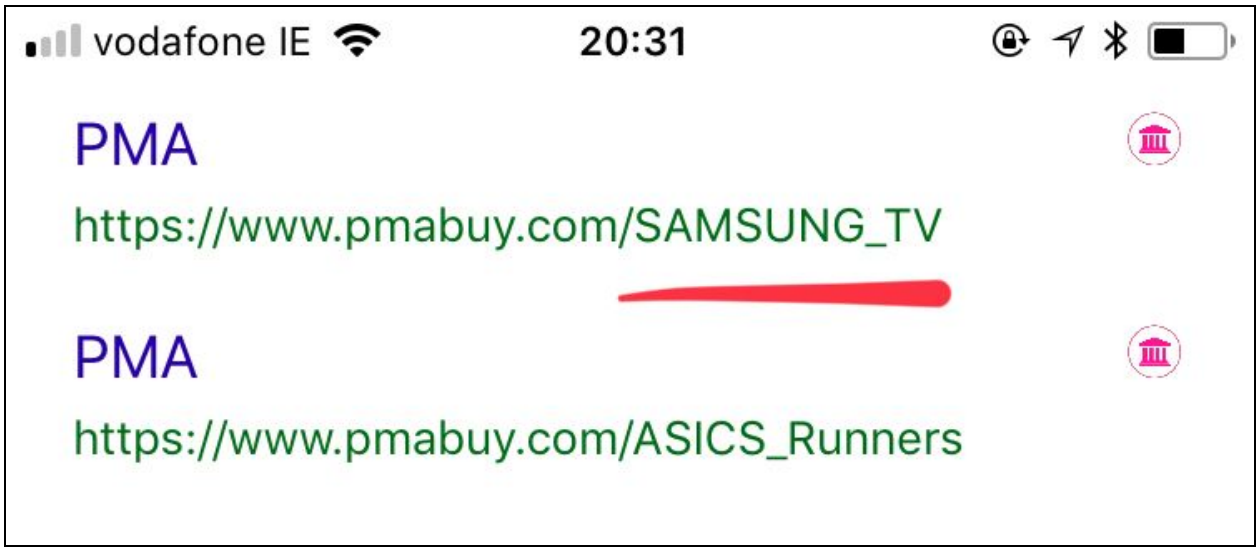


Fig. 66 PMA URLs in Notifications Screen

The PMA icon is also displayed to the right of the URL. The icon is intended to be eye-catching and match the colour scheme and style of the application. It was hoped that having a bright icon, as shown above in Fig. 66, would help the application stand out in a list of notifications and be easily identifiable to the user.

## 82 Product Screen UI

The layout of the product screen is broken down into 3 parts:

1. Title
2. Image
3. Description

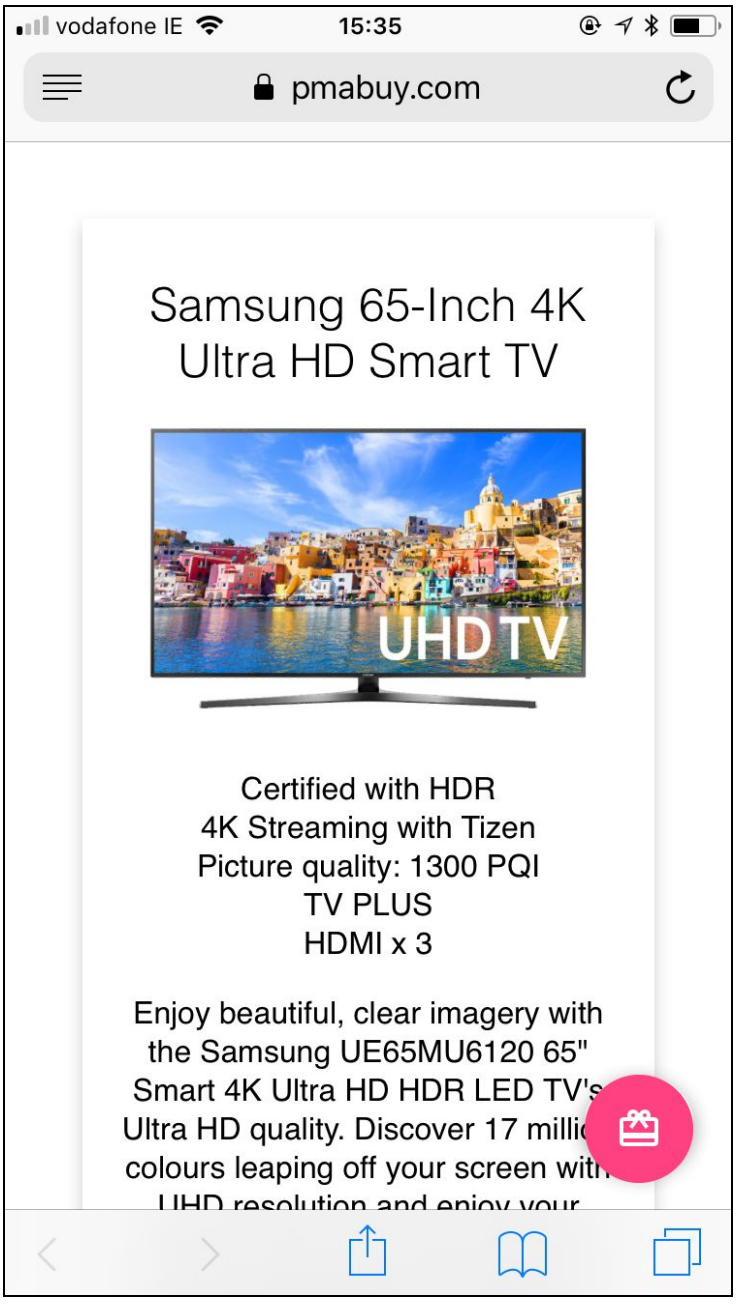


Fig. 67 Product Screen

The Product Screen is the first screen the user lands on after tapping the link on the notifications screen. All three parts mentioned above are grouped together and contained within a HTML **<card>** element. Using custom CSS, a shadow effect is applied to the card to make it stand out and appear raised from the background. Also, the shadow transitions to a darker colour when the card is in focus.

All text on the Product Screen is black on a white background making it clear and easy for the user to read.

## 8.3 Product Reviews Screen UI

To access the Product Reviews Screen, the user simply scrolls down. It was the intention to the make the user experience smooth with a continuous flow from one application component to the next. Each component is divided by a section header with white text on salmon pink background as seen below.

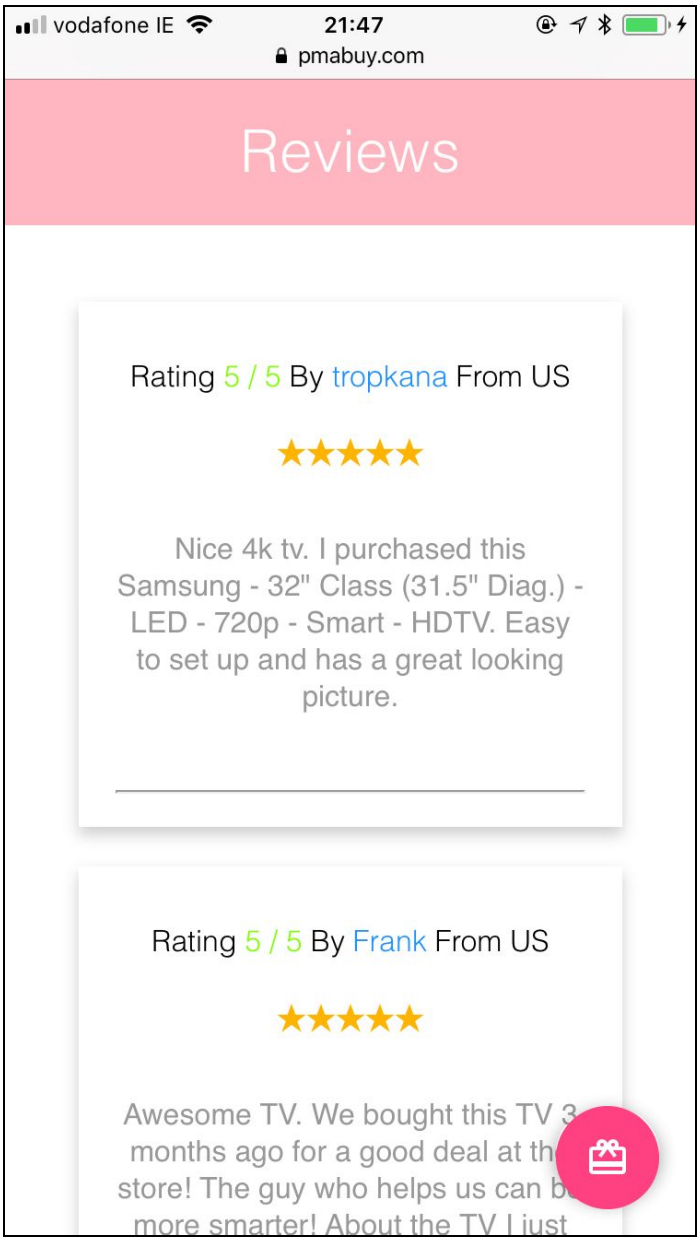


Fig. 68 Reviews Screen

Like the product description, all the product reviews information is grouped together on a HTML **<card>** element. Each customer review appears on its own centered individual card. The same shadow and transition effects apply to all cards on this screen. The layout is broken down as follows:

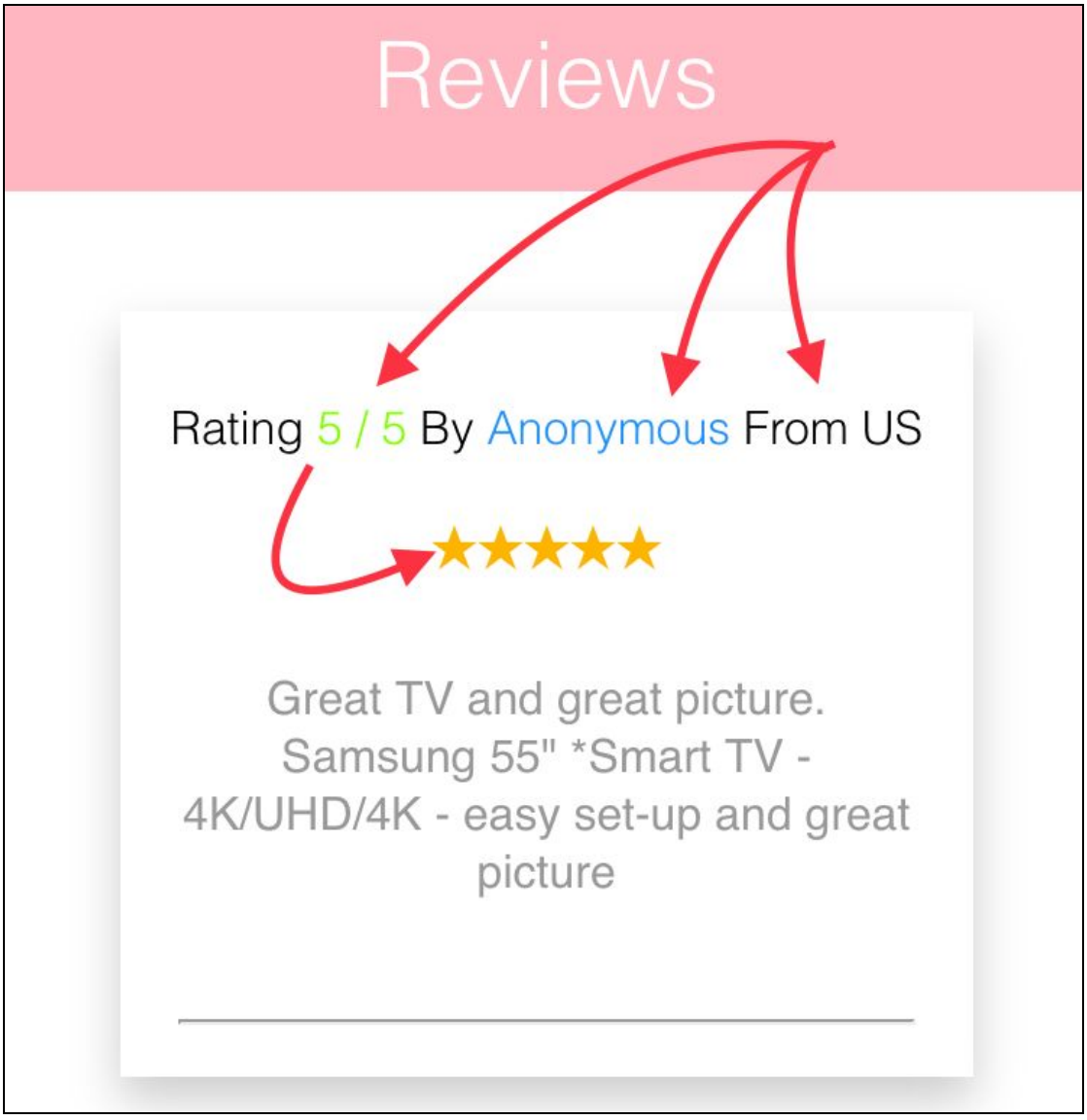


Fig. 69 Product Review Card

- **Rating -** Numerical product rating colored lime green in order to stand out from the other info on the card. Green is also a positive reaffirming colour.
- **Author -** The name of the person that wrote the review is listed in blue.

- **Location -** The location of the reviewer is listed in a regular black font colour.
- **Five Star Rating -** Five gold stars indicating the product approval rating. Gold stars are a universally recognised rating system. The number of stars directly correspond to the numerical rating value in green as shown by the arrow in Fig. 69.
- **Review Text -** The text body of the review is presented in a lighter grey colour.
- **Rule -** A horizontal rule indicates the end of the review.

## 8.4 Price Comparison Screen UI

A HTML **<card>** element is also used to group the Price Comparison information. The same style and effects are applied to this card and the layout is as follows:

- **Image -** Image of the product is set to the centre of the card.
- **Title -** Name the product is displayed in black text, along with the model and specs.
- **Site -** Website of the retailer is listed in black and bold text
- **Price -** The price of the product is also listed in bold black text.


Fig. 70 Price Compare

- **Rule -** Horizontal rule indicates the end of the Price Comparison

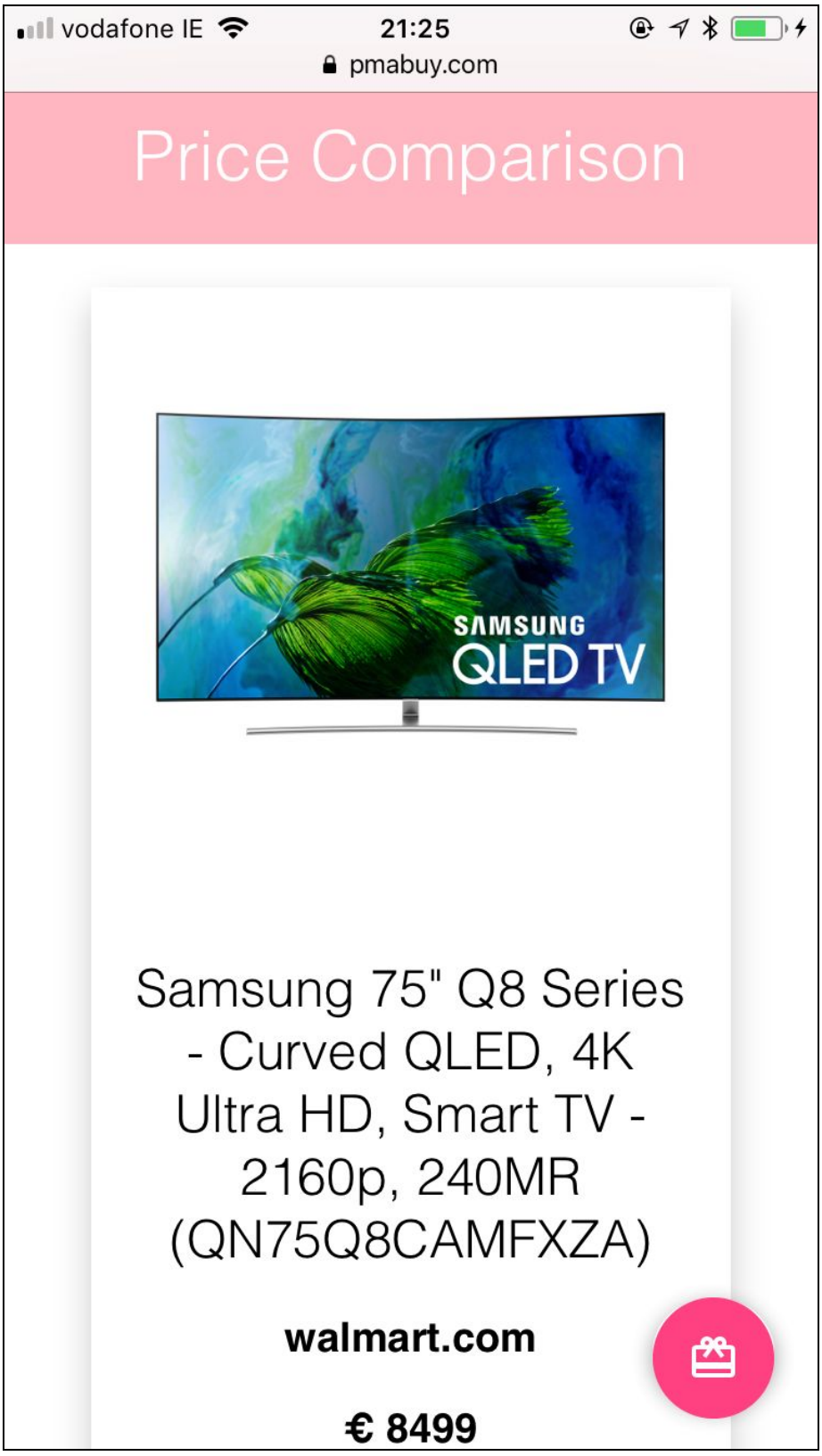


Fig. 71 Price Comparison Card

## 8.5 Product Vouchers Screen UI

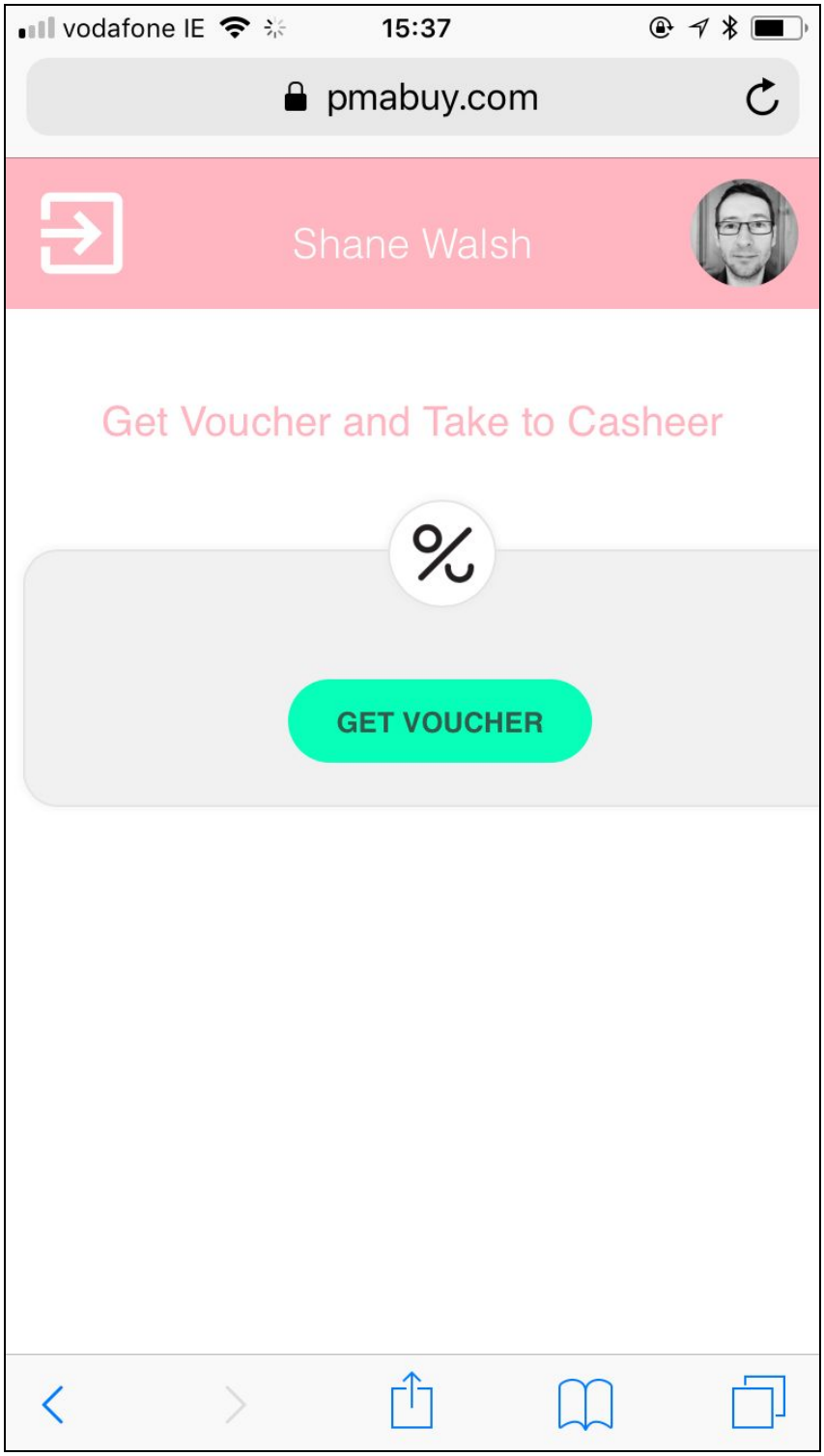


Fig. 72 Voucher Screen UI

The salmon pink colour scheme is continued into the vouchers component with a banner at the top of the screen containing:

- User photo - aligned right
- Username - aligned centre
- Sign-out button - aligned left

A instruction header follows under the banner, again salmon pink text colour in keeping with the UI theme.

The 'Get Voucher' feature consists of an iframe with a Voucherify logo centered at the top. Within the grey panel sits the voucher button which generates the voucher code.

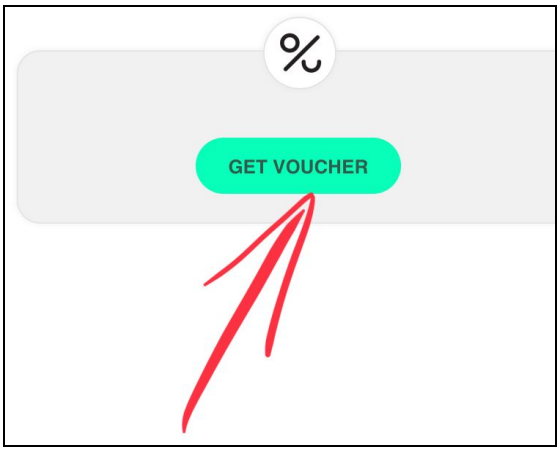


Fig. 73 Get Voucher iframe

Once triggered, the button switches to a 'pending' state as shown below.

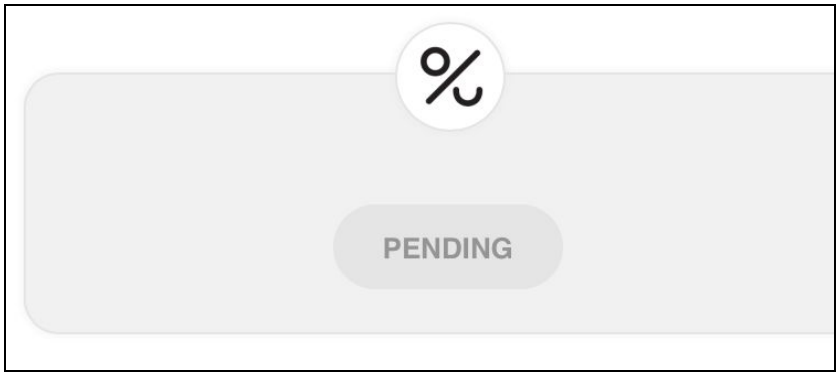


Fig. 74 Get Voucher iframe Pending State

The generated voucher code is returned as black text on a lime green background to the centre of the iframe as follows:

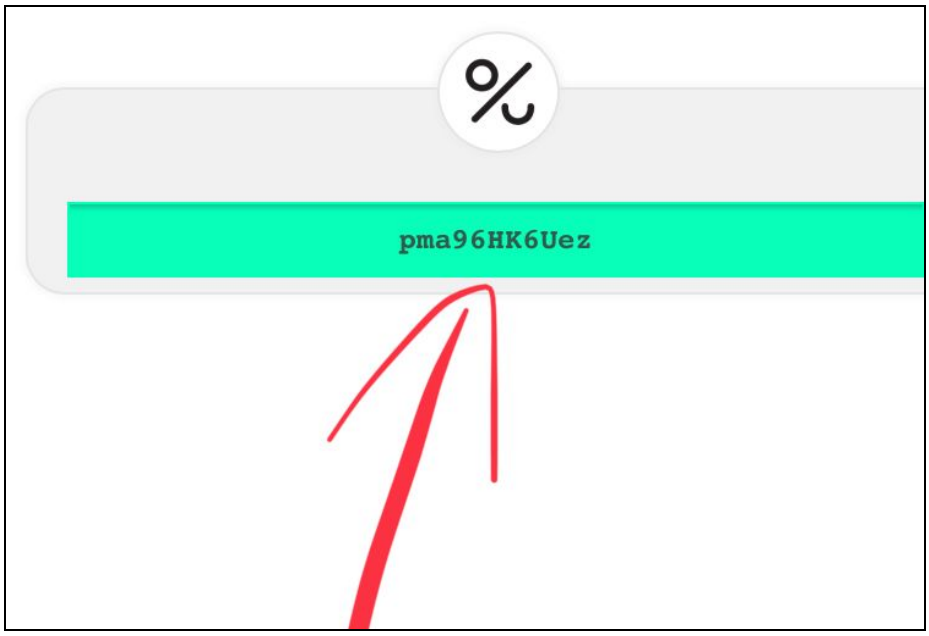


Fig. 75 Voucher Code

Please see Appendix C for further discussion on social sign-in and navigation UI.

# 9. Conclusion

The goal of this project was to develop a meaningful web application that allows users to access product reviews, compare product prices and redeem digital vouchers using Bluetooth beacon technology. At the end of this development journey I feel the project goal has been achieved and that a high value product with real-world application has been delivered. Without doubt building a web application with new and unfamiliar technology is a challenging task. Even though the learning curve was steep, on a personal level I found this project to be an exciting opportunity to implement new technology and grow as a software developer. The challenges discussed in the next section reflect the complexity of integrating hardware (Bluetooth beacons) and software (React/Firebase web app). Improvements to the project and future development plans are also highlighted and discussed in the pages below.

## 9.1 Challenges

**Flux Design Pattern**

As demonstrated in Section 4.4 Architectural Design, React implements a one way data flow design pattern. For me, this was a completely new concept in systems architecture as I was only familiar with the more traditional Model-View-Controller (MVC) pattern. It was necessary for me to rethink my approach to data control and ultimately learn new methods of data flow such as Flux.

**Google Support for Physical Web**

Late in 2017 Google's Chrome Team dropped support for the physical web on Chrome for iOS. This means that Chrome on iOS devices can no longer

automatically detect Bluetooth devices such as the Bluetooth beacons. For iOS owners that wish to use the PMA, it is necessary to install a Physical Web app that can instantly recognise Bluetooth beacons in nearby locations. Even though this is an extra step that was not necessary at the conception of this project, iOS users can still access and use the PMA by installing the Physical web app for free. Google dropping support for Chrome on iOS should not be viewed as a lack of confidence in beacon technology, however, as the decision was made due to the low numbers of Chrome users on iOS. Generally iOS users default to the Safari browser with very few installing Chrome on their devices. Recent figures indicate that Safari accounts for 96.9% of all traffic on iPhone and iPad devices - which suggest that Chrome's 3.1% market share does not warrant Google's resources and physical web support. [18]

### Routing Issues on Mobile

When testing URL routes on mobile, it was found that access was denied to all routes other than root. For instance:

https://www.pmabuy.com ← Loaded ok.

https://www.pmabuy.com/SAMSUNG_TV ← Access denied.

The issue did not occur on localhost:3000 which made it more difficult to debug and find the causation.

After much research, it was found that the issue was due to Firebase hosting permissions. The solution involved rewriting permissions into firebase.json so that the app can access the source and destination directory from mobile browser.


Fig. 76 firebase.json rewrites

[18] https://www.zdnet.com/article/which-browser-is-most-popular-on-each-major-operating-system

### API Calls not Responding

Making GET requests to WebHose API returned the following error message:

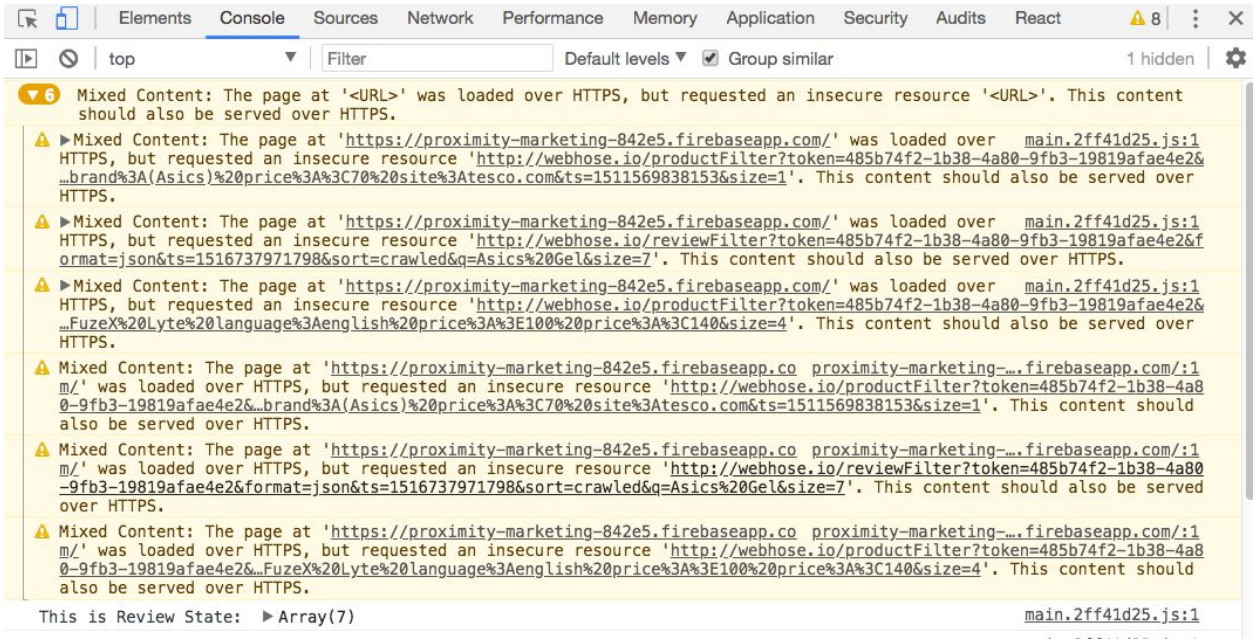


Fig. 77 Mixed Content Error

The error occured because all Physical Web content must be served securely over HTTPS. The solution involved serving the API GET requests with HTTPS instead of HTTP.

## 9.2 Improvements - What would I do differently?

### Product Reviews Content Control

In the current version of the PMA it is not possible to filter the content of the product reviews. Some reviews that return from the API call can be quite large and so expand the HTML card in which they are contained. This forces the user to scroll down for quite some time to reach the next section which can have a negative effect on the user experience. An improvement would be to only accept reviews that are less that so many characters. Also in relation to content control, some reviews are not very informative and so it would be of great benefit to implement a

filter system that selects only relevant reviews. However, it is possible to exert more control over API filtering with a paid subscription to WebHose.

**Graphic Design**

Customise the entire look and feel of the application from the logo to the UI navigation by a professional graphic designer.

**Ask for Advice Sooner**

On a number of occasions throughout the development journey I hit roadblocks on issues that I just could not figure out. The issues seemed trivial on the face of it, however I was wasting a lot of precious development time with little or no success. For instance, the product review API calls were returning empty JSON objects and for some time I could not understand the reason why. After mentioning the problem to my supervisor he suggested the returned JSON may be an array of objects and that I was trying to access the object incorrectly. His solution was correct and I got back on track shortly afterwards.

## 9.3 Future Development

Future iterations of the PMA would include the following features:

**Service Workers**

In an effort to make the application more progressive in nature, the PMA would benefit from offline capability. This can be achieved through the use of service workers. In production a service worker is setup to serve assets from local cache. This lets the application load faster on subsequent visits and gives it offline capabilities.

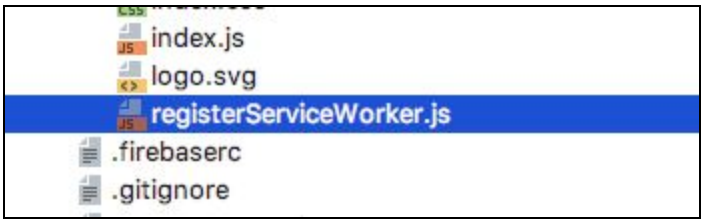


Fig. 78 Service Workers Registry

The PMA is setup to use service workers, however, they are not implemented as of yet.

### QR Codes

Instead of textual voucher codes, it is hoped that QR codes can be implemented in a future iteration - making the process of voucher redemption easier by simply scanning the code.

### Data Analytics

Analytics help brand owners make informed business decisions about the products they sell. By tracking the product data within PMA, analytics can provide indications as to what products are hot and what are not.

The Google Analytics Platform enables the measuring of  user interactions with web applications across various devices and environments. It provides all the computing resources to collect, store, process, and report on these user-interactions.[19] For the purposes of the PMA -  analytics.js would be used, which is a JavaScript tracking snippet that allows for the capture of user data such as: total time a user spends on the app, what internal links were clicked, geolocation and more.

[19] https://cloud.google.com/appengine/docs/standard/python/google-analytics

### In App Purchase

80% of all transactions in Sweden are made by cards. Digital payments via card or apps are so widely accepted that many Swedes no longer carry cash.[20] Sweden may be an early adopter of the cashless society ethos, and even though it may be some time before its widespread - it is good practice to keep in mind possible future trends. For this reason the ability to make in-app payments for in-store products must be considered for any future PMA development plans.

[20] https://sweden.se/business/cashless-society

# References

https://www.unacast.com/post/proximity-marketing-what-how-why

http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons

https://www.unacast.com/post/proximity-marketing-what-how-why

https://www.shopify.com/retail/the-ultimate-guide-to-using-beacon-technology-for-retail-stores

https://developer.kontakt.io

https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

https://medium.com/@CalinLeafshade/why-i-chose-react-over-vue-3dd9a230b507

http://www.insights.stackoverflow.com/trends

https://docs.webhose.io/docs/ecommerce-product-data-api

https://firebase.google.com/docs/hosting/deploying

http://www.prweb.com/releases/2012/1/prweb9086226.htm

https://www.socialmediatoday.com/content/why-online-reviews-matter

https://firebase.google.com/docs/firestore

https://www.zdnet.com/article/which-browser-is-most-popular-on-each-major-operating-system

https://cloud.google.com/appengine/docs/standard/python/google-analytics

https://sweden.se/business/cashless-society

https://www.npmjs.com/package/react-twitter-auth

# Appendices

**Appendix A**

## firebase.json and firestore.rules

```
{
  "database": {
    "rules": "database.rules.json"
  },
  "firestore": {
    "rules": "firestore.rules",
    "indexes": "firestore.indexes.json"
  },
  "hosting": {
    "public": "build",
    "rewrites": [
      {
        "source": "!{/src}/**",
        "destination": "/index.html"
      }
    ],
    "ignore": [
      "firebase.json",
      "**/.*",
      "**/node_modules/**"
    ]
  },
  "storage": {
    "rules": "storage.rules"
  }
}
```

Fig. 79 firebase.json

The firebase.json file acts as an index for the many rules that need to be imposed on Firebase services. For instance, read and write privileges must be set with regards Firestore - Firebase's dynamic cloud storage service.

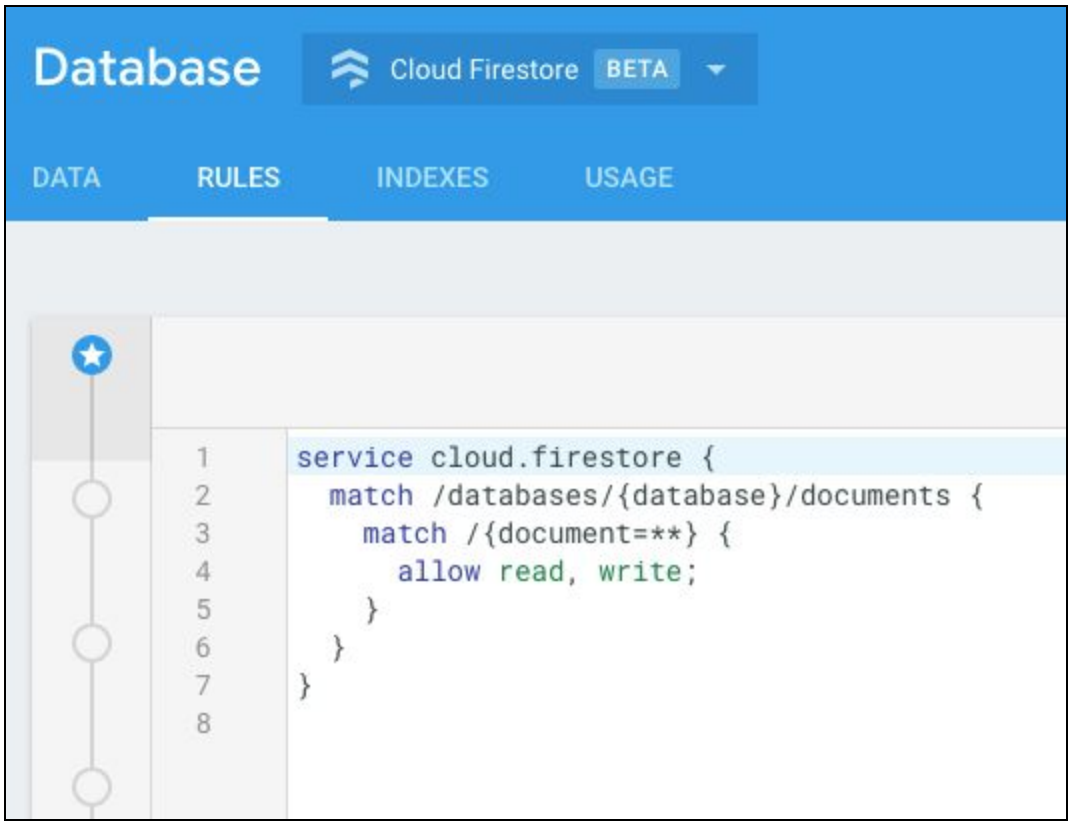


Fig. 80 Firestore rules

The Firestore rules above stipulate that read and write privileges are granted to all documents and collections within the Firestore database.

## Appendix B

## Authentication via Twitter

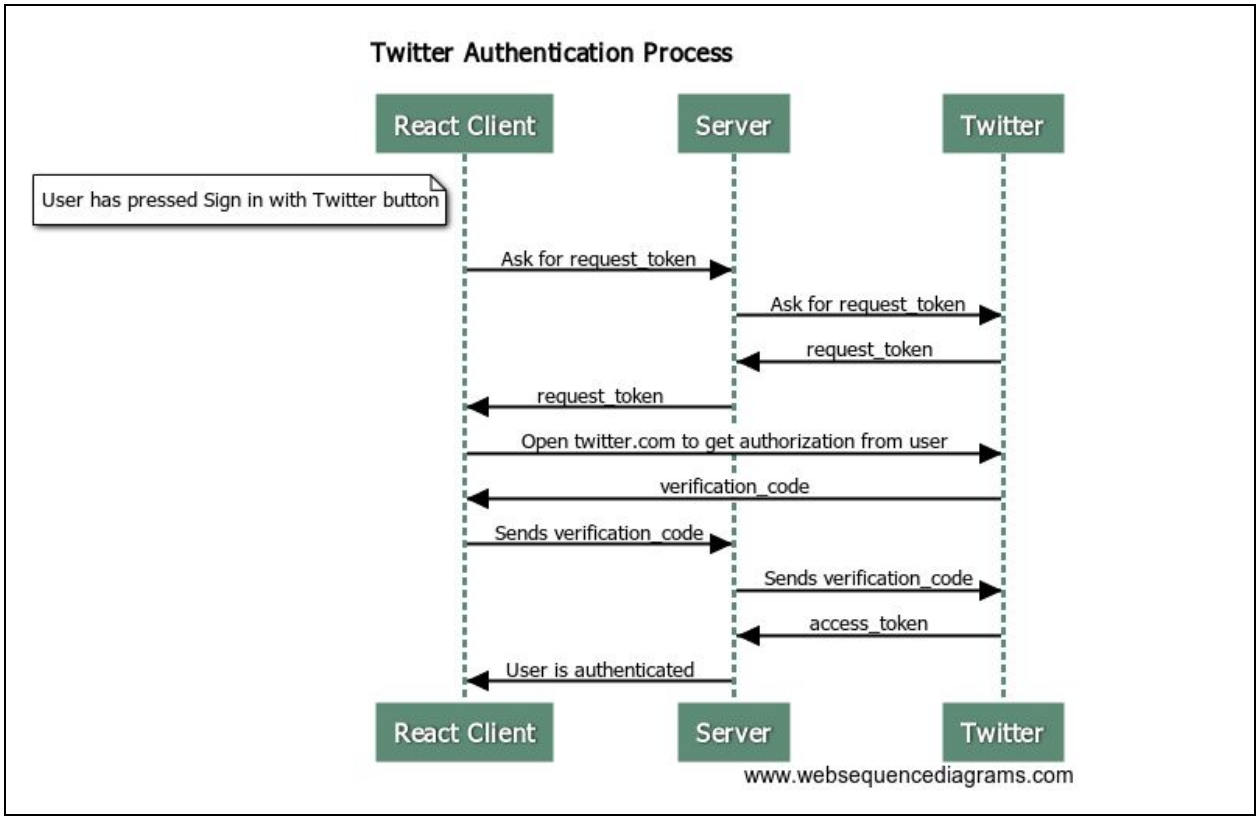


Fig. 81 Twitter Authentication Work Flow[21]

The sequence diagram above illustrates the workflow and steps taken to authenticate users from a React application.

Similar to Google sign-in, Twitter users can be authenticated using the Firebase Authentication SDK.

[21] https://www.npmjs.com/package/react-twitter-auth

```
33
34      handleTwitter(){
35
36          firebase.auth().signInWithRedirect(providerTwitter);
37
38
```

Fig. 82 Firebase Authentication (Twitter)

Users are redirected to Twitter's sign-in page for authentication before being pushed back to the PMA where they continue on to the product voucher component.

## Authentication via Facebook

The same authentication process as Google and Twitter, Facebook users are redirected to Facebook sign-in page before being pushed back to the PMA.

```
var provider = new firebase.auth.FacebookAuthProvider();
```

Fig. 83 Firebase Authentication (Facebook)

Again, the Firebase Authentication SDK plays a key role in handling the OAuth access token, user ID and monitoring status change.

## Appendix C

### Social Sign-in Screen UI

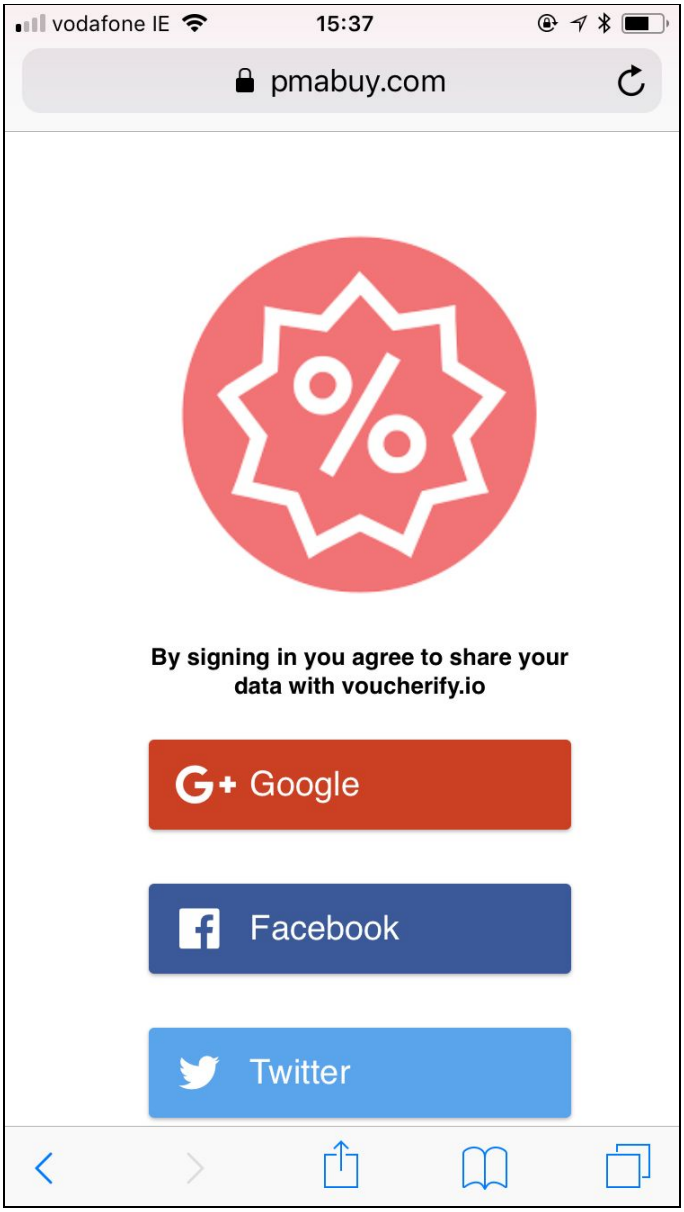


Fig. 83 PMA Sign-in Screen

The sign-in screen presents the user with three options to authenticate via Google, Facebook or Twitter. Above the provider buttons sits a disclaimer informing the

user that personal data will be shared with Voucherify once logged in. Positioned over the buttons and the disclaimer is an image of a percentage sign which suggests a monetary discount.

## Navigation UI

### Floating Button

A floating button serves a the primary navigation mechanism for the product description screen, the product reviews screen and the price comparison screen.

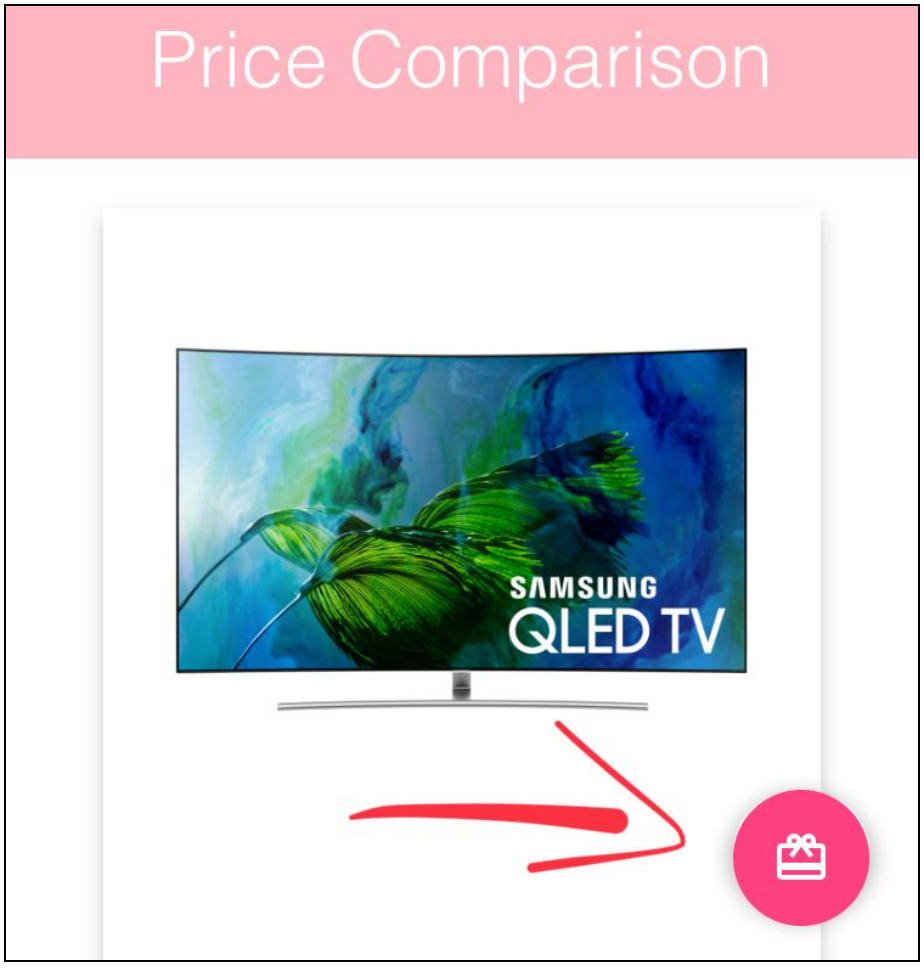


Fig. 84 Floating Action Button

The button, which displays a ‘Redeem Voucher’ icon, holds an absolute position in the bottom right of the screen. As the user scrolls up and down the button remains

ever-present, providing the user a link to the vouchers component via the social sign-in screen. The SVG icon displayed inside the button is made available by the Material-UI React component.

**Sign-out Button**

Located in the top left corner of the voucher screen, the sign-out button allows the authenticated user to exit the application.

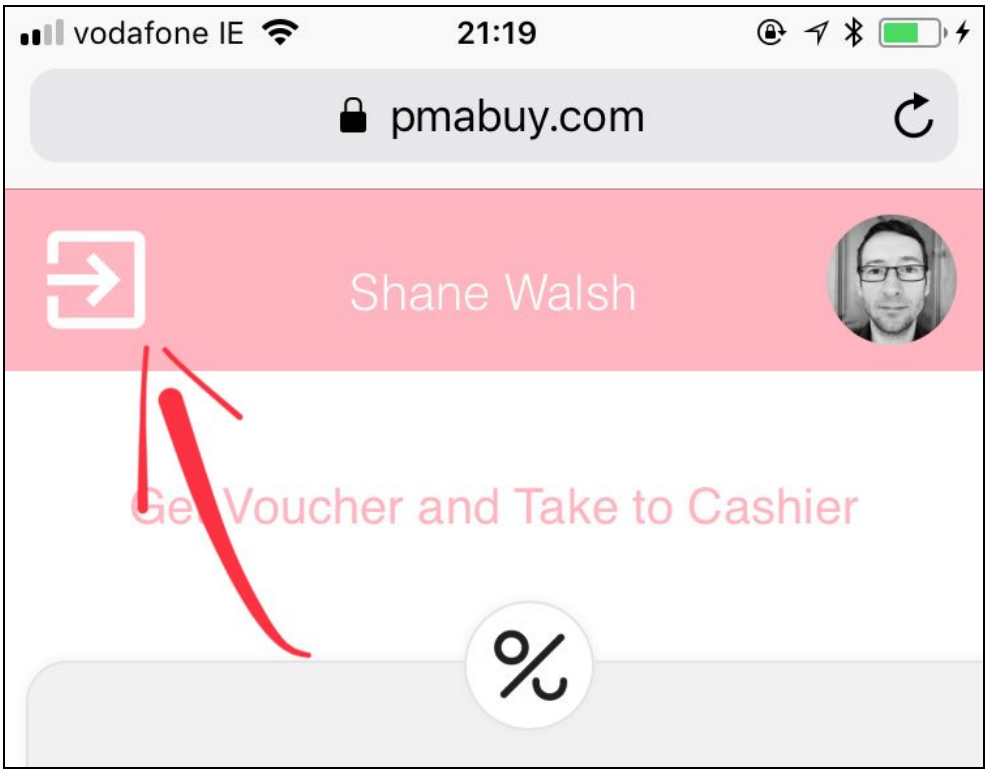


Fig. 85 Sign-out Button

Also made available by the Material-UI React component, once the sign-out button is tapped the user is logged out and brought back the to the product screen. The icon turns lime green on hover to indicate the button is in focus.

## Statement of Ownership

I hereby declare that all the work in this document is my own. Any non-original work is duly referenced.

_____

Shane Walsh