National College of Ireland

BSc in Computing

2014/2015

Serge Salcedo

X16483684

X16483684@student.ncirl.ie

Bizaare E-Commerce

Technical Report

National College of Ireland

Table of Contents

Executive Summary

This report's main purpose is to provide an in-depth analysis for my Software Project. The goal of this Software Project is to develop an online web application with commercial functionalities, better known as an E-commerce website. My final aim for this project is to successfully implement said website, to be accessible to multiple users where they can set up and sell their personalized products and make it accessible to other users to purchase.

To create this idea, I will utilize the following software programs and programming languages: React with Node.js and Yarn package manager, Google's Firebase, HTML & CSS, JavaScript and JavaScript XML and Visual Studio Code as my text editor. Google Firebase will handle the data and store it through the backend. Yarn with React uses JSX language to achieve a mix of HTML and JavaScript to provide a frontend to my project.

This Technical Report will go through the steps I took to complete this project at start to finish. Outlining the goals, requirements, difficulties faced, different aspects of the project to finally getting it live and operational.

1 Introduction

The purpose of this document is to serve as a technical report for the development of my web application. This application will provide features such as e-commerce, user authentication and database storage. I will briefly go through the project scope and do an overview of everything included in the requirement specifications section. A list of abbreviated terms with their definitions will be provided below.

1.1 Background

I based this project between my two interest of art and computing. I did research into this topic and found applications available to artists, designers and creators and found the topic of selling handmade artwork interesting. This led my investigation further into how artists would sell their personal works and found multiple ways they could achieve this. These ranged from creators setting up their own websites to subscription services such as Patreon, that would require a user to authenticate to set up a form of payment system to see their work- to setting up a physical showcase or gallery and renting out a space that people can see.

I decided on creating a web application from this research that would require a user to sign up to a system. From there they set up stalls and perform tasks with a minimal UI and a single page application where functions are used to call windows and dialogues. Here users can arrange stalls and arrange a price for each item. A basket handles all these orders until the user decides to check out. I want to implement a way where users can sort through stalls to find what interests them. I also want to develop another personalized section where users can set up profiles add other users and message each other, a blog section where users can post and comment under them.

Also, since I am studying under the Cyber Security Stream, I will have to implement several input validations to prevent any database attacks or data injections

1.2 Aims

My aim for this project is to create a successful web application that multiple users can set up an account for. Once logged in users can manage a stall and import items they would like to sell onto the site, they can also set up a profile image where they can edit their handles, description and set up posts for other users to see and let them contact each other with messages. Stalls will be displayed in the market area of the web application and users can sort through different stalls. Users setting up these stalls can choose images and set a price for each item they are selling. Images stored on the site will be stored in Googles Firebase platform.

Two factor authentications can also be set up for users to give extra security to prevent their accounts from being easily hacked. I want to also implement a gateway for making online payments, the most popular being Paypal. The application will need to be tested to ensure that security features are being implemented before it is released to live hosting. The application as mentioned before will require the use of Googles Firebase services to handle data storage on the backend. A text editor such as Visual Studio Code will do to handle coding and it will most likely be made using JavaScript since Google Firebase can be used in tandem with it. React uses JSX that joins HTML with JavaScript as its own language and it can be used to create the front end of the application.

1.3 Technologies

My web application will be built using HTML and CSS, JSX uses JavaScript

to enable the use of HTML and JavaScript together in the same document. I will be using Visual Studio Code as my preferred IDE to code with and install the Node package manager to enable the use of the React module within Node. I will be using Google Firebase as my method of database and hosting as it does all the backend work for you using its own framework and installing it is simple.

For performing transaction tasks, I have investigated different forms of online payment and Paypal is by far the most popular and most integrated platform.

Transactions will need to be confirmed and sent to users to tell them that their purchase has been received. Images will have to be either stored locally or stored as mentioned above into the Google Firebase system. Encryption of user data is therefore handled on Googles side as it receives data and encrypts it before anyone can see it.

1.4 Structure

This document is structured as follows:

Section 1. Introduction: describes the background, purpose, technologies and aims of the project. It is self-explanatory in its headings and it goes into detail for each one.

Section 2. System: contains the applications functionality, integrations, data handling. Security and environmental requirements. Overall, giving an idea of how the website itself works.

Under this section is also the design and architecture which will further explain how the application is structured. It explains how a connection to a database is created. Implementation explains certain code and how they were inserted into the application and how they affect it. The Graphical User Interface or GUI as its more commonly known shows how the websites front end is created and what is shown to the user when they access the application. Testing describes the different types of testing methods used on the application to check and display these vulnerabilities. Finally, the conclusion is the observation by me, the developer on this section for System.

Section 3. Further Development: outlines any improvements that can be made to improve the application from what it currently is now. As well as provide any updates that can possibly enhance the experience of its users.

Section 4. References: the references used to help achieve the implementation of the application as well as any information that I found of use to be interesting are displayed here.

Section 5. Appendix: lists the project proposal, monthly journals and finally a guide on how to set up the application for yourself.

Abbreviations and Definitions

Term	**Definition**
User	Person who interacts with application
UI	User Interface: Front end design the user interacts with
GUI	Graphical User Interface: Branch of the UI specifically the graphics of the application and what the user sees
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets: style of a document
JavaScript	Object Oriented scripting language
JSX	JavaScript XML a mixture of both HTML and JavaScript put into one document
Gateway	Connection between two networks
Firebase	Googles platform for creating mobile and web applications
Two Step Authentication	Requires the user to send another information other than their password
IDE	Integrated Development Environment, a code editor

2 System

2.1 Requirements

This section highlights all the functional requirements in my system and gives a detailed description of the system along with all of its features as well as how these users will interact with these features.

1. <Register/Login>

Description & Priority

A user will need to Register or Login to access the Bizaare Web Application. This allows users to process an order.

Use Case

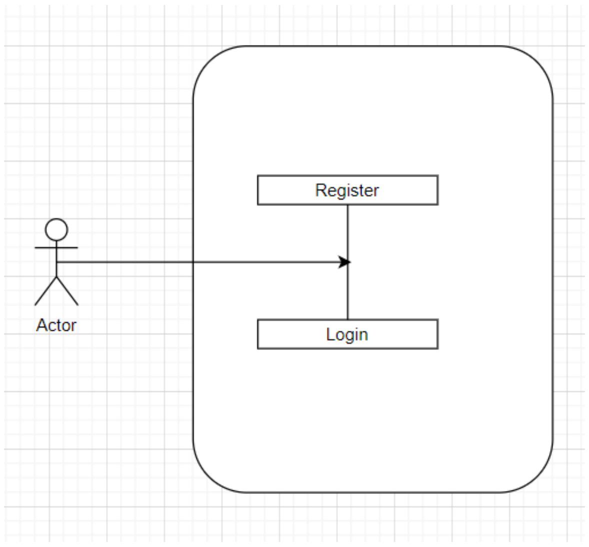
Scope

Show how the user creates an account and how that is processed in the application.

Description

The use case describes a user creating an account by using their email account.

Diagram



Flow Description

Precondition

System is idle and User is on the Application and is currently connected to an internet connection

Activation

The use case begins once the user presses the sign up/login button

Main Flow

1. User clicks on the Signup/Login button in the Navigation Bar
2. A window pop asks user to choose a Gmail account to use
3. The user selects and sets a Gmail account (A1)
4. Google initializes and authenticates user
5. A cookie token is placed in the browser
6. Window closes and user is placed back in the application
7. Use case ends

Alternate Flow

A1: <Create account>

1. User decides to login with an email account already added to their browser
2. User creates a new email account with Google mail
3. Continue use case from this step

Exceptional Flow

User forgets their email account details:

1. User remembers password or resets password
2. User closes login window

Termination

If Login is a success, then user is redirected back to page

Post Condition

Success:

1. User cookie is saved to the browser
2. User can now confirm their order at checkout

Failure:

1. User is unable to proceed to checkout

2.<Logout>

Description & Priority

A user wants to remove their logged in data from the application. This removes user data and disables their ability to confirm an order

Use Case

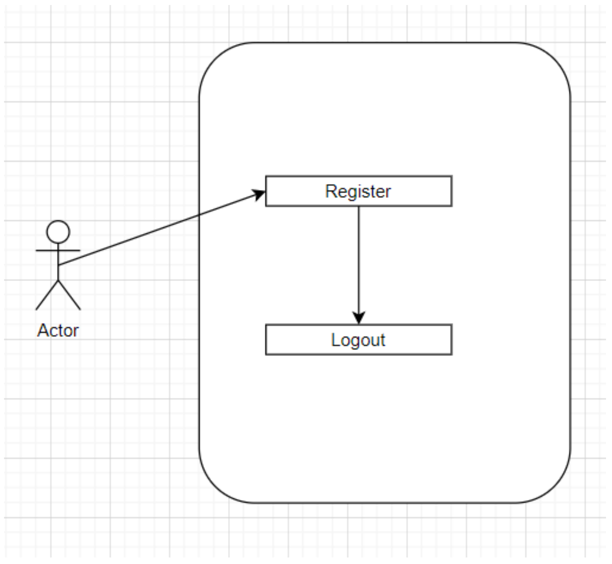
Scope

Show how the user successfully logs out of the application

Description

The use case describes how a user presses the logout button and removes their data from the application

Diagram



Flow Description

Precondition

The user is currently logged in or has registered and is idle

Activation

The use case begins once the user clicks on the Logout button

Main Flow

1. The user clicks on the Logout button on the Navigation Bar
2. Remove access to confirm order button

Alternate Flow

N/A

Exceptional Flow

N/A

Termination

If Logout is a success user has to login or signup again to access confirm button

Post Condition

The user is now logged out and has lost access to Confirm Order

3.<Selecting Item & Add to basket>

Description & Priority

Use Case

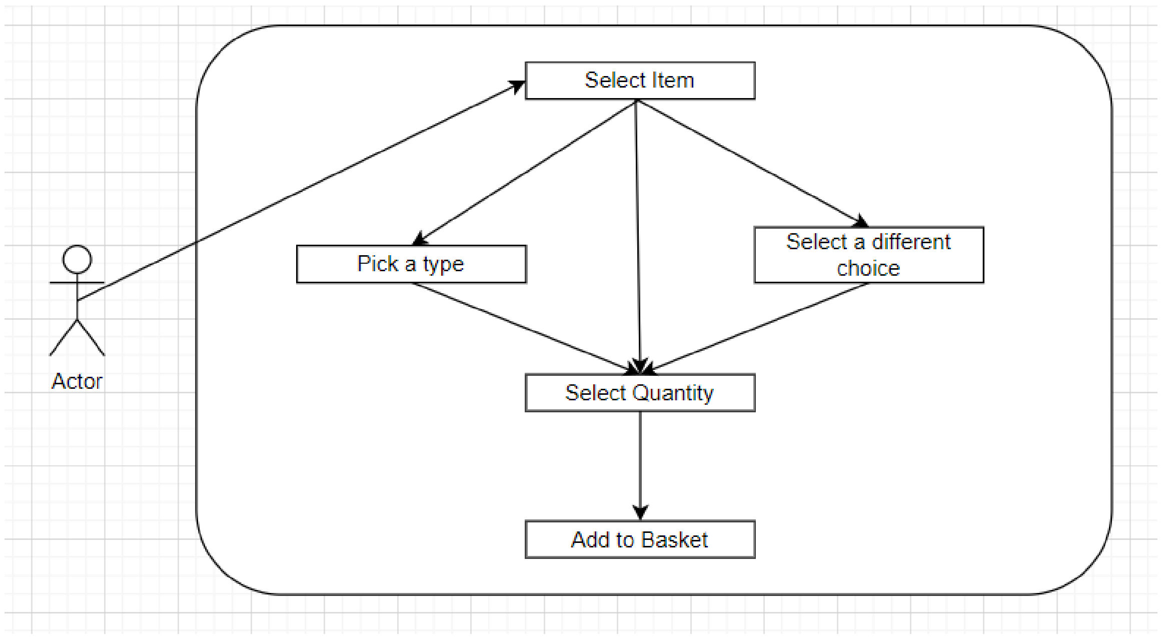
Scope

To show how a user can select an item within a card and send it to the basket

Description

The use case describes how a user can select a card and select an item from it. Depending on the item the user is forced to select an option and/or they can select a choice. User selects a quantity then it is saved on to the basket as an order.

Diagram



Flow Description

Precondition

The user is idle. The user selects a card, and it shows information on an item.

Activation

The user browses the stalls and clicks on a card displaying an item that interests them

Main Flow

1. User clicks on a card
2. Card displays information on the item (A1)
3. User selects an option if its available (A2)
4. User sets a quantity to how many of this item they want
5. User clicks add to basket
6. Item along with conditions if any, are sent to basket
7. Items are now displayed in basket

Alternate Flow

A1: <Close card>

1. The user decides not to add to basket and closes window
2. User clicks outside of window in greyed are
3. Window closes
4. Resume on Main Flow 1

A1: <Selecting a condition>

1. If the user is given a choice on the item
    a) User selects from the various options
    b) Use case continues
2. If the user is forced to select from given options on an item
    a) Use selects from the various options
    b) Use case continues

Exceptional Flow

N/A

Termination

The basket has been updated with the item along with its preconditions

Post Condition

Success Condition:

1. The users selected items are now inside the basket

4.<Editing Basket>

Description & Priority

Use Case

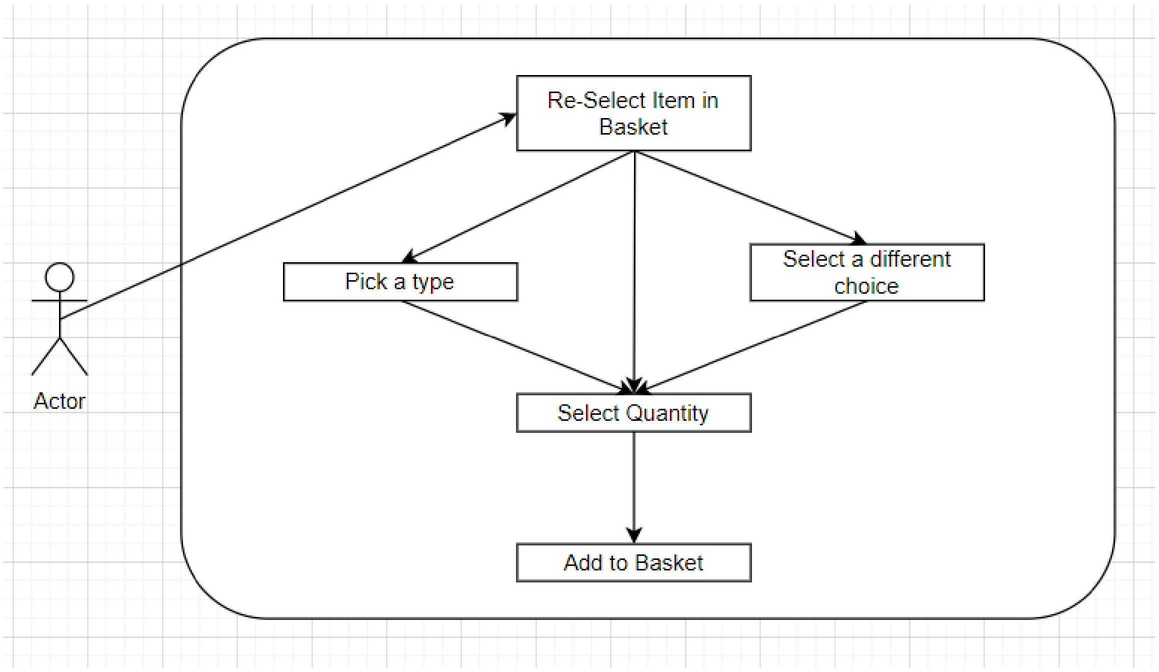
Scope

The user wants to edit their choices on an item or reselect its quantity if it is in their basket

Description

The use case describes how a user can select on an item in the basket and change the choices of an item or change the quantity they have selected prior to editing

Diagram



Flow Description

Precondition

The user has items added in their basket on the application

Activation

User clicks on an item in their basket

Main Flow

1. User clicks on item in the basket
2. Window pops up allowing them to change the current options (A1)
3. User changes option
4. Item conditions change
5. User reselects add to basket
6. Item is added to basket with new conditions

Alternate Flow

A1 <User decides not to edit>

1. User selects outside of window
2. Window dialogue closes

Exceptional Flow

N/A

Termination

The item in the basket has been successfully edited and replaced with new values

Post Condition

Success Condition:

1. Item has been edited
2. Item successfully added to basket
3. Item has replaced prior item

5.<Removing Items from Basket>

Description & Priority

Use Case

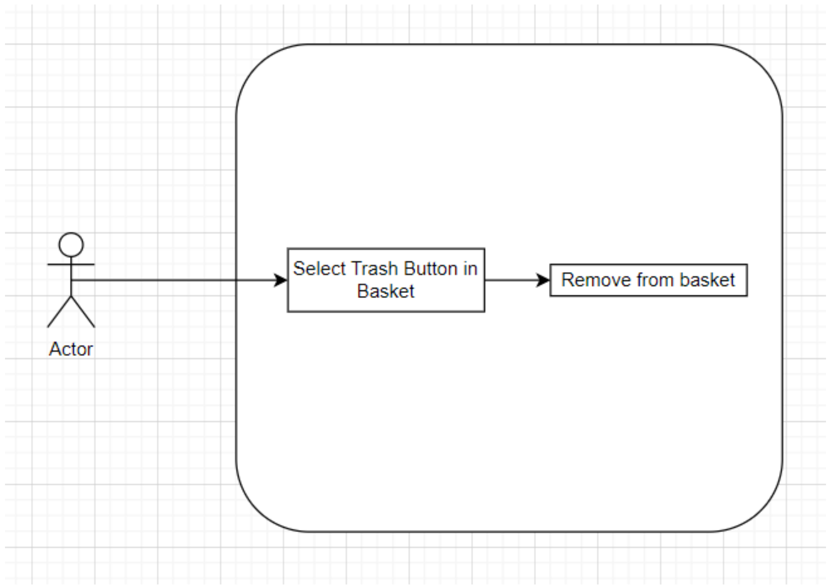
Scope

The user would like to remove an item from their current Order list

Description

The use case describes what a user needs to do if they want to remove an item from the basket

Diagram



Flow Description

Precondition

The user has items added in their basket on the application

Activation

The user has items in their basket

Main Flow

1. The user clicks on the Trash icon beside an item in their basket
2. Basket updates to remove item

Alternate Flow

N/A

Exceptional Flow

N/A

Termination

Basket is updated to have item removed

Post Condition

Success Condition:

1. The item in the basket has been removed

6.<Confirming Order>

Description & Priority

Use Case

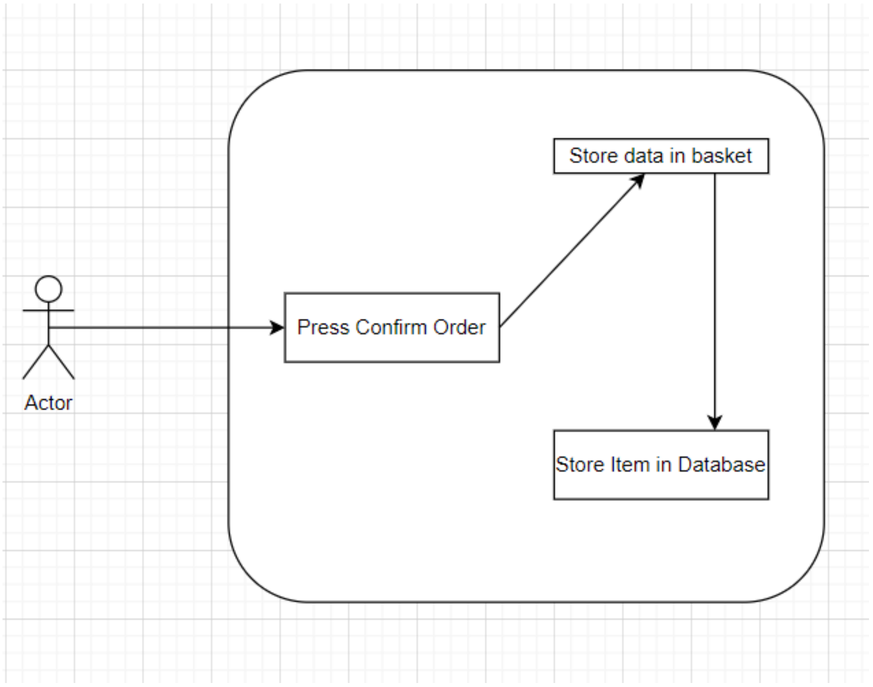
Scope

The user has items in their basket and has decided to check out and confirm their order.

Description

The use case describes the user selecting Confirm Order once they have logged in and.

Diagram



Flow Description

Precondition

The user has items added in their basket on the application and are logged in

Activation

The user is logged in and has items in the current basket

Main Flow

1. The user clicks on Confirm Button
2. Application sends data to Database
3. Database receives data and stores it
4. Data is now saved in database

Alternate Flow

N/A

Exceptional Flow

E1: <Connection to Database has been declined>

1. Order has failed to send to database

Termination

User has confirmed their order and it has sent it to the database

Post Condition

Success Condition:

1. Basket has been saved to database

2.1.1 Functional requirements

This section lists the functional requirements describe the possible effects that can occur within the system i.e., what the system must accomplish.

2.1.2 Non-Functional requirements

Availability:

The application must have a 100% uptime since it is a web application. The user must be able to access the service from anywhere and on any device. Firebase hosting handles this requirement.

Recover:

Users who have forgotten their Gmail account are able to reset their password with the help of Google API since it is already enabled on the website.

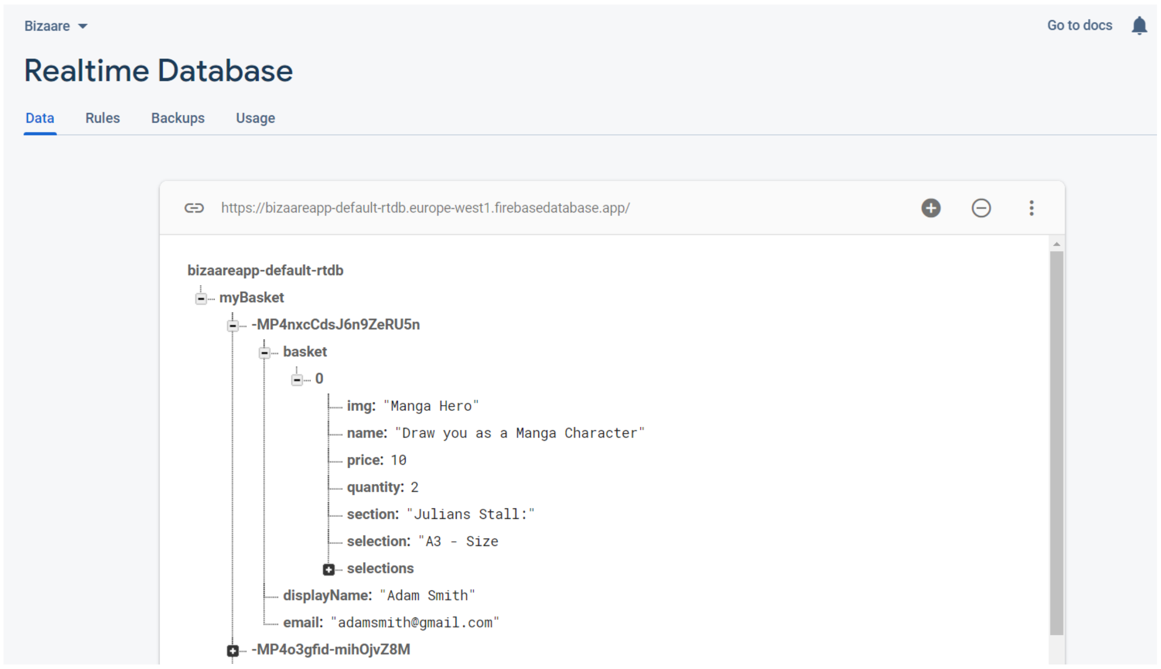
Security:

The majority of security is handled by Google when users are signing in with their email account.

2.1.3 Data requirements

This section explains the data requirements in the application, which is essential to its features.

For storing data, I will be using Google Firebase Realtime Databases. Google provides two different types of database in their console: Firestore and Realtime Database. Firebase Firestore uses a non-relational structure while Firebase Realtime Database uses an SQL relational style structure with tables. Below is the table for a single basket order from the user Adam Smith and what that person has ordered.


Bizaare
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://bizaareapp-default-rtdb.europe-west1.firebasedatabase.app/

```
bizaareapp-default-rtdb
  myBasket
    -MP4nxcCdsJ6n9ZeRU5n
      basket
        0
          img: "Manga Hero"
          name: "Draw you as a Manga Character"
          price: 10
          quantity: 2
          section: "Julians Stall:"
          selection: "A3 - Size
          selections
      displayName: "Adam Smith"
      email: "adamsmith@gmail.com"
    -MP4o3gfid-mihOjvZ8M
```

2.1.4 User requirements

To be able to successfully access the website the user must have the following requirements:

- Internet Access: This an online web application, therefore the user must have access to the internet if they wish to access it.
- Device with Internet Access: This can range from a phone to a laptop or pc that has a browser where they can enter a URL to access the web application.

2.1.5 Environmental requirements

When developing the application, I had access to the following:

- Internet Access: Deploy or Host the web application.
- IDE: I used Visual Studio Code to edit and do all my coding.
- PC: I used a personal computer with internet access to run my programs and software.
- Google Firebase: A google account is also required in order to access the firebase tools.

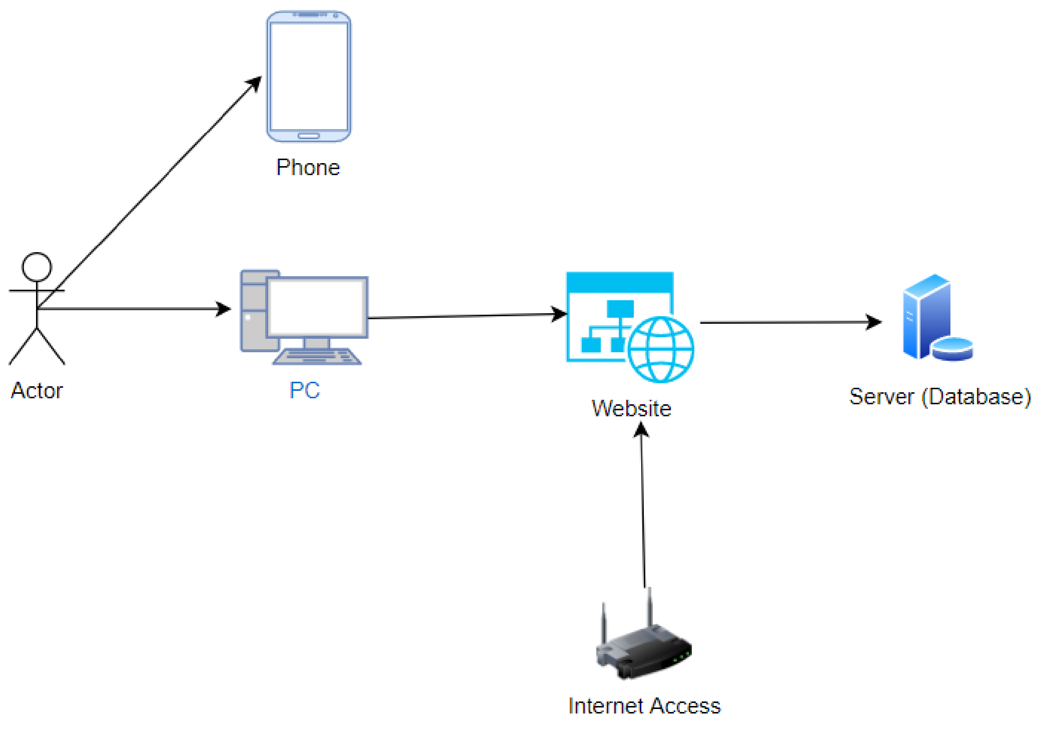
2.1.6 Usability requirements

To access and use the web application the user must complete the following criteria.

- User has a valid existing Gmail account.
- They have access to internet via Wi-Fi, Ethernet, or Mobile Data.
- A device with a working browser.

2.2 Design and Architecture

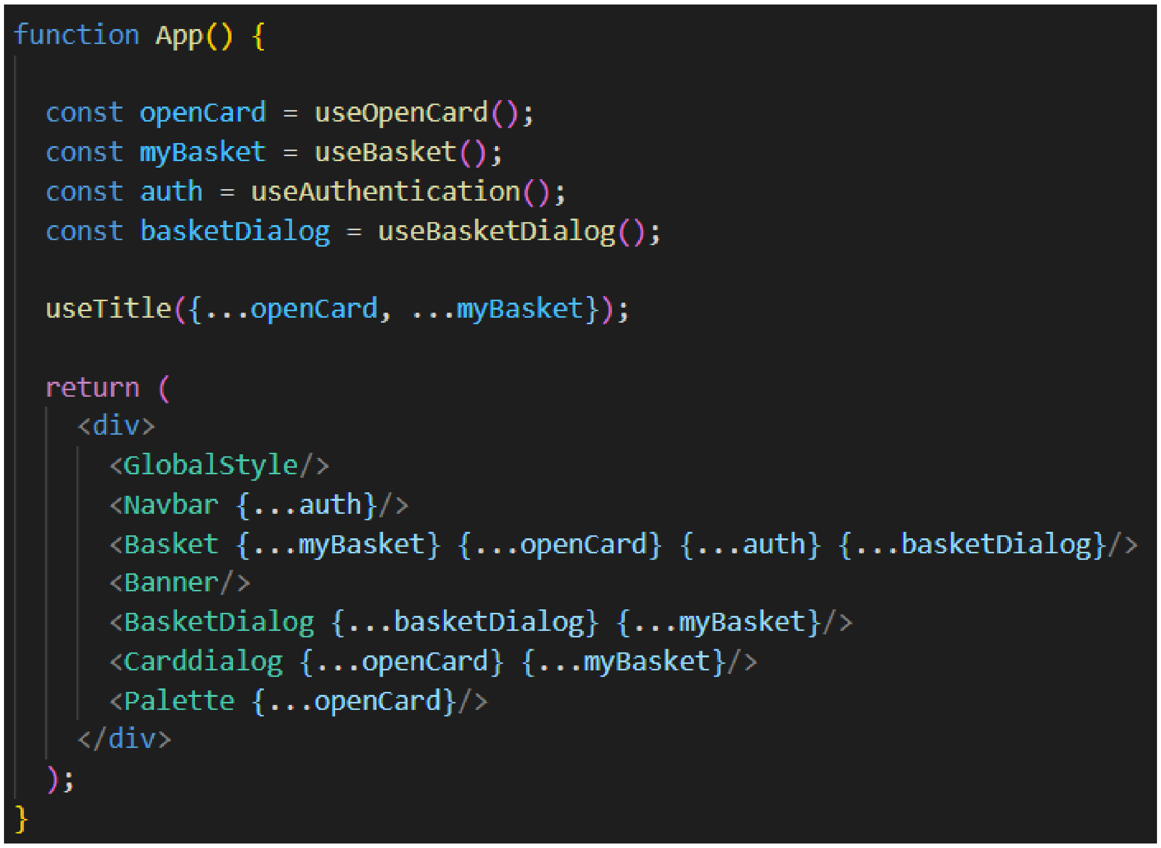
The following diagram describes the systems architecture. The user is using a device be it a phone or laptop that is connected to the internet. The web application is accessed through the devices browser and then communicates with the server to send data to the database.



2.3 Implementation

To begin implementation, I will first briefly explain how the coding structure for React works. The package uses a JSX format where HTML and JavaScript run side by side in the same document by using functions and constants. These functions carry out JavaScript style functionality while constants can hold information such as text and CSS style formatting.

As an example, this is the App.js file where the homepage of the site lives



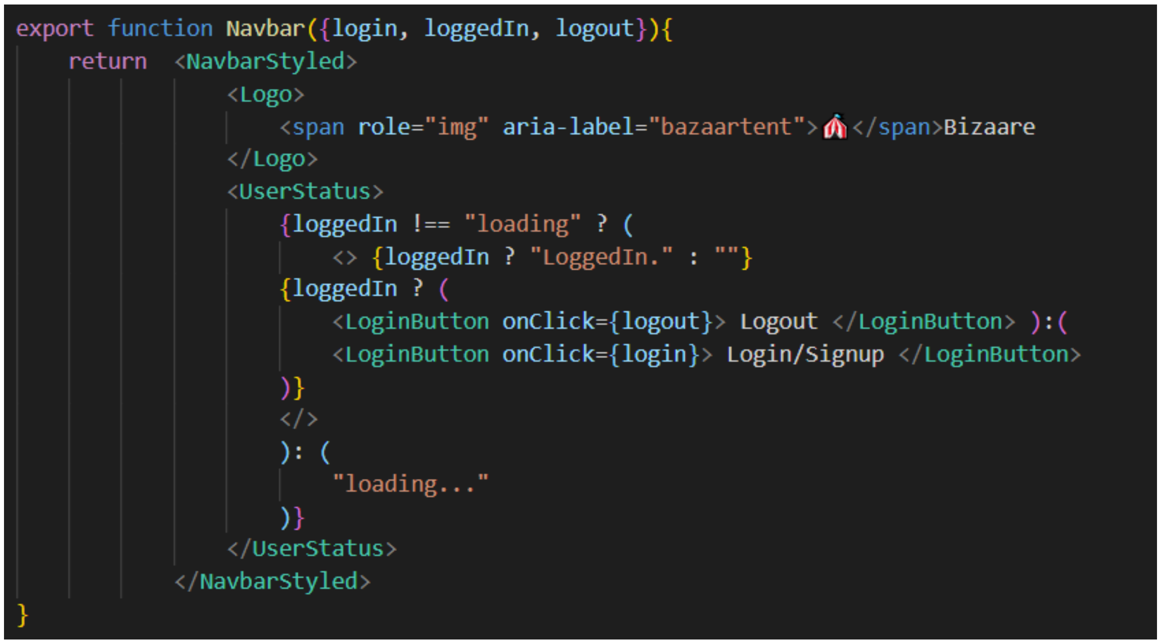
```
function App() {

  const openCard = useOpenCard();
  const myBasket = useBasket();
  const auth = useAuthentication();
  const basketDialog = useBasketDialog();

  useTitle({...openCard, ...myBasket});

  return (
    <div>
      <GlobalStyle/>
      <Navbar {...auth}/>
      <Basket {...myBasket} {...openCard} {...auth} {...basketDialog}/>
      <Banner/>
      <BasketDialog {...basketDialog} {...myBasket}/>
      <Carddialog {...openCard} {...myBasket}/>
      <Palette {...openCard}/>
    </div>
  );
}
```

In the return code, we are getting all the data pulled within the green tags and displayed within a div tag. These green tags are imported from different files in the folder at the top of the document in a directory. The <Navbar> tags purpose for example is to return all the values within this navbar:



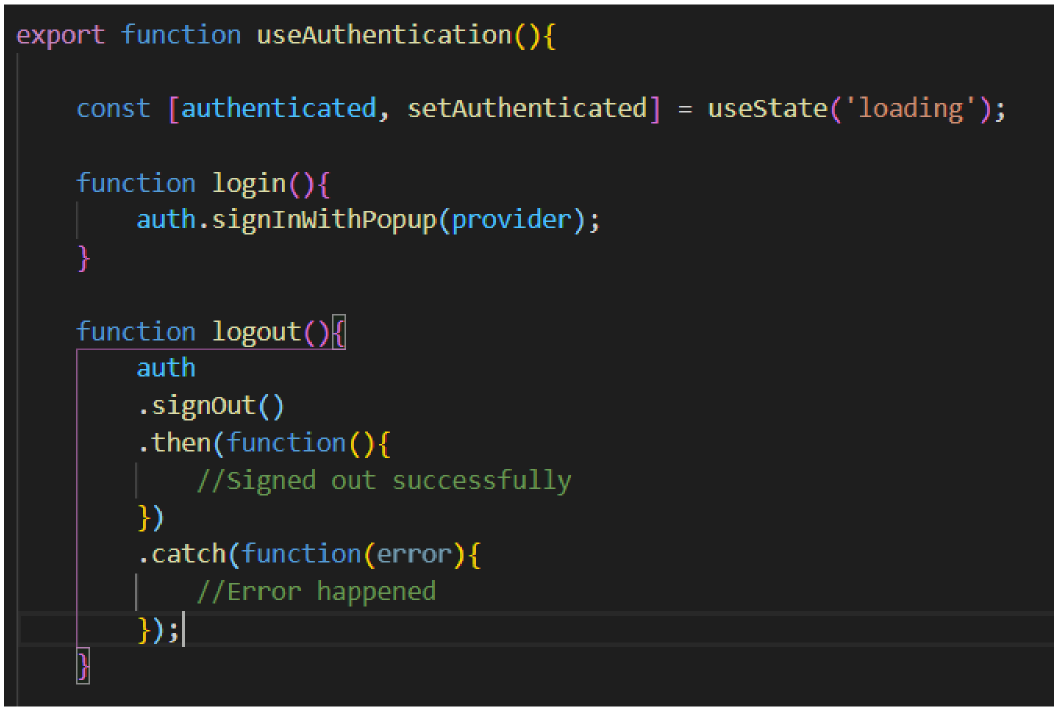
```
export function Navbar({login, loggedIn, logout}){
    return  <NavbarStyled>
                <Logo>
                    <span role="img" aria-label="bazaartent">🎪</span>Bizaare
                </Logo>
                <UserStatus>
                    {loggedIn !== "loading" ? (
                        <> {loggedIn ? "LoggedIn." : ""}
                    {loggedIn ? (
                        <LoginButton onClick={logout}> Logout </LoginButton> ):(
                        <LoginButton onClick={login}> Login/Signup </LoginButton>
                    )}
                    </>
                    ): (
                        "loading..."
                    )}
                </UserStatus>
            </NavbarStyled>
}
```

The export function lets us use the function in this document in another document, in this case the App.js document.

Hooks:

React hooks lets us always use functions instead of having to constantly switch between a function and a class and allows us to use useState functionalities within react.

Below is the snippit of a function for the authentication for firebase auth

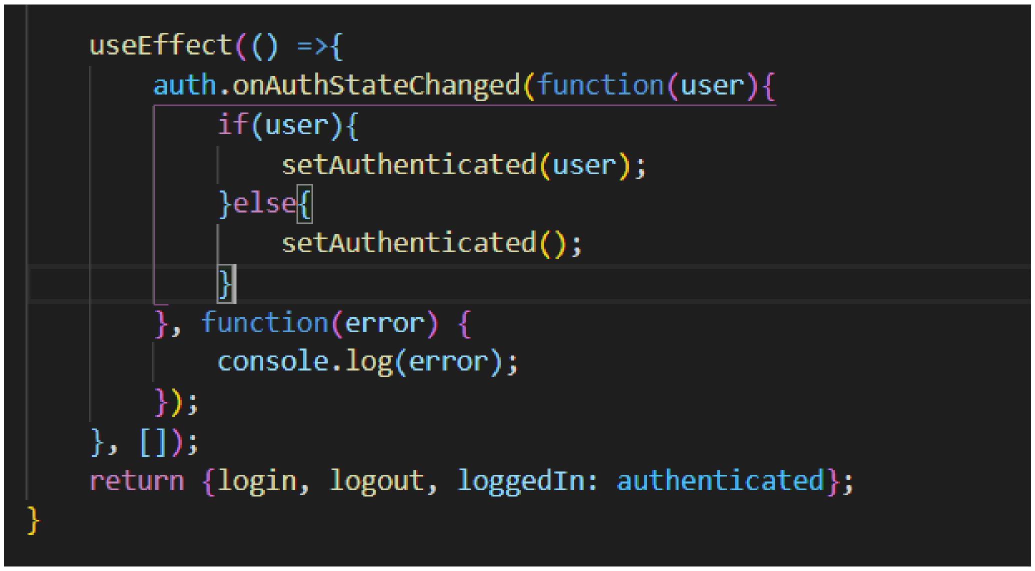


```
export function useAuthentication(){

    const [authenticated, setAuthenticated] = useState('loading');

    function login(){
        auth.signInWithPopup(provider);
    }

    function logout(){
        auth
        .signOut()
        .then(function(){
            //Signed out successfully
        })
        .catch(function(error){
            //Error happened
        });
    }
```

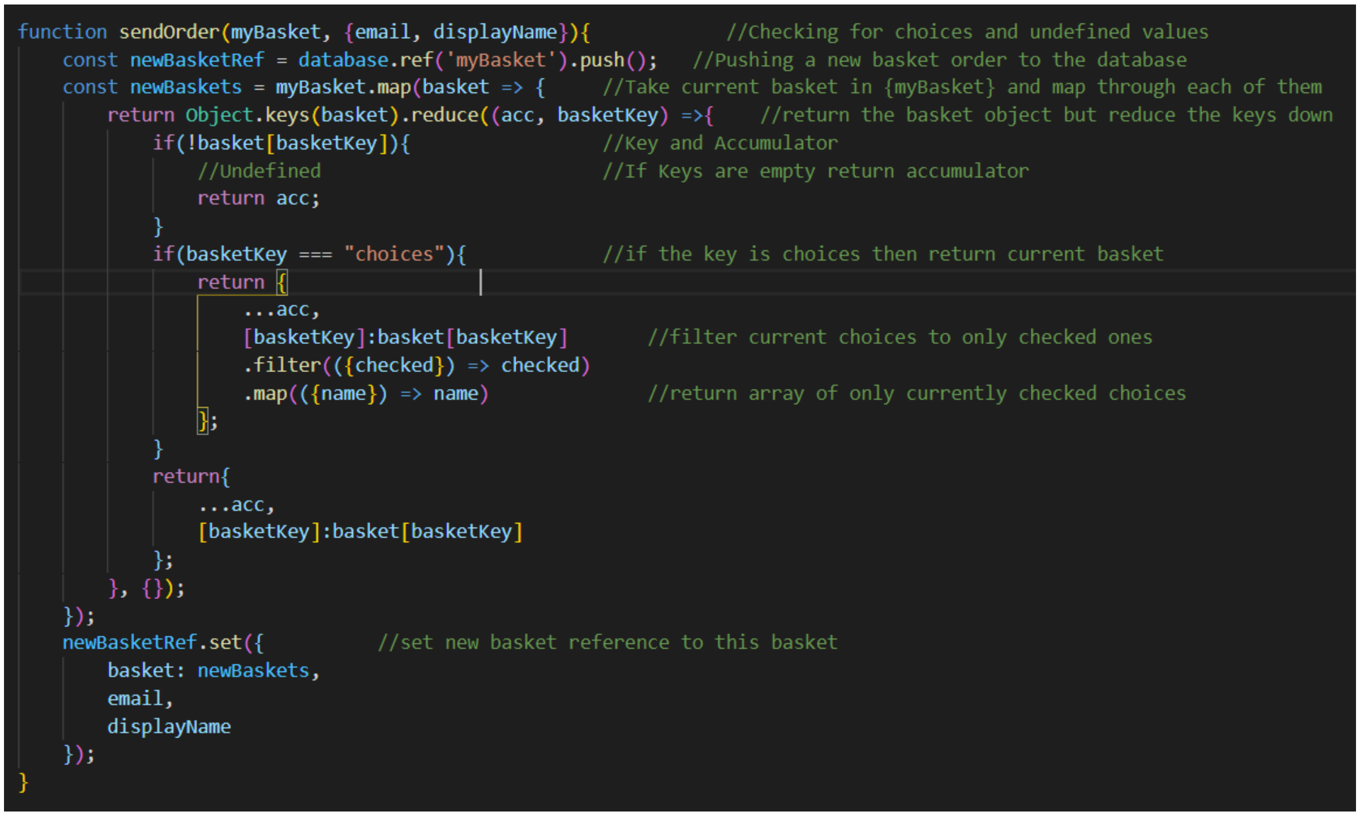


```
    useEffect(() =>{
        auth.onAuthStateChanged(function(user){
            if(user){
                setAuthenticated(user);
            }else{
                setAuthenticated();
            }
        }, function(error) {
            console.log(error);
        });
    }, []);
    return {login, logout, loggedIn: authenticated};
}
```

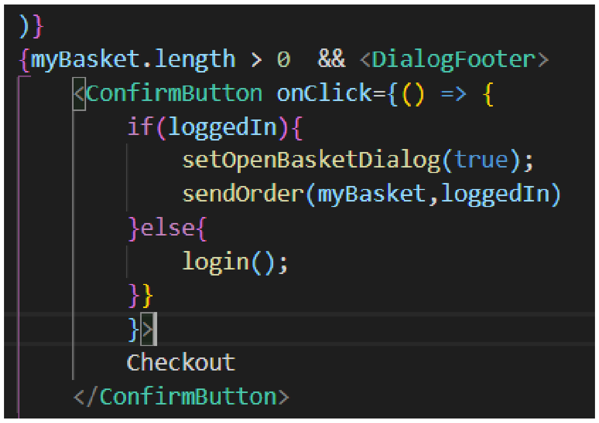
useEffect in this case states that once a user logs in ‘loading’ occurs and this just simply tells the user their authentication is being checked hence it is loading.

Sending a cleaned order to the database

The code is commented to best describe what is currently happening.

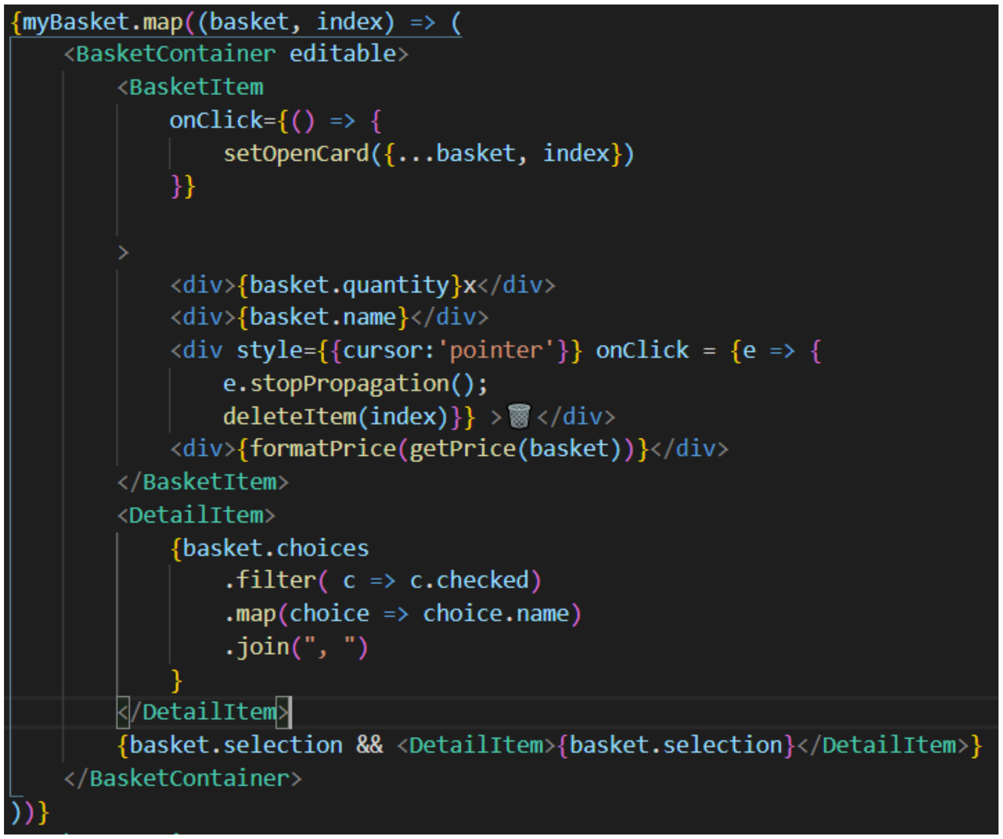

```
function sendOrder(myBasket, {email, displayName}){            //Checking for choices and undefined values
    const newBasketRef = database.ref('myBasket').push();   //Pushing a new basket order to the database
    const newBaskets = myBasket.map(basket => {      //Take current basket in {myBasket} and map through each of them
        return Object.keys(basket).reduce((acc, basketKey) =>{     //return the basket object but reduce the keys down
            if(!basket[basketKey]){                   //Key and Accumulator
                //Undefined                           //If Keys are empty return accumulator
                return acc;
            }
            if(basketKey === "choices"){              //if the key is choices then return current basket
                return {
                    ...acc,
                    [basketKey]:basket[basketKey]       //filter current choices to only checked ones
                    .filter(({checked}) => checked)
                    .map(({name}) => name)              //return array of only currently checked choices
                };
            }
            return{
                ...acc,
                [basketKey]:basket[basketKey]
            };
        }, {});
    });
    newBasketRef.set({          //set new basket reference to this basket
        basket: newBaskets,
        email,
        displayName
    });
}
```

This function is essentially a form of validation to remove extra data that could be sent to the database and removed before it reaches the end point.


```
)}
{myBasket.length > 0  && <DialogFooter>
    <ConfirmButton onClick={() => {
        if(loggedIn){
            setOpenBasketDialog(true);
            sendOrder(myBasket,loggedIn)
        }else{
            login();
        }}
        }>
        Checkout
    </ConfirmButton>
```

Here we can see that the confirm button executes the code above

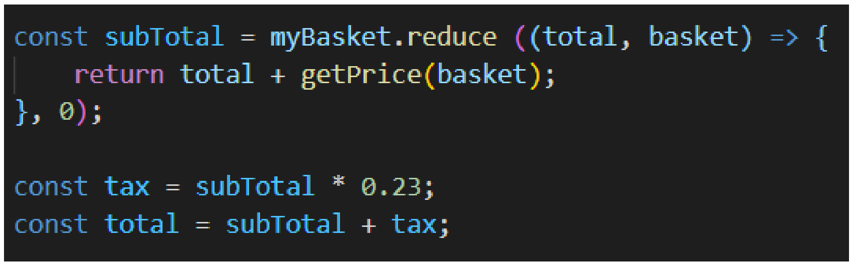
The code below outlines what happens when to display when a user successfully clicks on an item in a card and sends it to their basket. It displays the item along with its values, base price, tax, and full cost



```
{myBasket.map((basket, index) => (
    <BasketContainer editable>
        <BasketItem
            onClick={() => {
                setOpenCard({...basket, index})
            }}

        >
            <div>{basket.quantity}x</div>
            <div>{basket.name}</div>
            <div style={{cursor:'pointer'}} onClick = {e => {
                e.stopPropagation();
                deleteItem(index)}} >🗑</div>
            <div>{formatPrice(getPrice(basket))}</div>
        </BasketItem>
        <DetailItem>
            {basket.choices
                .filter( c => c.checked)
                .map(choice => choice.name)
                .join(", ")
            }
        </DetailItem>
        {basket.selection && <DetailItem>{basket.selection}</DetailItem>}
    </BasketContainer>
))}
```

This function calculates the tax from the base price that is the subtotal and sets it as the get price as the final price

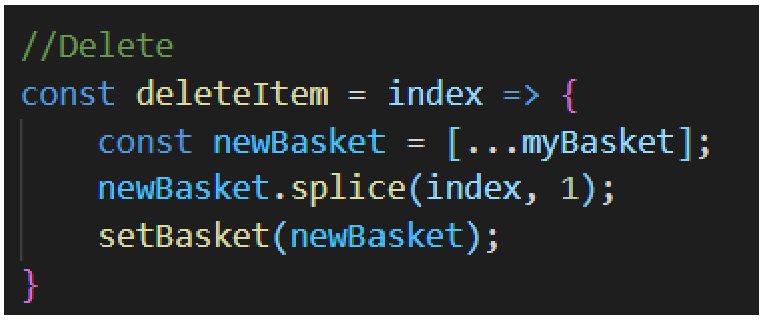


```
const subTotal = myBasket.reduce ((total, basket) => {
    return total + getPrice(basket);
}, 0);

const tax = subTotal * 0.23;
const total = subTotal + tax;
```

The editable function in the basket container opens up that saved card dialog and allows the user to change its values if they so choose.

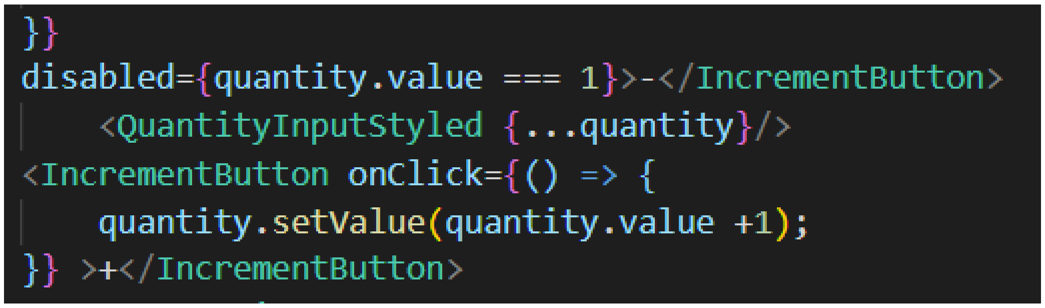
Deleting an item



```
//Delete
const deleteItem = index => {
    const newBasket = [...myBasket];
    newBasket.splice(index, 1);
    setBasket(newBasket);
}
```

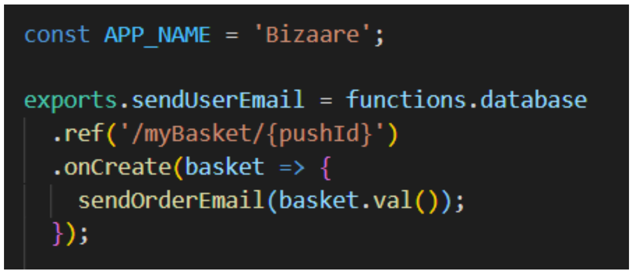
This function simple removes the current item from the index which is the list of all current items stored in the basket function.

Quantity input lets the user change the value how many of a given item they want to add to their basket



```
}}
disabled={quantity.value === 1}>-</IncrementButton>
    <QuantityInputStyled {...quantity}/>
<IncrementButton onClick={() => {
    quantity.setValue(quantity.value +1);
}} >+</IncrementButton>
```

Sending an email to users via Firebase functions



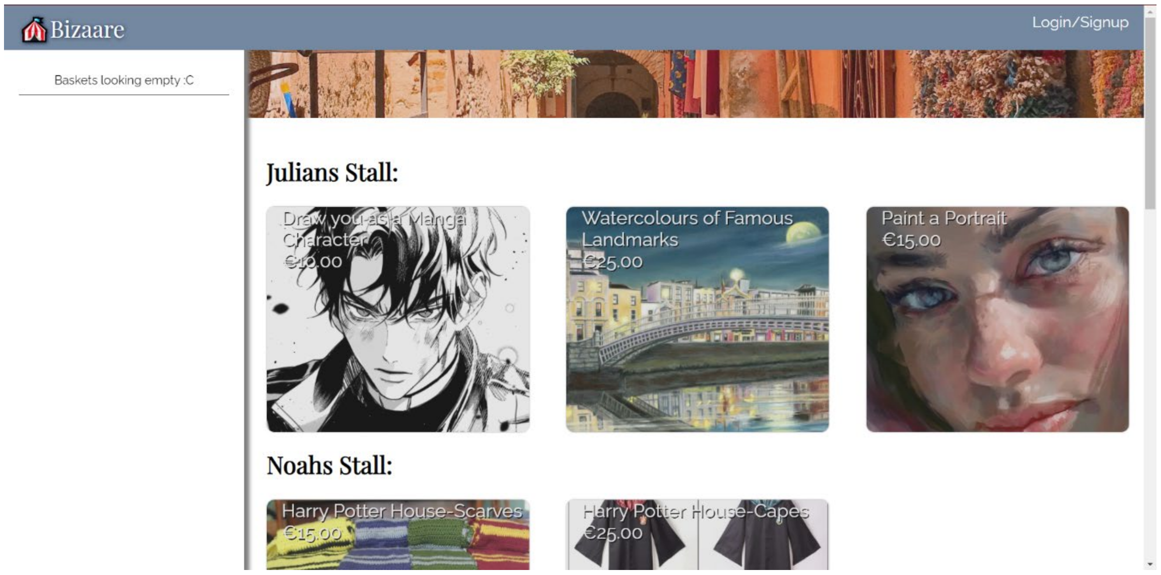
```
const APP_NAME = 'Bizaare';

exports.sendUserEmail = functions.database
  .ref('/myBasket/{pushId}')
  .onCreate(basket => {
    sendOrderEmail(basket.val());
  });
```

This function is exported to Firebase where it checks when a user who has registered successfully creates an item in the basket table. This carries out the send order email which sends an email to the user saved to the basket table. This function however is disabled as it does not send any data to the users email.

2.4 *Graphical User Interface (GUI) Layout*

Home Page GUI:



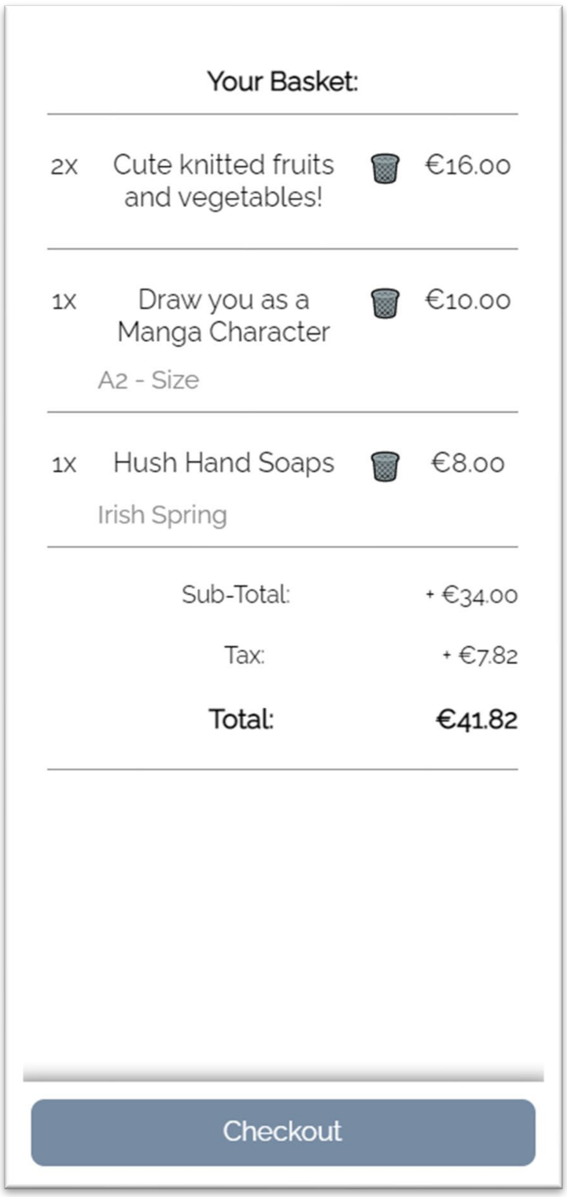
Navbar with a Login/Signup at the top right

User can scroll down to reveal more stalls, each stall has cards that the user can select if they want to add them to their basket

Basket GUI:

Two States for the basket GUI

Baskets looking empty :C

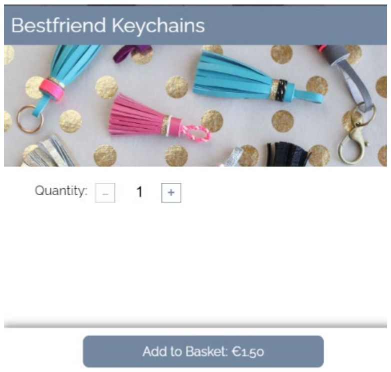
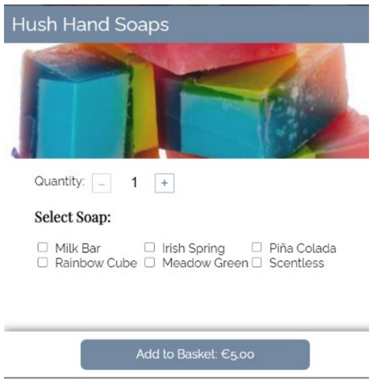
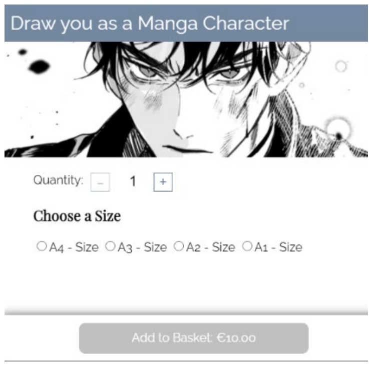


Your Basket:

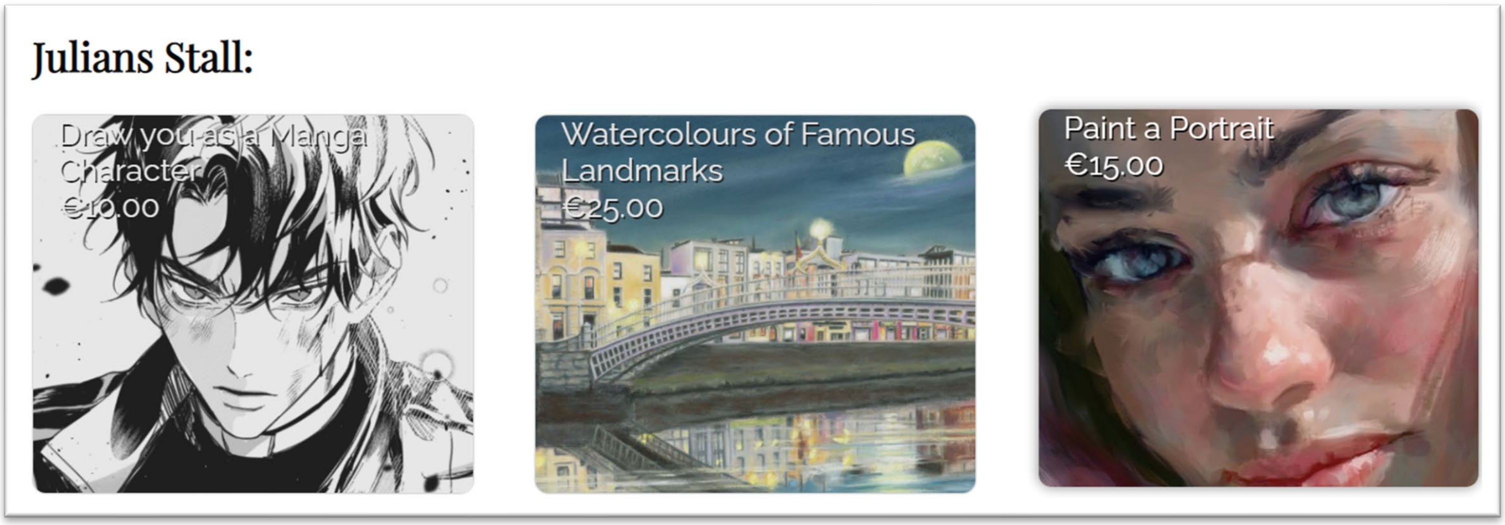
2x	Cute knitted fruits and vegetables!	€16.00
1x	Draw you as a Manga Character A2 - Size	€10.00
1x	Hush Hand Soaps Irish Spring	€8.00

Sub-Total:	+ €34.00
Tax:	+ €7.82
Total:	**€41.82**

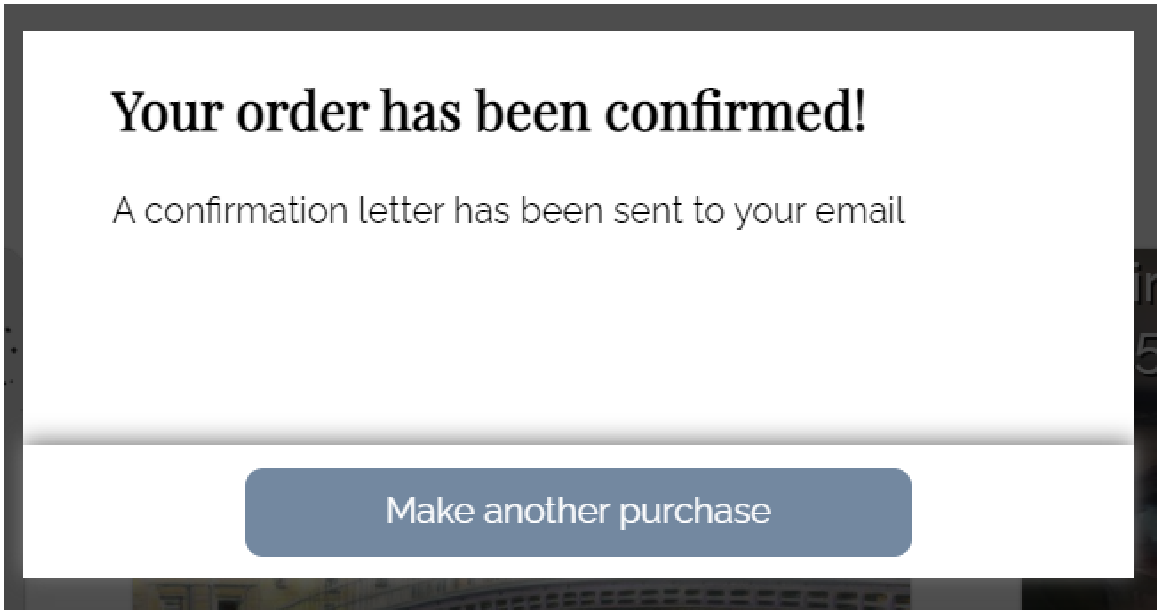
Item Card GUI:

	 Hush Hand Soaps Quantity: 1 Select Soap: Milk Bar Irish Spring Piña Colada Rainbow Cube Meadow Green Scentless Add to Basket: €5.00	 Draw you as a Manga Character Quantity: 1 Choose a Size A4 - Size A3 - Size A2 - Size A1 - Size Add to Basket: €10.00
Basic Card Interface Single Quantity Option	User can select multiple options	Single option selection, user is forced to choose, and button is disabled

Stall GUI:


Julians Stall:
Draw you as a Manga Character
€10.00
Watercolours of Famous Landmarks
€25.00
Paint a Portrait
€15.00

Confirmation purchase window GUI



Your order has been confirmed!

A confirmation letter has been sent to your email

2.5 Testing

Application testing

This is testing the website to check all the functions available are working properly

#	Requirements	Tested	Passed	Comments
1	Signup/Login	✓	✓	User cookies are saved to browser
2	Logout	✓	✓	User logs out and is unable to checkout
3	Access Card	✓	✓	User can access Card and Item values
4	Add Item	✓	✓	User can add item to basket
5	Edit Item	✓	✓	User can edit item in basket
6	Delete Item	✓	✓	User can delete item from basket
7	Checkout	✓	✓	User can checkout and clear basket

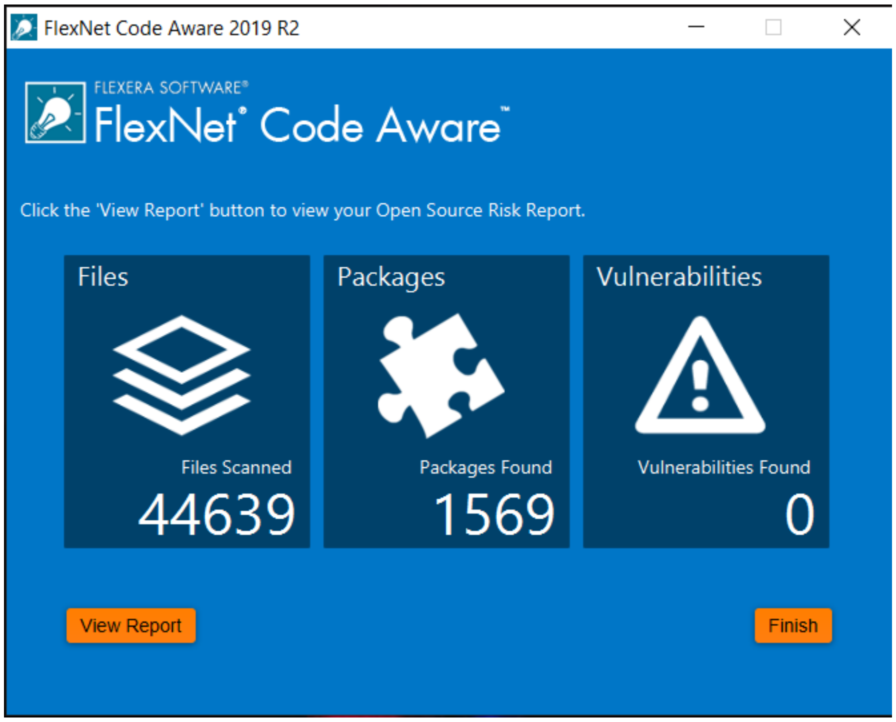
Website Testing

#	*Requirements*	*Tested*	*Passed*	*Comments*
1	Hosted	✓	✓	Website is live at https://bizaareapp.web.app
2	Site Access	✓	✓	Site becomes accessible only with internet connection
3	Functions	✓	X	Error with running script on email command, user is not sent their confirmation email

2.6 *Further Security Testing*

Security Testing is carried out on the application to check the code and search for any vulnerabilities. To begin, I checked the source folder using FlexNet Code Aware software to scan for vulnerabilities.

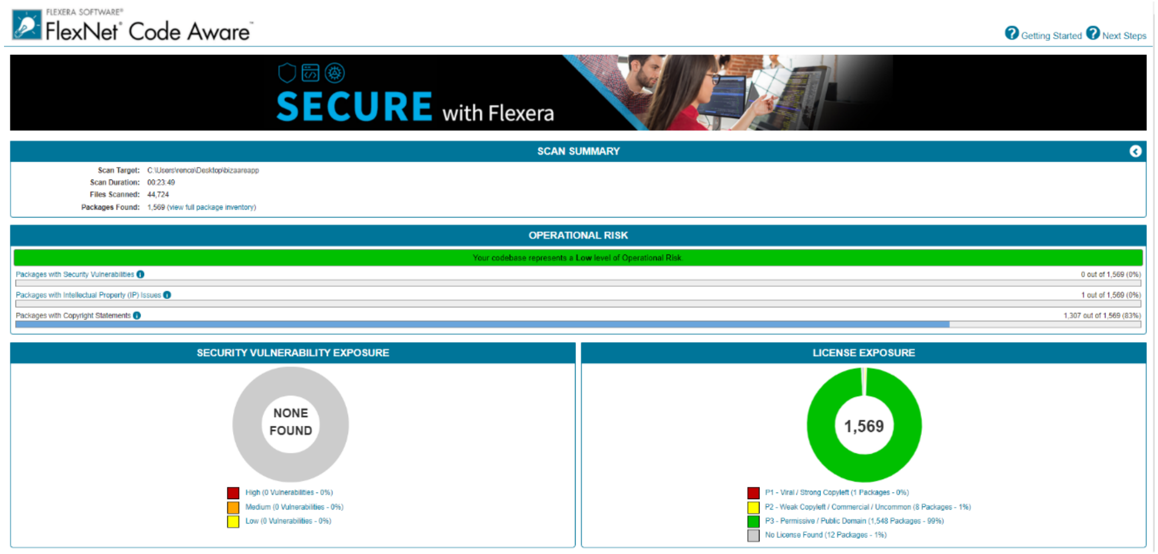
Here are the results from scanning 44639 files and 1569 packages:



Click the 'View Report' button to view your Open Source Risk Report.

Files Scanned	Packages Found	Vulnerabilities Found
44639	1569	0

Viewing the report displays it to a browser window further explaining the results



SCAN SUMMARY

Scan Target:	C:\Users\rence\Desktop\bizaareapp
Scan Duration:	00:23:49
Files Scanned:	44,724
Packages Found:	1,569 (view full package inventory)

OPERATIONAL RISK

Your codebase represents a **Low** level of Operational Risk

Packages with Security Vulnerabilities	0 out of 1,569 (0%)
Packages with Intellectual Property (IP) Issues	1 out of 1,569 (0%)
Packages with Copyright Statements	1,307 out of 1,569 (83%)

SECURITY VULNERABILITY EXPOSURE

- 🟥 High (0 Vulnerabilities - 0%)
- 🟧 Medium (0 Vulnerabilities - 0%)
- 🟨 Low (0 Vulnerabilities - 0%)

LICENSE EXPOSURE

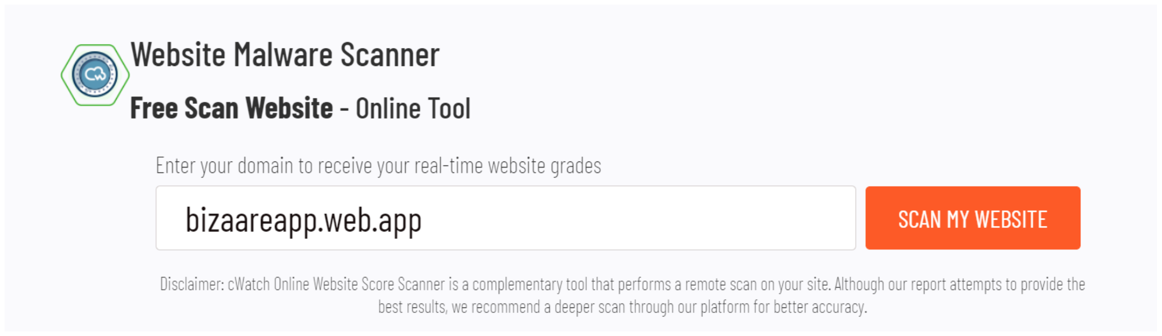
- 🟥 P1 - Viral / Strong Copyleft (1 Packages - 0%)
- 🟨 P2 - Weak Copyleft / Commercial / Uncommon (8 Packages - 1%)
- 🟩 P3 - Permissive / Public Domain (1,548 Packages - 99%)
- ⬜ No License Found (12 Packages - 1%)

Web Hosting Testing:

To test the live hosted website, I used two programs to check its security

cWatch – Website Scanner

Below are the results from scanning my website



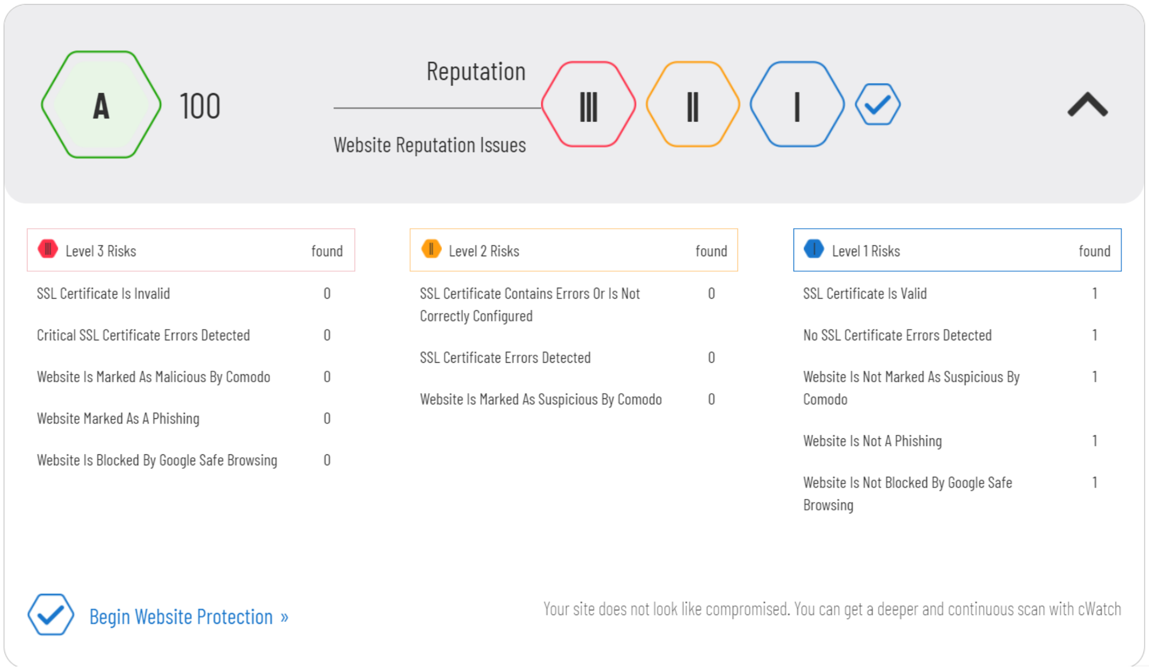
Website Malware Scanner

Free Scan Website - Online Tool

Enter your domain to receive your real-time website grades

bizaareapp.web.app

Disclaimer: cWatch Online Website Score Scanner is a complementary tool that performs a remote scan on your site. Although our report attempts to provide the best results, we recommend a deeper scan through our platform for better accuracy.



A 100

Reputation

Website Reputation Issues

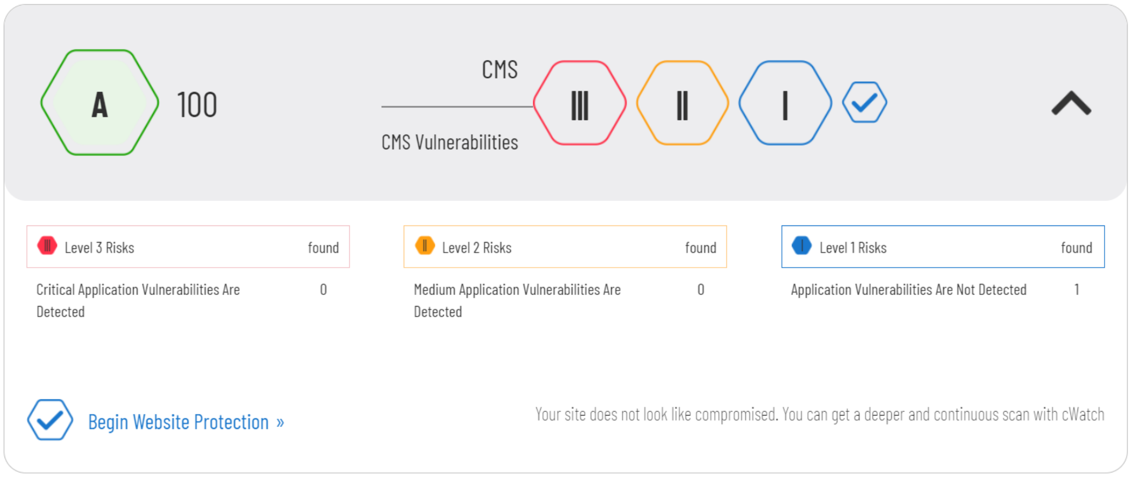
Level 3 Risks	found
SSL Certificate Is Invalid	0
Critical SSL Certificate Errors Detected	0
Website Is Marked As Malicious By Comodo	0
Website Marked As A Phishing	0
Website Is Blocked By Google Safe Browsing	0

Level 2 Risks	found
SSL Certificate Contains Errors Or Is Not Correctly Configured	0
SSL Certificate Errors Detected	0
Website Is Marked As Suspicious By Comodo	0

Level 1 Risks	found
SSL Certificate Is Valid	1
No SSL Certificate Errors Detected	1
Website Is Not Marked As Suspicious By Comodo	1
Website Is Not A Phishing	1
Website Is Not Blocked By Google Safe Browsing	1

Begin Website Protection »

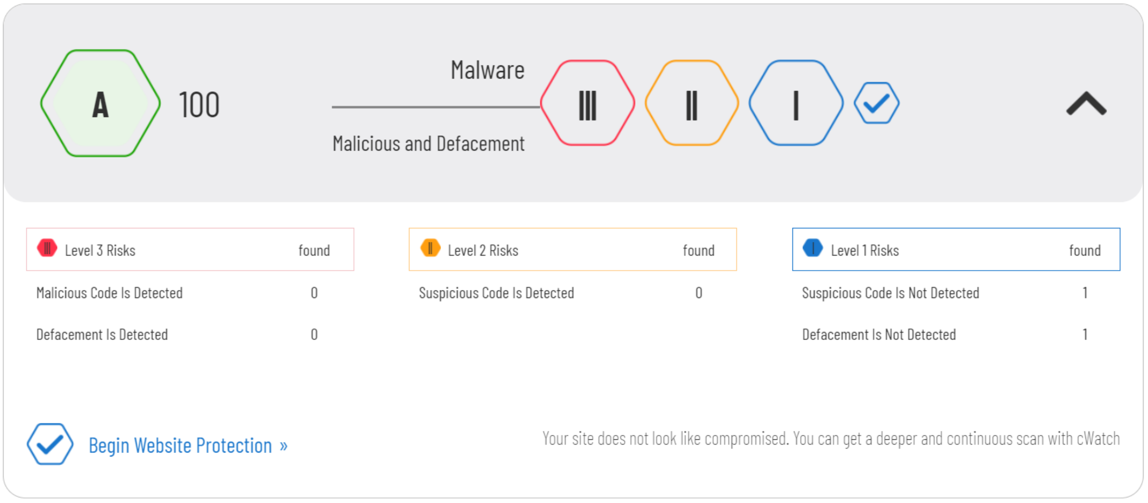
Your site does not look like compromised. You can get a deeper and continuous scan with cWatch



Level 3 Risks	found	Level 2 Risks	found	Level 1 Risks	found
Critical Application Vulnerabilities Are Detected	0	Medium Application Vulnerabilities Are Detected	0	Application Vulnerabilities Are Not Detected	1

Begin Website Protection »

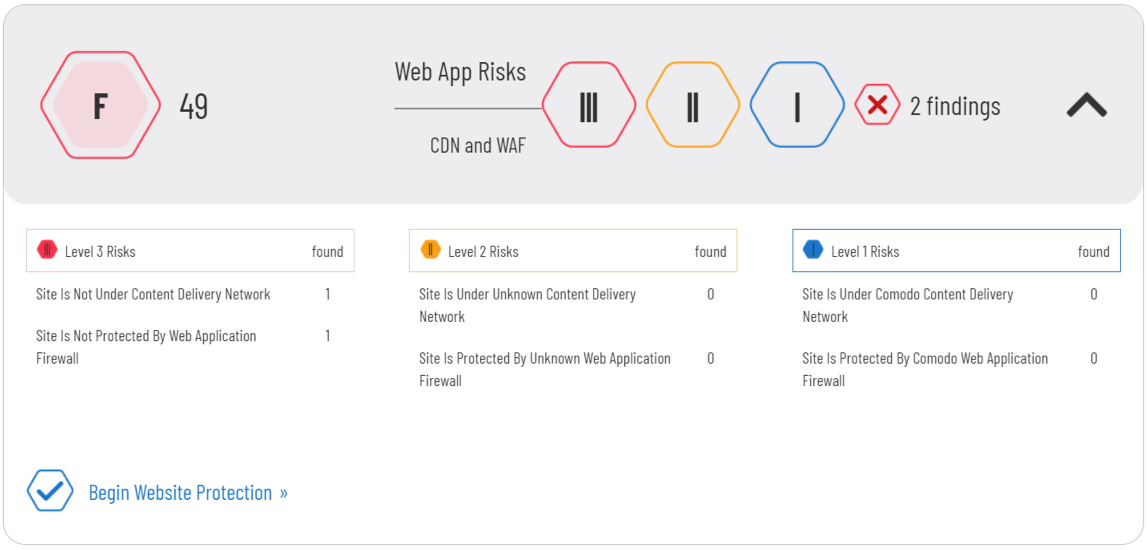
Your site does not look like compromised. You can get a deeper and continuous scan with cWatch



Level 3 Risks	found	Level 2 Risks	found	Level 1 Risks	found
Malicious Code Is Detected	0	Suspicious Code Is Detected	0	Suspicious Code Is Not Detected	1
Defacement Is Detected	0			Defacement Is Not Detected	1

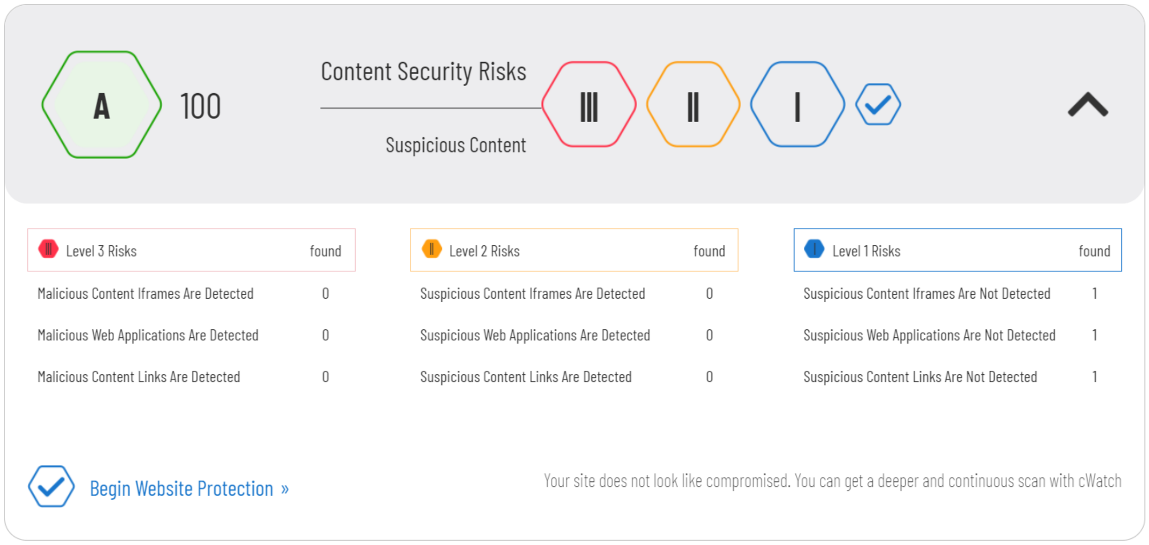
Begin Website Protection »

Your site does not look like compromised. You can get a deeper and continuous scan with cWatch



Level 3 Risks	found	Level 2 Risks	found	Level 1 Risks	found
Site Is Not Under Content Delivery Network	1	Site Is Under Unknown Content Delivery Network	0	Site Is Under Comodo Content Delivery Network	0
Site Is Not Protected By Web Application Firewall	1	Site Is Protected By Unknown Web Application Firewall	0	Site Is Protected By Comodo Web Application Firewall	0

Begin Website Protection »



Content Security Risks

Suspicious Content

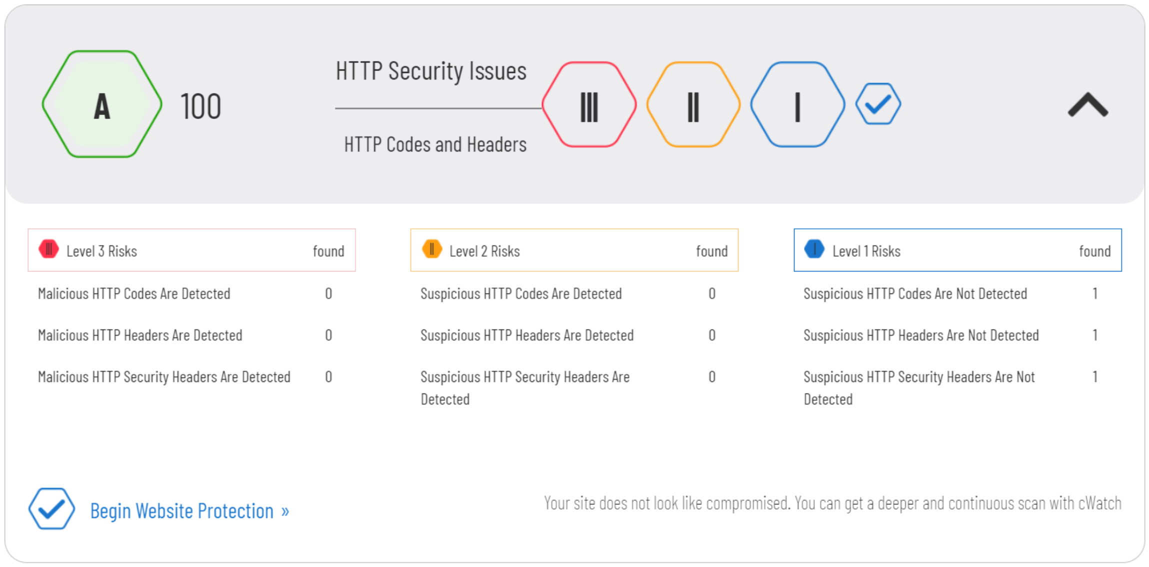
Level 3 Risks	found
Malicious Content Iframes Are Detected	0
Malicious Web Applications Are Detected	0
Malicious Content Links Are Detected	0

Level 2 Risks	found
Suspicious Content Iframes Are Detected	0
Suspicious Web Applications Are Detected	0
Suspicious Content Links Are Detected	0

Level 1 Risks	found
Suspicious Content Iframes Are Not Detected	1
Suspicious Web Applications Are Not Detected	1
Suspicious Content Links Are Not Detected	1

Begin Website Protection »

Your site does not look like compromised. You can get a deeper and continuous scan with cWatch



HTTP Security Issues

HTTP Codes and Headers

Level 3 Risks	found
Malicious HTTP Codes Are Detected	0
Malicious HTTP Headers Are Detected	0
Malicious HTTP Security Headers Are Detected	0

Level 2 Risks	found
Suspicious HTTP Codes Are Detected	0
Suspicious HTTP Headers Are Detected	0
Suspicious HTTP Security Headers Are Detected	0

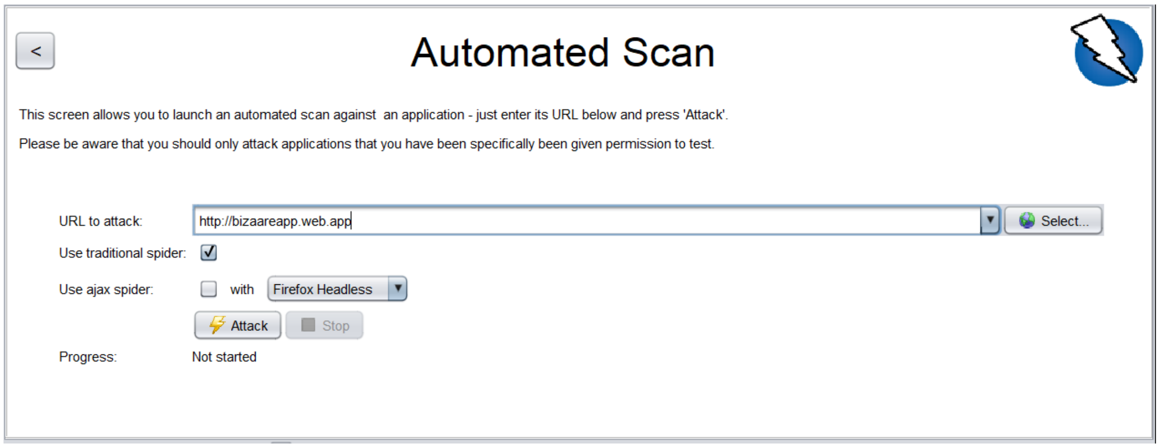
Level 1 Risks	found
Suspicious HTTP Codes Are Not Detected	1
Suspicious HTTP Headers Are Not Detected	1
Suspicious HTTP Security Headers Are Not Detected	1

Begin Website Protection »

Your site does not look like compromised. You can get a deeper and continuous scan with cWatch

The second program I used to scan my website is OWASP Zap

Here are the results from that scan:



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

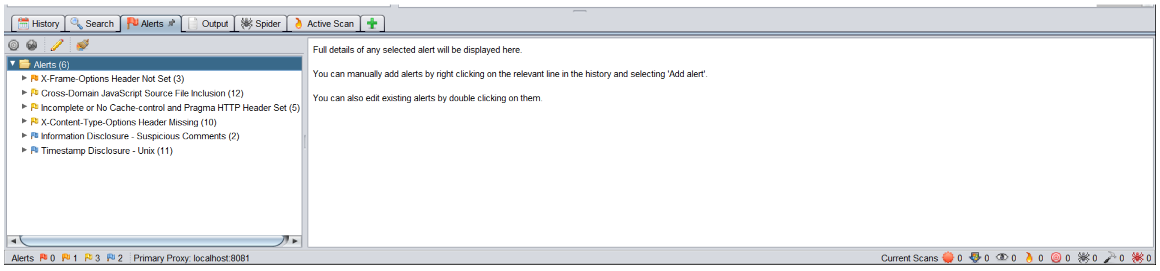
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://bizaareapp.web.app

Use traditional spider: ✓

Use ajax spider: ☐ with Firefox Headless

Progress: Not started



Alerts (6)
- X-Frame-Options Header Not Set (3)
- Cross-Domain JavaScript Source File Inclusion (12)
- Incomplete or No Cache-control and Pragma HTTP Header Set (5)
- X-Content-Type-Options Header Missing (10)
- Information Disclosure - Suspicious Comments (2)
- Timestamp Disclosure - Unix (11)

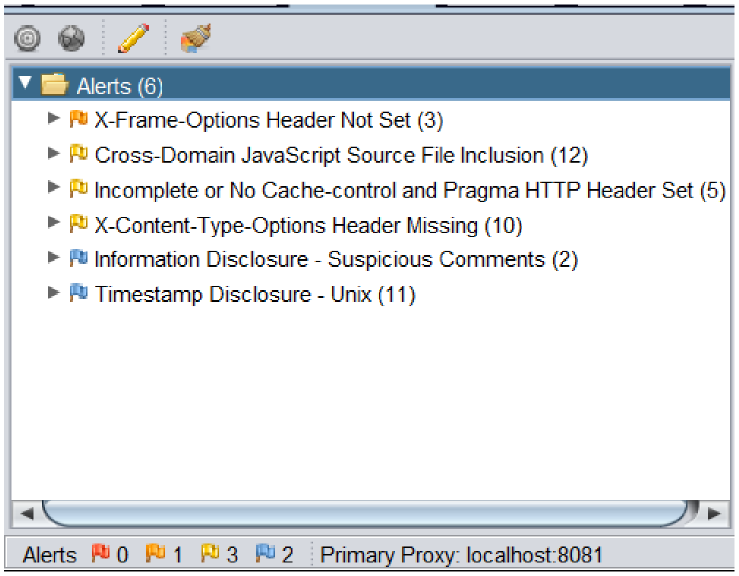
Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 0 1 3 2 Primary Proxy: localhost:8081

Current Scans 0 0 0 0 0 0 0 0



Alerts (6)
- X-Frame-Options Header Not Set (3)
- Cross-Domain JavaScript Source File Inclusion (12)
- Incomplete or No Cache-control and Pragma HTTP Header Set (5)
- X-Content-Type-Options Header Missing (10)
- Information Disclosure - Suspicious Comments (2)
- Timestamp Disclosure - Unix (11)

Alerts 0 1 3 2 Primary Proxy: localhost:8081

2.7 Evaluation

Functionality Evaluation:

From functionality standpoint the application has successful and working CRUD functionalities. It is evident however that more thought must be put into user authentication functionality as I am using Google Mail as a 3rd party authenticator. Database connection was a success and users can create baskets and send this data straight into the real-time database in Firebase. Unfortunately, there are issues with sending this data out in the form of a confirmation email as Google Firebase Functions refuses to connect to the application causing the functionality to become redundant.

Code Evaluation:

Revising the code took some time and I made sure to check thoroughly for any issues or bugs inside the code, especially front-end code that could affect the end products user interface. The validation has also been revised to reduce the chances of any injection attacks that could occur such as rechanging the prices for each item.

Website Evaluation:

This mostly comes down to how the overall site feels for the end user once it is visible as an online website. I interviewed some colleagues on their experience when using the site and their feedback averaged at a good experience. The main issue was the site being accessed on smaller devices where the user interface is not friendly to small screen, most notably the basket window taking up a sizeable amount of real estate on the screen.

3 Conclusions

To conclude, the Bizaare application has served its purpose to offer an E-commerce platform for users. Purchases can be made and pushed towards the database and stored there on the platform. Functionality wise there is still much more missing that could be added and fixed in the application.

User Profiles proved to be a much greater feat to accomplish as my knowledge in using the software showed to be inadequate and I fell short of my achieving margin. Nonetheless, the application is live and users can access it from the url: bizaareapp.web.app thanks to Google Hosting.

4 Further development or research

Bizaare E-commerce can be taken to the next level if more resources and time were allocated to developing the web application itself. For instance, having users have their own individual accounts that display information about them and what they like to do, what they sell or what they have bought recently. A Wishlist for each user to see what they would want. A function that would let users create a stall and set values and prices. Allow users to create a social media platform within the site so they can communicate with each other. Paypal integration to allow actual purchases instead of mock data.

5 References

(React Documentation, 2020)

(Firebase Google Docs, 2020)

(Yarnpkg Docs, 2020)

(The beginners guide to react, 2020)

(Firebase-Ultimate Beginners Guide, 2020)

(React + Cloud Firestore: Step by step tutorial, 2020)

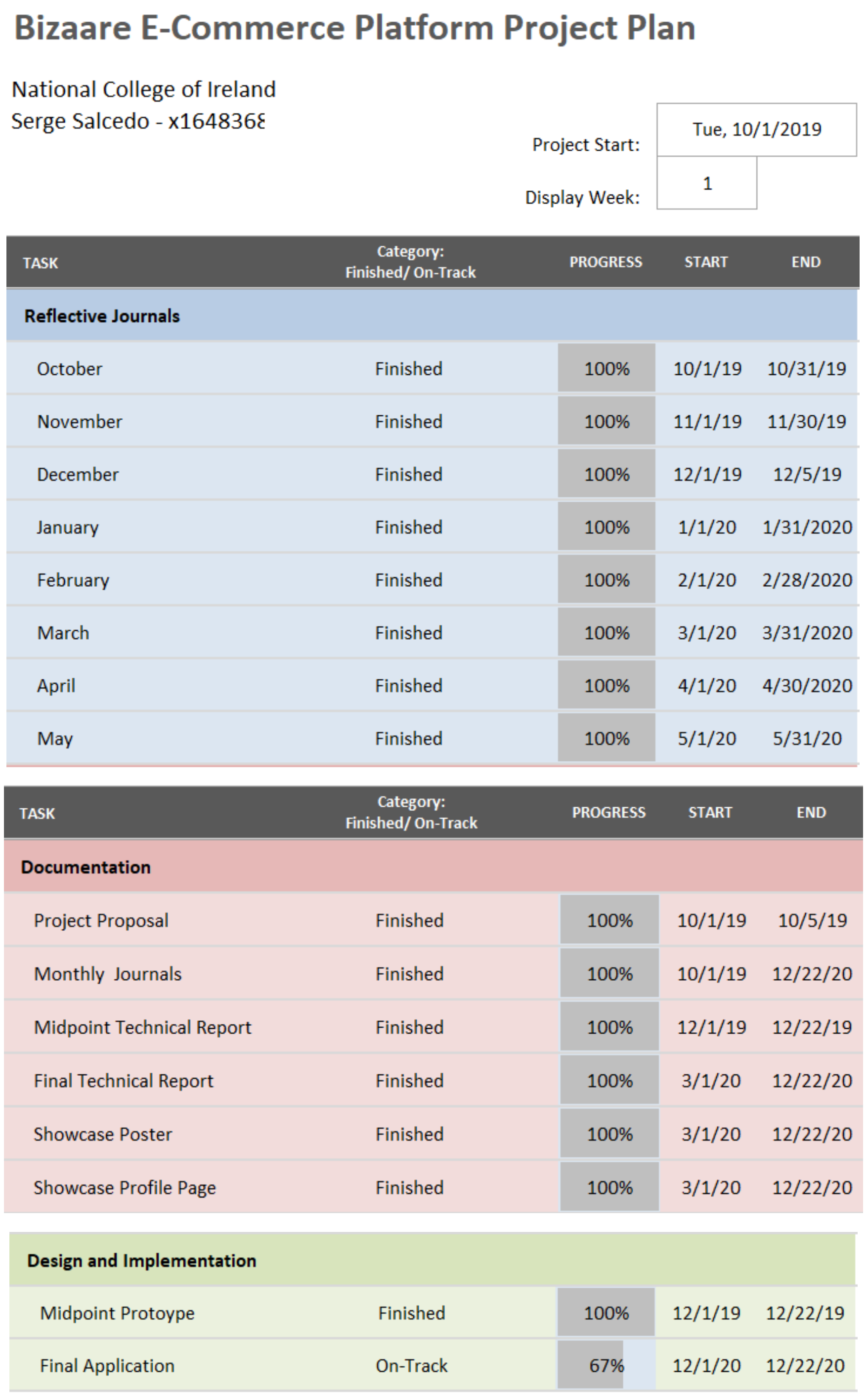
(Vergara, 2020)

(Full Stack React & Firebase Tutorial - Build a social media app, 2020)

Some references may have been lost, in which case the closest reference is set instead

6 Appendix

6.1 Project Plan



Bizaare E-Commerce Platform Project Plan

National College of Ireland
Serge Salcedo - x1648368

Project Start:	Tue, 10/1/2019
Display Week:	1

TASK	Category: Finished/ On-Track	PROGRESS	START	END
Reflective Journals				
October	Finished	100%	10/1/19	10/31/19
November	Finished	100%	11/1/19	11/30/19
December	Finished	100%	12/1/19	12/5/19
January	Finished	100%	1/1/20	1/31/2020
February	Finished	100%	2/1/20	2/28/2020
March	Finished	100%	3/1/20	3/31/2020
April	Finished	100%	4/1/20	4/30/2020
May	Finished	100%	5/1/20	5/31/20

TASK	Category: Finished/ On-Track	PROGRESS	START	END
Documentation				
Project Proposal	Finished	100%	10/1/19	10/5/19
Monthly Journals	Finished	100%	10/1/19	12/22/20
Midpoint Technical Report	Finished	100%	12/1/19	12/22/19
Final Technical Report	Finished	100%	3/1/20	12/22/20
Showcase Poster	Finished	100%	3/1/20	12/22/20
Showcase Profile Page	Finished	100%	3/1/20	12/22/20
Design and Implementation				
Midpoint Protoype	Finished	100%	12/1/19	12/22/19
Final Application	On-Track	67%	12/1/20	12/22/20

6.2 Monthly Journals / Meeting Minutes

October Journal:

My original idea for the software project was to create an image tracking and watermarking program that lets user hide strings within an image to prove its authenticity once it is scanned. I ran this idea through my supervisor Cristina, and she lets me know that my idea is basic and will just be a simple photo storage program. I was back to thinking of ideas and started to think of things that interested me. Finally, I settled on an idea of an online commercially available web application that lets others sell artwork. I let my supervisor know what she thought of it and said that it would do just fine and would be challenging enough for me to develop.

November Journal:

This month I finalized my project idea and went ahead with changing it from my first idea. I was initially not confident with it since it was too much of an abstract idea and I was not sure how it would as an end of year project. But now that I have decided I will be going ahead with my idea of a web application. It will be based on a Patreon style web application where users can sign up and display their individual artworks on an online platform. Other users can look/ buy from other artists

Now as I get into the final weeks of the first semester, I can finally start finding the resources I need for my project. I will have to look into forms of databases that can hold large images and videos so I can store them for my website

December Journal:

In December, I worked on the prototype of my web application for the mid term presentations. I researched the databases I could use to store information from my website like images and videos. A friend I worked with on a project in another module introduced me to the Firebase App that is run by Google. From the testing I did on it, it was clear that this is what I needed to full the rest of the requirements on my project. Starting on Firebase was not an easy task and there was a noticeable learning curve in learning the way the system worked. I was a complete novice when it came to Firebase. It runs with the help of JavaScript and luckily once I got my website live, it was just a matter of expanding my knowledge on how to get the rest of the features live on Firebase.

January Journal:

I spent more time on understand how Firebase works and put it aside in favour of focusing on the rest of my current modules in the new semester. I reviewed some documentation on Googles help site to later use to implement in my application.

February Journal:

No entry for this month, just not focusing on the module as the project is low on my priority list due to having other tasks to finish first.

March Journal:

Covid outbreak has caused my workload to increase as I must find more ways to finish my project since my personal computer at home is inadequate when it comes to running high level programs and is much slower at processing than the college computers.

April Journal:

No work made this month, modules have switched to online learning and I’m not sure if I can complete this project on my current computer, will update in the coming months on my decision.

May Journal:

Due to the Covid outbreak I have decided to defer the module for the next Academic year and finish my studies then.

November Journal 2020:

Returning to finish the project, the lack of effort is showing and getting Firebase to work again is starting to get annoying. Made some progress on the application and got the login signup and registration forms working on the backend of Google functions. Users are also relational with posts and can set likes and dislikes as well as comments

December Journal 2020:

Overdrive mode activated. Firebase has completely failed, and I have to now look for other resources that can help me get some sort of application running. Firebase Functions has proven to be a lot more difficult than I imagined once I began complicating the application. I have restarted the application more times than I would like to admit due to issues I am unable to fix. But I have made some progress on the commerce side and it will have to do with the upcoming deadline.

6.3 Other Material Used

Installation Instructions:

1. Download Source Code

2. Open folder in an IDE
3. Download Node.js package manager
4. Download Yarn package manager
5. Run the following commands in the folder directory

   run: yarn start

   run: yarn i styled-components

   (If an error is requesting to download modules download these first before running yarn start)

   Go to the firebase website and create your firebase application and grab the SDK Javascript functions in the firebase application settings

   run: firebase init

   (Follow steps and install Firebase Hosting, Database and Functions)

   When asked which directory to choose: set to build instead of (public)

   Open the index.html document in public folder and paste database SDK details just above the closing body tags

   run: yarn build

   run: firebase deploy