

National College of Ireland

Software Project

Business Information Systems

Academic Year 2020/2021

Emma Singleton

x18105718

x18105718@student.ncirl.ie

1Cart

Technical Report

Contents

Executive Summary.....	2
1.0 Introduction	2
1.1. Background	2
1.2. Aims.....	2
1.3. Technology.....	2
1.4. Structure	2
2.0 System.....	3
2.1. Requirements.....	3
2.1.1. Functional Requirements.....	4
2.1.1.1. Use Case Diagram	4
2.1.1.2. Use Case Descriptions	4
2.1.2. User Requirements	11
2.1.3. Environmental Requirements	11
2.1.4. Usability Requirements.....	11
2.2. Design & Architecture	11
2.3. Implementation	12
2.4. Graphical User Interface (GUI).....	23
3.0 Testing.....	29
4.0 Evaluation	31
5.0 Conclusions	31
6.0 Further Development or Research	32
7.0 Appendices.....	33
7.1. Project Proposal (Original).....	33
8.0 Objectives.....	34
9.0 Background	34
10.0 Technical Approach.....	37
11.0 Project Plan	37
12.0 Technical Details	38
13.0 Evaluation	38
14.0 Invention Disclosure Form	39
14.1. Reflective Journals	42

Executive Summary

This report will give a detailed account of the project I am working on for the Software Project module. Key points in the report will outline the functionalities and user interface of the application and give details about why they are necessary and how they work. I will then detail my implementation and testing methods and evaluate the system and development process. I will make future recommendations and conclusions of my work towards the end of the report.

1.0 Introduction

2.0 Background

I, myself am a big online shopper. I love to browse websites for hours while I procrastinate other important tasks. I like to add loads of things to my shopping basket and promise myself I will come back on pay day and buy them. Payday arrives, and I am always left thinking “which app was this on? which app was that on?”. I search through my 50+ shopping apps and end up buying 1 full cart on one and leave myself short money of something else I wanted to order on another app.

This is so frustrating to me because then the next payday I have the same issues. First world problems, I know. As I was ordering one day. I thought to myself “I wish there was 1 place I could add all my items to and pay for them all together or see exactly how much the items from all my baskets would total too together”.

I have studied the Java language over the past few years and decided that making an application that could solve this problem would be the best thing for me to do for the module.

2.1. Aims

My plan for this project was to ultimately create a one stop shopping basket that the user can put items from different stores into.

I wanted to be able to create a functional and convenient app on Android Studio with a unique feature that no other apps on the market have.

2.2. Technology

I am using Android Studio to create and execute the app and storing all the stores and items in JSON format. The interfaces are developed using HTML and the code is written in Java.

2.3. Structure

I will discuss the system as a whole. This includes its requirements, functional requirements including the use case and their descriptions, and my updated project plan. This will all describe how the system will work and the main functionalities of the application. We will then take a look at the requirements I set for the project. These will be classified into user requirements, environmental requirements, and usability requirements. Design and architecture will look at the algorithm the project follows, and the implementation describes the purpose and contents of the classes included in the making of the software. I have included some code snippets in this section. The graphical user interface section gives a look into the main screens in the app and describes the features included. I will then describe how I tested and evaluated the finished product and give my evaluation,

conclusion, and further development ideas. The report concludes with the Appendices which includes the reflective journals and the original project proposal.

3.0 System

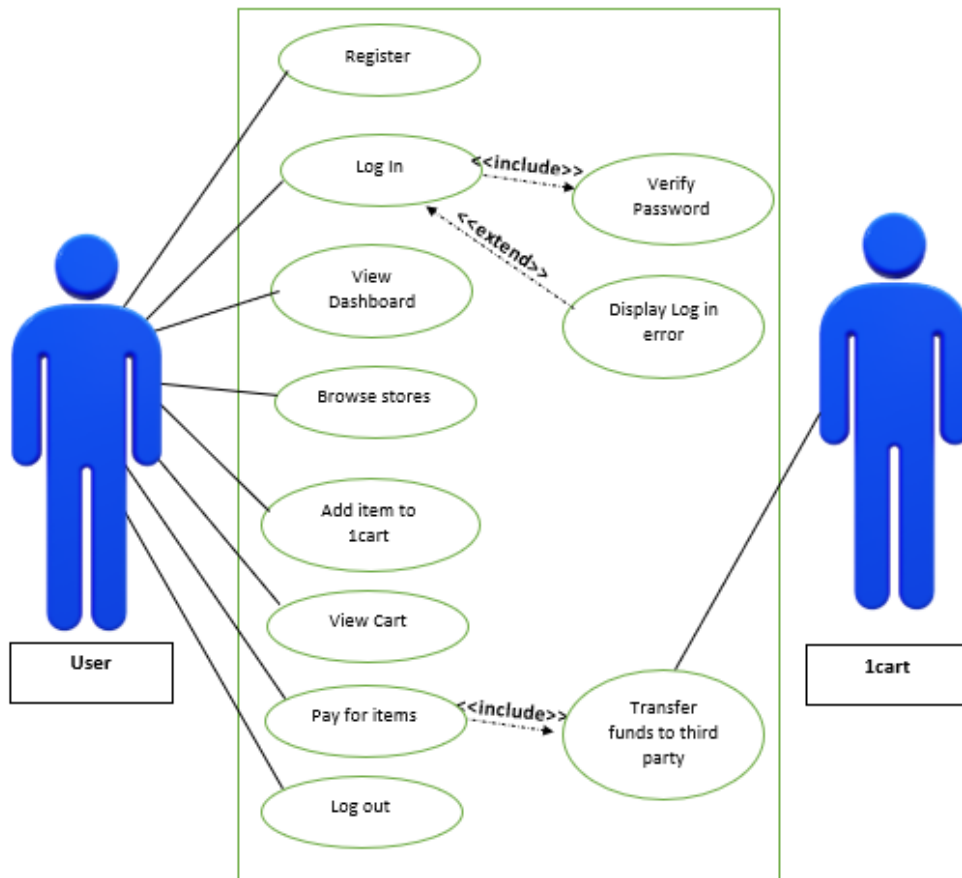
3.1. Requirements

The person using the application must have an android device and be able to run android applications on their system.

1. The system must be able to give the user the opportunity to register an account.
2. The system must then be able to return an error message if the user's credentials do not meet the system requirements.
3. The system must be able to then recognise the users previously registered details and successfully log them into the system if entered correctly.
4. The system must be able to recognise if the user has entered incorrect credentials and return an error message to the user and allow them 4 more attempts to log in.
5. The system must then be able to direct the user to the dashboard where they can choose a store to browse.
6. The system must be able to direct the user to their cart if they choose to view it on the dashboard.
7. The system must be able to scrape data from an external website into the application if the user chooses to add an item to their cart.
8. The system must be able to direct the user to the payment screen and take user input.
9. The system must be able to return a message to the user that their payment has been either successful or unsuccessful.
10. The system must be able to then redirect the user back to the system dashboard.

3.1.1. Functional Requirements

3.1.1.1. Use Case Diagram



3.1.1.2. Use Case Descriptions

ID	<u>UC1</u>
Name	<u>Register</u>
Priority	High
Actors	User, 1cart
Description	The system requires the user to register an account. The user should choose a unique username and enter a password so they can access their account.
Precondition	The user should have downloaded the 1cart app and chosen to register an account.
Activation	The use case starts when the user clicks the “Register” button
Main Flow	<ol style="list-style-type: none"> 1. The user will download the app and then choose to register an account. 2. They click the register button on the screen and the system takes them to the register page. 3. The system asks the user to choose a unique username and a password.

	4. The system allows the users inputs and registers their account.
Alternate Flow	Not applicable in this use case
Termination	The use case ends when the users account details have been registered successfully
Post Condition	After the use case ends, the system has remembered the user's unique username and chosen password in its database.

ID	UC2
Name	Log In
Priority	High
Actors	User, 1cart
Description	This function will allow the user to log into their account using the credentials they have previously registered with the system.
Precondition	The user must have previously registered and account with the system.
Activation	The use case starts when the user has successfully registered an account with the system and clicks the "Log In" button.
Main Flow	<ol style="list-style-type: none"> 1. The user will choose to log into their previously registered account. 2. They enter their credentials, and the system recognises their input.
Alternate Flow	Not applicable in this use case
Termination	The use case ends when the user successfully gains entry to their account
Post Condition	After the use case ends, the user's log in details have been verified by the system and is now able to grant access to the service.

ID	UC3
Name	Verify Password
Priority	High
Actors	User, 1cart
Description	This function will allow the user to re-enter their password if they have entered it incorrectly in the log in stage.
Precondition	The user must have previously registered an account with the system, chosen to log in and entered their log in details incorrectly.
Activation	The use case starts when the user has entered their password incorrectly once at the log in stage.

Main Flow	<ol style="list-style-type: none"> 1. The user will choose to log into their previously registered account. 2. The user enters their credentials but has entered their password incorrectly. The system prompts the user to re-enter their password.
Alternate Flow	Not applicable in this use case
Termination	The use case ends when the user either re-enters their password successfully or re-enters it incorrectly another number of times. If they re-enter it incorrectly several times, the use case will end by redirecting the user to UC4.
Post Condition	If the use case ends successfully, the user will be led to UC5, otherwise they will be led to UC4.

ID	UC4
Name	Display Error Message
Priority	Medium
Actors	User, 1cart
Description	This function will be displayed to the user if they enter their password incorrectly a certain number of times at the log in stage.
Precondition	The user must have previously registered an account with the system, chosen to log in and entered their log in details incorrectly a certain number of times.
Activation	The use case starts when the user has entered their password incorrectly a certain number of times at the log in stage.
Main Flow	<ol style="list-style-type: none"> 1. The user will choose to log into their previously registered account. 2. The user enters their credentials but has entered their password incorrectly. 3. The system prompts the user to re-enter their password. 4. The user enters their password incorrectly another 4 times. 5. The system locks the user out until they close the app completely and reopen it.
Alternate Flow	Not applicable in this use case
Termination	The use case ends when the user chooses to leave the page.
Post Condition	After the use case ends, the user will be led back to UC2.

ID	UC5
Name	View Dashboard
Priority	Medium
Actors	User, 1cart
Description	This function allows the user to view the systems dashboard containing the online stores that pair with the 1cart system.
Precondition	The user must have registered an account, chosen to log in and entered their log in details correctly.
Activation	The use case starts when the user has successfully logged into their account.
Main Flow	<ol style="list-style-type: none"> 1. The user logs in successfully 2. The system directs the user to the dashboard containing the online stores that are compatible with the 1cart system.
Alternate Flow	<ol style="list-style-type: none"> 1. If the user is browsing another page, they can tap the back button and they can get back to the dashboard. 2. The system directs them from their current page to the dashboard page
Termination	The use case ends when the user chooses to view an in app store
Post Condition	After the use case ends, the users online chosen stores will be displayed on screen waiting for their next request to happen.

ID	UC6
Name	Browse Stores
Priority	High
Actors	User, 1cart
Description	This function redirects the user to the online stores website that they have chosen to browse from the system dashboard.
Precondition	The user must have registered an account, chosen to log in, entered their log in details correctly, and chosen the desired store from the dashboard.
Activation	The use case starts when the user has chosen their desired store from the dashboard.
Main Flow	<ol style="list-style-type: none"> 1. The user chooses a store from the 1cart dashboard and browses through the items contained in the store's product list.
Alternate Flow	Not applicable in this use case

Termination	The use case ends when the user chooses to go back to the dashboard or view their 1Cart basket
Post Condition	After the use case ends,

ID	UC7
Name	Add Item to 1cart
Priority	High
Actors	User, 1cart
Description	This function allows the user to add items from the third part website to their 1cart basket
Precondition	The user must have registered an account, chosen to log in, entered their log in details correctly, chosen the desired store from the dashboard, and chosen an item from the store to add to their 1cart basket.
Activation	The use case starts when the user has chosen a desired item from the online store to add to their 1cart basket
Main Flow	<ol style="list-style-type: none"> 1. After the user has chosen an item / items from the online store, they choose to add the item to their cart. 2. They choose the "Add to 1cart" option and it adds the item to their 1cart. 3. The system updates the user's cart with the selected item / items.
Alternate Flow	Not applicable for this use case
Termination	The use case ends when the user has finished adding their chosen items to their basket
Post Condition	After the use case ends, the user will have their 1cart basket updated and the items from the stores can be found there.

ID	UC8
Name	View Cart
Priority	High
Actors	User, 1cart
Description	This function allows the user to view the items they have chosen to put into their 1cart basket from the various stores.
Precondition	The user must have registered an account, chosen to log in, entered their log in details correctly, and chosen to view the items in their basket from the button located in the dashboard.

Activation	The use case starts when the user has clicked the View Cart button.
Main Flow	<ol style="list-style-type: none"> 1. The user will be able to view all the items they have collected in their 1cart basket by choosing the “View Cart” option. 2. The system will display the user’s items in a list.
Alternate Flow	Not applicable for this use case
Termination	The use case ends when the user chooses to proceed on to UC9, or go back to UC5, UC6 or UC7.
Post Condition	After the use case ends, the user’s cart will be displayed with their chosen items, ready for the next action to be requested.

ID	UC9
Name	Pay for Items
Priority	High
Actors	User, 1cart
Description	This function allows the user to pay for the items they have added to their cart.
Precondition	The user must have registered an account, chosen to log in, entered their log in details correctly, chosen the desired store from the dashboard, chosen an item from the store to add to their 1cart basket, chosen to view the items in their basket and have then decided to pay for the items.
Activation	The use case starts when the user has added items from the online stores to their basket and now wants to pay for them.
Main Flow	<ol style="list-style-type: none"> 1. The user will choose to pay for the collected items in their basket. 2. The system will prompt the user to enter their card and delivery details to complete their transaction. 3. Once completed, the system returns a thank you message to the user
Alternate Flow	Not applicable for this use case
Termination	The use case ends when the user has entered their card details correctly and the payment can be completed
Post Condition	After the use case ends, the payment will be forwarded on to the third-party store via the 1cart system.

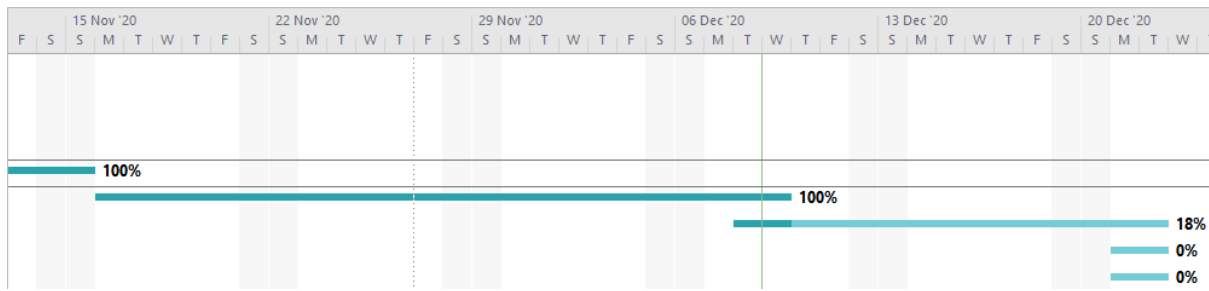
ID	UC 10
----	-----------------------

Name	Log Out
Priority	High
Actors	User, 1cart
Description	This function allows the user to log out of their account.
Precondition	The user must have registered an account, chosen to log in, and entered their log in details correctly.
Activation	The use case starts when the user has chosen to log out of their account.
Main Flow	<ol style="list-style-type: none"> 1. The user will be able to log out of their account by clicking the “Log Out” button. 2. The system will then be closed to the user until they log in again.
Alternate Flow	Not applicable for this use case
Termination	The use case ends when the user has chosen to log out of their account
Post Condition	After the use case ends, the system leads the user back to UC1.

Project Plan

	Task Mode	Task Name	Duration	Start	Finish
✓	★	Prepare pitch video	2 days	Fri 16/10/20	Sun 18/10/20
✓	★	Milestone - Project Pitch upload	1 day	Sun 18/10/20	Sun 18/10/20
✓	★	Prepare Project Proposal	17 days	Sun 18/10/20	Sat 07/11/20
✓	★	Milestone - Project Proposal upload	1 day	Sun 08/11/20	Sun 08/11/20
✓	★	Planning phase	6 days	Mon 09/11/20	Sun 15/11/20
✓	★	Begin development code	18 days	Mon 16/11/20	Wed 09/12/20
	★	Prepare prototype, documents and video for mid point	11 days	Tue 08/12/20	Tue 22/12/20
	★	Milestone - Documentation & Video Presentation	2 days	Mon 21/12/20	Tue 22/12/20

★	Implementation of application	41 days	Mon 04/01/21	Mon 01/03/21
★	Testing of application code	25 days	Mon 01/03/21	Fri 02/04/21
★	Integration of application code	6 days	Mon 05/04/21	Mon 12/04/21
★	Operations and maintenance of application	21 days	Mon 12/04/21	Sat 08/05/21
★	Milestone - Final Implementation & Documentation	1 day	Sun 09/05/21	Sun 09/05/21
★	Milestone - Video Presentation	1 day	Sun 16/05/21	Sun 16/05/21
★	Milestone - Project Showcase	1 day	Mon 24/05/21	Mon 24/05/21



The only timeline that was changed in this plan compared to my previous plan was the development aspect, which I got finished much faster than previously anticipated. This is now reflected in the above plan. I am currently on track for the midpoint delivery.

3.1.2. User Requirements

- The user should be able to register an account with ease.
- The user's details should be stored for login.
- The user should then be able to get into the application after they input their credentials.
- The user should be able to choose an online store by using the system dashboard.
- The user should be able to navigate back to the dashboard and browse other stores.
- The user should be able to choose items they want to purchase and add them to their cart.
- The user should be able to navigate the payment screen.
- The user should be able to make purchases through the app successfully.

3.1.3. Environmental Requirements

- The user must have an android phone. The app will not run on IOS as it has been made in Android Studio.
- The software will only run-on Android devices that have a minimum SDK API 16: Android 4.1 (Jellybean). The application has the ability to run on approximately 99.8% of devices.
- The user must be connected to Wi-Fi or have mobile data switched on.

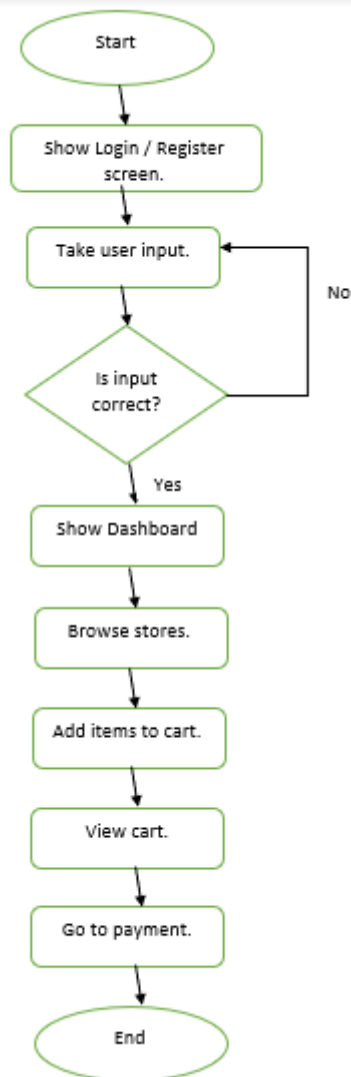
3.1.4. Usability Requirements

- The system should be easy to navigate and 100% of customers should be able to perform a transaction with an online store the first time without assistance.
- It should be extremely user friendly with an easy to navigate interface.
- It should be error tolerant and inform the user if they input information incorrectly and advise them what they should do next.
- The users should be able to flow through the application without being interrupted by irrelevant ads or popups.
- The application must be fit for purpose and be able to handle the requests the user sends.
- The app should give the customer a good experience from the time they open it.

3.2. Design & Architecture

The app is designed to let a user register, then log in. They can then see the stores located inside the 1Cart app. They choose a store they want to browse, and they will be

able to see the products that store is offering. They can then add their chosen items to their cart and go to browse another store. They can then add items from a different store to their cart and keep doing this until they have all their chosen items in the cart. I used a Recycler view layout with a Scrollable object to create the dashboard screen. The user interfaces are stored in .xml files. These are all made from Recycler views that get their objects from a Json file. The classes consist of java code to make the required functionalities work. Below is the algorithm describing exactly how the system should react in the best-case scenario:



3.3. Implementation

Main Activity:

The main activity is the first activity the user encounters when the app opens. It contains the code for the user log in, remember me checkbox and the attempts counter. I am using a class called "Credentials" to get and set the username and password. I have set messages that will display to the user if they log in incorrectly, or if they haven't entered required information into the entry fields.

```

public class Credentials {

    private String Username;
    private String Password;

    Credentials(String username, String password){
        this.Username = username;
        this.Password = password;
    }

    public String getUsername() { return Username; }

    public String getPassword() { return Password; }

    public void setUsername(String username) { Username = username; }

    public void setPassword(String password) { Password = password; }
}

```

```

sharedPreferences = getApplicationContext().getSharedPreferences( name: "CredentialsDB", MODE_PRIVATE);
sharedPreferencesEditor = sharedPreferences.edit();

if (sharedPreferences != null) {
    String savedUsername = sharedPreferences.getString( key: "Username", defValue: "");
    String savedPassword = sharedPreferences.getString( key: "Password", defValue: "");

    Register.cred = new Credentials(savedUsername, savedPassword);

    if (sharedPreferences.getBoolean( key: "RememberMeCheckbox", defValue: false)) {
        nAme.setText(savedUsername);
        pAssword.setText(savedPassword);
        rememberMe.setChecked(true);
    }
}

Toast.makeText( context: MainActivity.this, text: "Please enter valid username and password", Toast.LENGTH_LONG)
}

```

Register:

If the user is registering for the first time, they will enter a unique username and a password. This information is saved to shared preferences and retrieved from the credentials class so it can be recognised anytime they log in.

```

//make a reference to shared preferences
SharedPreferences sharedPreferences;

SharedPreferences.Editor sharedPreferencesEditor;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    registerName = findViewById(R.id.regName);
    registerPassword = findViewById(R.id.regPassword);
    registerButton = findViewById(R.id.btnReg);

    sharedPreferences = getApplicationContext().getSharedPreferences("CredentialsDB", MODE_PRIVATE);
    sharedPreferencesEditor = sharedPreferences.edit();
}

```

Dashboard:

The Dashboard class contains logos of all the stores located inside the app. Each logo has an On Click Listener coded in so when the user taps on a logo, they get directed to that stores page where they can choose their desired products. The dashboard implements the store list adapter class. All of the data relating to the stores and their products are fed in from a JSON file. The JSON file contains the stores names, descriptions, products and prices.

```

{"name": "Elizabeth Arden..."},
{"name": "The Body Shop..."},
{"name": "Disney store..."},
{"name": "Woodies..."},
{"name": "IKEA..."},
{"name": "Harvey Norman..."},
{"name": "Next..."},
{
  "name": "Zara",
  "desc": "Browse and buy from the the Womens and Mens ranges",
  "shipping": 7,
  "image": "data:image/png;base64,
  iVBORw0KGgoAAAANSUheUgAABFMAAAI8CAYAAAA9VuQkAAAAAXNSR0IArs4c6QAAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAFiUAABYLA
  ULSJPAAAAASdEVYdFVhZnR3YXJlAEdyZWVuc2hvdF5VCAUAAP+HSURBVHhe7P33L13Zdd
"products": [
  {
    "name": "Eight Hour Skin Protectant. 50ml.",
    "price": 55.00,
    "url": "https://www.elizabetharden.co.uk/dw/image/v2/AAHP_PRD/on/demandware
    .static/-/Sites-elizabethardenuk-master-catalog/default/dw009ff07e/images/10014252000.jpg?sw=550&sh=550&sm=fit"
  },
  {
    "name": "Eight Hour Miracle Oil. 75ml.",
    "price": 29.99,
    "url": "https://www.elizabetharden.co.uk/dw/image/v2/AAHP_PRD/on/demandware
    .static/-/Sites-elizabetharden-master-catalog/default/dw917a12c9/images/1001eign40143.jpg?sw=535&sh=535&sm=fit"
  },
  {
    "name": "Nighttime Miracle Moisturiser. 120ml.",
    "price": 49.99,
    "url": "https://www.elizabetharden.co.uk/dw/image/v2/AAHP_PRD/on/demandware
    .static/-/Sites-elizabethardenuk-master-catalog/default/dw6af1ef06/images/1001eign00132.jpg?sw=535&sh=535&sm=fit"
  },
  {
    "name": "Intensive Lip Repair Balm. 10ml.",
    "price": 15.00,
    "url": "https://www.elizabetharden.co.uk/dw/image/v2/AAHP_PRD/on/demandware

```

Here I have implemented the JSON file with the store model within the dashboard class so it can retrieve the store models information:

```
private List<StoreModel> getStoreData() {
    InputStream is = getResources().openRawResource(R.raw.appfile); //appfile is the json file name from raw folder
    Writer writer = new StringWriter();
    char[] buffer = new char[10899999];
    try {
        Reader r = new BufferedReader(new InputStreamReader(is, Charset.forName("UTF-8")));
        int p;
        while ((p = r.read(buffer)) != -1) {
            writer.write(buffer, 0, p);
        }
    } catch (Exception e) {

    }

    String jsonStr = writer.toString(); // get data from json file
    Gson gson = new Gson();
    StoreModel[] storeModels = gson.fromJson(jsonStr, StoreModel[].class);
    List<StoreModel> storeList = Arrays.asList(storeModels);

    return storeList;
}
```

Store Model:

The Store Model class implements Parcelable. Parcelable is used to transfer the properties in an android file to another android file. The store model class contains getter and setter methods for the names, images, shipping price and descriptions of the stores. It also contains a CREATOR method which is used to implement the Parcelable feature. The store model passes the product array list on to the Product model class by implementing Parcelable.

```
public class StoreModel implements Parcelable {

    private String name, desc, image;
    private float shipping;
    private List<Product> products;

    protected StoreModel(Parcel in) {
        name = in.readString();
        desc = in.readString();
        image = in.readString();
        shipping = in.readFloat();
        products = in.createTypedArrayList(Product.CREATOR);
    }

    public static final Creator<StoreModel> CREATOR = new Creator<StoreModel>() {
        @Override
        public StoreModel createFromParcel(Parcel in) { return new StoreModel(in); }

        @Override
        public StoreModel[] newArray(int size) { return new StoreModel[size]; }
    };
}
```


Product model:

Like the store model, the product model retrieves and sets up information for the store's products. It also implements Parcelable. In this class we have getters and setters for the products names, prices, images, and the total products the user has placed in their cart. The price is set to float because we are using up to 2 decimal places in the item prices.

```
public class Product implements Parcelable {
    private String name;
    private float price;
    private int totalInCart;
    private String url;

    public int getTotalInCart() { return totalInCart; }

    public void setTotalInCart(int totalInCart) { this.totalInCart = totalInCart; }

    protected Product(Parcel in) {
        name = in.readString();
        price = in.readFloat();
        url = in.readString();
        totalInCart = in.readInt();
    }

    public static final Creator<Product> CREATOR = new Creator<Product>() {
        @Override
        public Product createFromParcel(Parcel in) { return new Product(in); }
    }
}
```

I have added validation in this class that checks if the user has any items in their cart when they click the checkout button. If they don't, they will see a message that tells them to add items to their 1cart. If they do have items in their cart, they will be redirected to the place order class.

```
buttonCheckout = findViewById(R.id.buttonCheckout);
buttonCheckout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { //validation
        if(itemsInCartList != null && itemsInCartList.size() <= 0) { //if there is no items in the cart when the user tries to checkout
            Toast.makeText(context, StoreProductActivity.this, text: "Add items to your 1Cart", Toast.LENGTH_SHORT).show();
            return;
        }
        storeModel.setProducts(itemsInCartList); //update the model
        //direct the user to the place order screen
        Intent intent = new Intent(context, StoreProductActivity.this, PlaceOrder.class);
        intent.putExtra("StoreModel", storeModel);
        startActivityForResult(intent, requestCode: 1000);
    }
});
```

I have included a method to calculate how many items are in the cart and display them.

```

@Override
public void onAddToCartClick(Product product) { //calculates how many items are in the cart
    if(itemsInCartList == null) {
        itemsInCartList = new ArrayList<>();
    }
    itemsInCartList.add(product);
    totalItemInCart = 0;

    for(Product p : itemsInCartList) { //display how many items in cart on checkout button
        totalItemInCart = totalItemInCart + p.getTotalInCart();
    }
    buttonCheckout.setText("Checkout with (" +totalItemInCart +" ) items");
}

```

There are also methods to update the cart list as the user changes the quantities of the products they want and remove products from the list.

```

@Override
public void onUpdateCartClick(Product product) { //update the list when item quantity changes
    if(itemsInCartList.contains(product)) {
        int index = itemsInCartList.indexOf(product);
        itemsInCartList.remove(index);
        itemsInCartList.add(index, product);

        totalItemInCart = 0;

        for(Product p : itemsInCartList) { //get the total number of items and update text on the button
            totalItemInCart = totalItemInCart + p.getTotalInCart();
        }
        buttonCheckout.setText("Proceed To Checkout With (" +totalItemInCart +" ) Items");
    }
}

```

```

@Override
public void onRemoveFromCartClick(Product product) { //remove items from the menu and update interface
    if(itemsInCartList.contains(product)) {
        itemsInCartList.remove(product);
        totalItemInCart = 0;

        for(Product p : itemsInCartList) {
            totalItemInCart = totalItemInCart + p.getTotalInCart();
        }
        buttonCheckout.setText("Checkout (" +totalItemInCart +" ) items");
    }
}

```

Store Product Class:

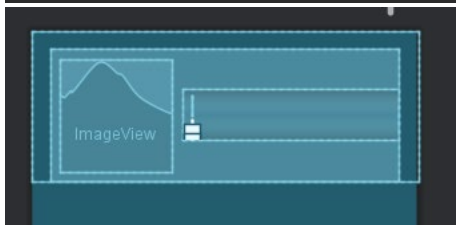
The store product class implements the product list adapter and uses Parcelable to pass the array of products added into the cart to the place order activity.

```
private List<Product> productList = null;
private ProductListAdapter productListAdapter;
private List<Product> itemsInCartList;
private int totalItemInCart = 0;
private TextView buttonCheckout;
```

Store List Adapter:

The store list adapter is one of three adapter classes in my project. Adapter classes create views for each item in a dataset and give access to the data items. I am using RecyclerView to display all the products in the store's interfaces and the stores logo and description in the dashboard. This shows the stores and products in a list-like order.

```
public class StoreListAdapter extends RecyclerView.Adapter<StoreListAdapter.MyViewHolder> {
```



The store list adapter class has methods to get the store name, store description and store image for the dashboard class.

```
@Override
public void onBindViewHolder(@NonNull StoreListAdapter.MyViewHolder holder, int position) {
    holder.storeName.setText(storeModelList.get(position).getName());
    holder.storeDesc.setText(storeModelList.get(position).getDesc());

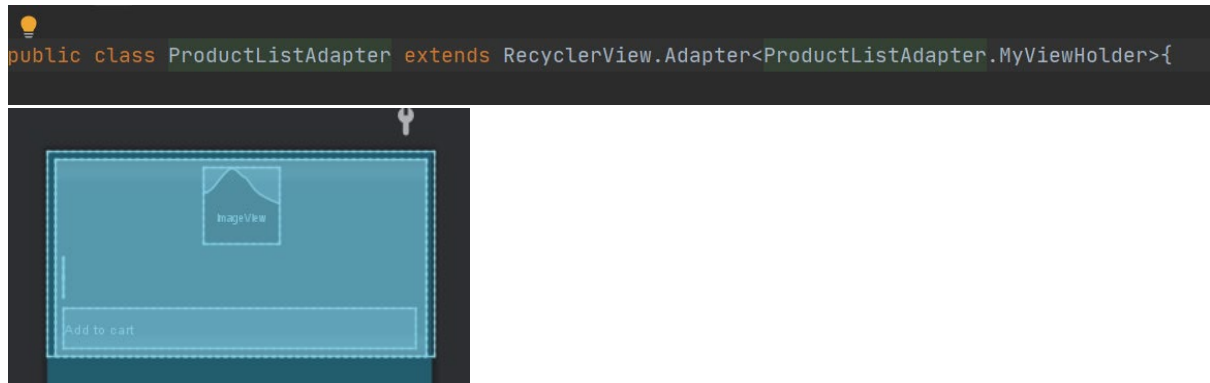
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { clickListener.onItemClick(storeModelList.get(position)); }
    });
    Glide.with(holder.storeImage)
        .load(storeModelList.get(position).getImage())
        .into(holder.storeImage);
```

```
static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView storeName, storeDesc;
    ImageView storeImage;

    public MyViewHolder(View view) {
        super(view);
        storeName = view.findViewById(R.id.storeName);
        storeDesc = view.findViewById(R.id.storeDesc);
        storeImage = view.findViewById(R.id.storeImage);
```

Product List Adapter:

Just like the previous adapter class, the products list adapter class uses recycler view to display the products inside the stores.



This class controls all the elements you see in the different stores. This includes the product name, image, price, add to cart button and the quantity buttons.

```
@Override
public int getItemCount() { return productList.size(); }

static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView productName;
    TextView productPrice;
    TextView addToCartButton;
    ImageView thumbImage;
    ImageView imageMinus;
    ImageView imageAddOne;
    TextView tvCount;
    LinearLayout addMoreLayout;
```

Place Order Adapter:

The Place Order Adapter class also uses recycler view to get and display the products the user has added to their cart. It contains a method to get the selected items from the products array and then displays them in the user's cart with their quantity and price.

```
@Override
public void onBindViewHolder(@NonNull PlaceOrderAdapter.MyViewHolder holder, int position) {
    holder.productName.setText(productList.get(position).getName());
    holder.productPrice.setText("Price: €"+String.format("%.2F", productList.get(position).getPrice()*productList.get(position).getTotalInCart());
    holder.productQty.setText("Item Quantity: " + productList.get(position).getTotalInCart());
    Glide.with(holder.thumbImage)
        .load(productList.get(position).getUrl())
        .into(holder.thumbImage);
```

```

@Override
public int getItemCount() { return productList.size(); }

static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView productName;
    TextView productPrice;
    TextView productQty;
    ImageView thumbImage;

    public MyViewHolder(View view) {
        super(view);
        productName = view.findViewById(R.id.productName);
        productPrice = view.findViewById(R.id.productPrice);
        productQty = view.findViewById(R.id.productQty);
        thumbImage = view.findViewById(R.id.productImage);
    }
}

```

Place Order Class:

This class contains the essential code that the user will use to make their purchase. I am also using a recycler view layout here. First, I declared and initialised all the variables contained in the carts display page:

```

public class PlaceOrder extends AppCompatActivity {

    private EditText inputName, inputAddress1, inputAddress2, inputCounty, inputPostcode, inputCardNumber, inputExpiry, inputCVV;
    private RecyclerView cartItemsRecyclerView;
    private TextView tvSubtotalAmount, tvDeliveryChargeAmount, tvDeliveryCharge, tvTotalAmount, buttonPlaceYourOrder;
    private SwitchCompat switchDelivery;
    private boolean yesToDelivery; //checks if the user wants delivery or collection
    private PlaceOrderAdapter placeOrderAdapter; //reference adapter to use recyclerview

    inputName = findViewById(R.id.inputName);
    inputAddress1 = findViewById(R.id.inputAddress1);
    inputAddress2 = findViewById(R.id.inputAddress2);
    inputCounty = findViewById(R.id.inputCounty);
    inputPostcode = findViewById(R.id.inputPostcode);
    inputCardNumber = findViewById(R.id.inputCardNumber);
    inputExpiry = findViewById(R.id.inputExpiry);
    inputCVV = findViewById(R.id.inputCVV);
    tvSubtotalAmount = findViewById(R.id.tvSubtotalAmount);
    tvDeliveryChargeAmount = findViewById(R.id.tvDeliveryChargeAmount);
    tvDeliveryCharge = findViewById(R.id.tvDeliveryCharge);
    tvTotalAmount = findViewById(R.id.tvTotalAmount);
    buttonPlaceYourOrder = findViewById(R.id.buttonPlaceYourOrder);
    switchDelivery = findViewById(R.id.switchDelivery);

    cartItemsRecyclerView = findViewById(R.id.cartItemsRecyclerView);
}

```



Then I initialised the switch. The switch determines whether the customer wants to collect their products in store or if they want them delivered. If the Boolean is checked, this means they want them delivered. In this case the customer must enter their delivery details. The delivery details field disappears if they want to collect them.

```
switchDelivery.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override //applies if the user wants items delivered - system asks for details
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked) {
            inputAddress1.setVisibility(View.VISIBLE);
            inputAddress2.setVisibility(View.VISIBLE);
            inputCounty.setVisibility(View.VISIBLE);
            inputPostcode.setVisibility(View.VISIBLE);
            tvDeliveryChargeAmount.setVisibility(View.VISIBLE);
            tvDeliveryCharge.setVisibility(View.VISIBLE);
            yestoDelivery = true; //delivery = true
            calculateTotalAmount(storeModel);
        } else { //if they don't want items delivered don't ask for details
            inputAddress1.setVisibility(View.GONE);
            inputAddress2.setVisibility(View.GONE);
            inputCounty.setVisibility(View.GONE);
            inputPostcode.setVisibility(View.GONE);
            tvDeliveryChargeAmount.setVisibility(View.GONE);
            tvDeliveryCharge.setVisibility(View.GONE);
            yestoDelivery = false; // delivery = false
            calculateTotalAmount(storeModel);
        }
    }
});
```

Then we want to calculate the total amount the user has to pay in their cart. “%.2f” means we are using up to 2 decimal places to display prices.

```
private void calculateTotalAmount(StoreModel storeModel) { //calculates the price of products based on prices set in the store model
    float subTotalAmount = 0f;

    for(Product p : storeModel.getProducts()) {
        subTotalAmount += p.getPrice() * p.getTotalInCart();
    }

    tvSubtotalAmount.setText("€"+String.format("%.2f", subTotalAmount)); //format the price to 2 decimal places
    if(yestoDelivery) {
        tvDeliveryChangeAmount.setText("€"+String.format("%.2f", storeModel.getShipping())); //gets the shipping charge from store model
        subTotalAmount += storeModel.getShipping(); //adds total plus shipping
    }
    tvTotalAmount.setText("€"+String.format("%.2f", subTotalAmount));
}
```

The class also contains validations which will be displayed if the user leaves any required fields blank when they try to place their order.

```
private void onPlaceOrderButtonClick(StoreModel storeModel) { //validate delivery details if fields are left empty
    if(TextUtils.isEmpty(inputName.getText().toString())) {
        inputName.setError("Name cannot be left empty ");
        return;
    } else if(yestoDelivery && TextUtils.isEmpty(inputAddress1.getText().toString())) {
        inputAddress1.setError("Address line 1 cannot be left empty");
        return;
    } else if(yestoDelivery && TextUtils.isEmpty(inputAddress2.getText().toString())) {
        inputAddress2.setError("Address line 2 cannot be left empty ");
        return;
    } else if(yestoDelivery && TextUtils.isEmpty(inputCounty.getText().toString())) {
        inputCounty.setError("County cannot be left empty ");
        return;
    } else if( TextUtils.isEmpty(inputCardNumber.getText().toString())) {
        inputCardNumber.setError("Card Number cannot be left empty ");
        return;
    } else if( TextUtils.isEmpty(inputExpiry.getText().toString())) {
        inputExpiry.setError("Expiry date cannot be left empty ");
        return;
    } else if( TextUtils.isEmpty(inputCVV.getText().toString())) {
        inputCVV.setError("CVV cannot be left empty ");
        return;
    }
}
```

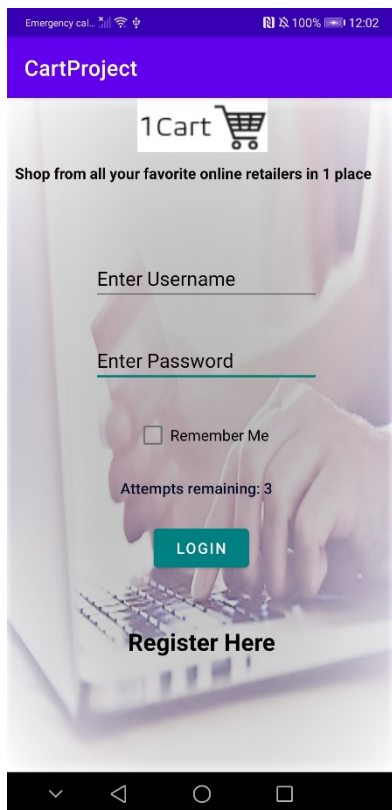
We then want to direct the user to the thank you screen if all the required inputs are correct.

```
//direct user to thank you screen if payment is successful
Intent intent = new Intent( packageContext: PlaceOrder.this, ThankYou.class);
```

3.4. Graphical User Interface (GUI)

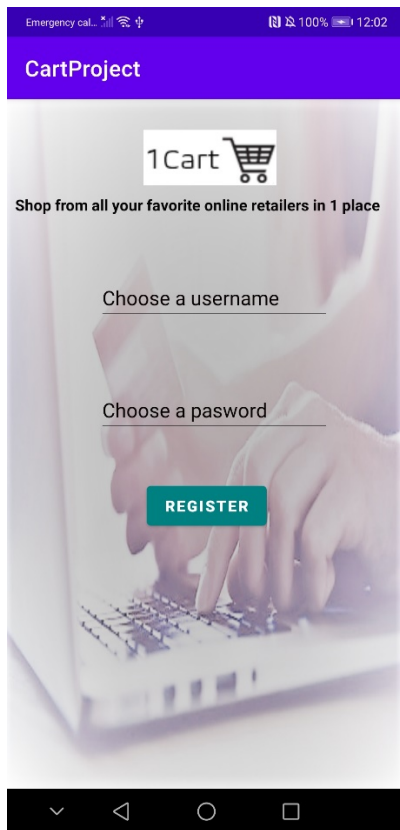
Login Screen:

The Log in screen is the first screen the user sees when they open the app. It contains text views where they are asked to enter their username and password. If they are not registered yet, they can click the Register button underneath the log in button. I have also included a remember me checkbox to save the user from having to log in repeatedly. You can also see an attempt remaining counter. This counts down to 0 if the user keeps logging in incorrectly and locks them out when counted down. I have displayed the project logo and slogan up the top for visual effect. Once the user has logged in successfully, they are directed to the Dashboard.



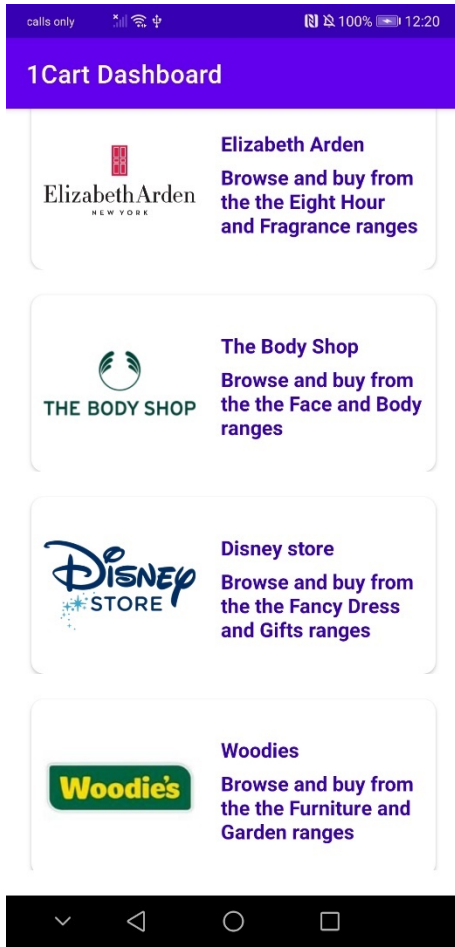
Register Screen:

The registration screen is very similar to the log in screen. It contains the same logo and slogan features as well as the username and password fields. Once the user chooses their username and password they are directed back to the log in screen.

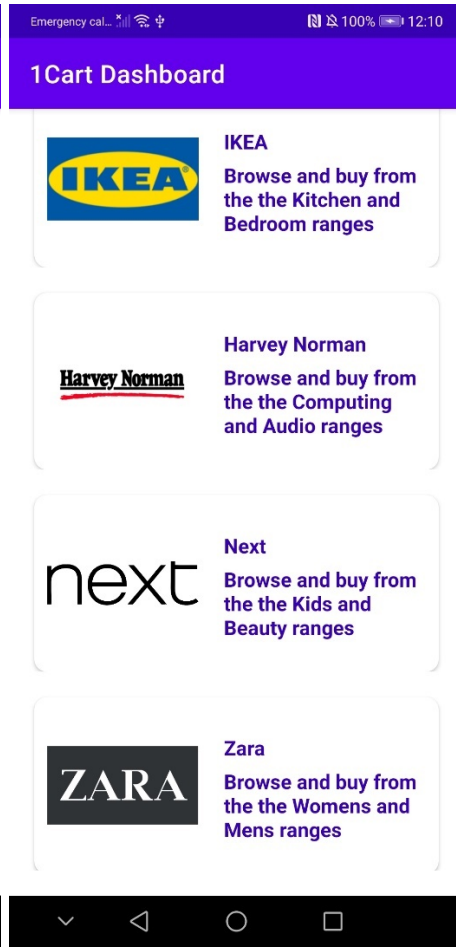


Dashboard and stores:

The Dashboard contains all image views of the store's logos. When the user taps a logo, they then get redirected to that store's screen. I have included Elizabeth Arden, The Body Shop, Disney Store, Woodies, Ikea, Harvey Norman, Zara and Next. The interfaces for the stores can also be seen below. I decided to keep all the stores as similar as possible to ensure ease of use. If I used the data scraping technique to create the stores, they would all look different, and this may deter some users who are not tech savvy from using the app again if they are not sure how to navigate through it easily. As all the stores are based inside 1 app, I wanted it to be as easy to use as possible. This will make users of all levels comfortable with how the app feels as they browse through it. All the stores contain different categories of available products with a photo and title to indicate which products each store offers. The user can very easily add items to their cart by pressing the add to cart button below the product, then enter the quantity of the product they require.



(First half of dashboard)




(Second half of dashboard)

(Elizabeth Arden Store)


Emergency cal... 100% 11:47

← Elizabeth Arden
Browse and buy from the the Eight...




Eight Hour Skin Protectant. 50ml.
Price: €55.0

Add to cart




Eight Hour Miracle Oil. 75ml.
Price: €29.99

Add to cart



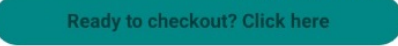
Nighttime Miracle Moisturiser. 120ml.
Price: €49.99

Add to cart



Intensive Lip Repair Balm. 10ml.
Price: €15.0

Add to cart




Ready to checkout? Click here

(Disney Store)


Emergency cal... 100% 11:48

← Disney store
Browse and buy from the the Fancy...




Raya Costume Set Fan Favorite
Price: €58.0

Add to cart




Elsa Costume Frozen 2
Price: €50.0

Add to cart



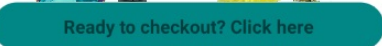
Spider-Man Miles Morales
Price: €50.0

Add to cart



Captain America Costume
Price: €54.99

Add to cart




Ready to checkout? Click here

(The Body Shop Store)


Emergency cal... 100% 12:00

← The Body Shop
Browse and buy from the the Face...




Vitamin E Cream Cleanser. 250ml.
Price: €8.99

Add to cart




Vitamin C Face Mist. 100ml.
Price: €12.0

Add to cart



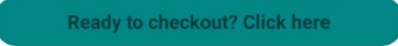
Aloe Soothing Day Cream. 50ml.
Price: €15.0

Add to cart



Anti-Imperfection Night Mask. 75ml.
Price: €12.0

Add to cart




Ready to checkout? Click here

(Woodies)


Emergency cal... 100% 11:50

← Woodies
Browse and buy from the the Furnit...




Verona Wardrobe
Price: €520.0

Add to cart




Malmo Bar Stool
Price: €120.0

Add to cart



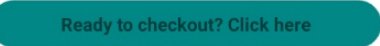
Metal Folding Stool
Price: €12.0

Add to cart



Rocco Velvet Chair
Price: €100.0

Add to cart



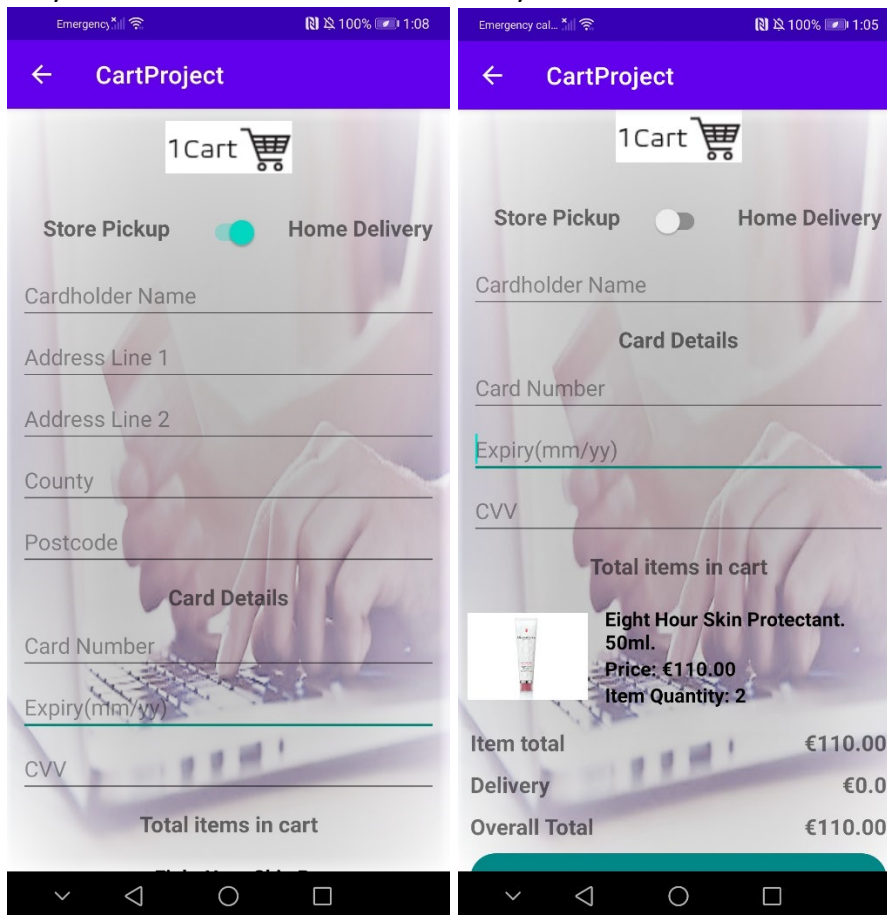
Ready to checkout? Click here

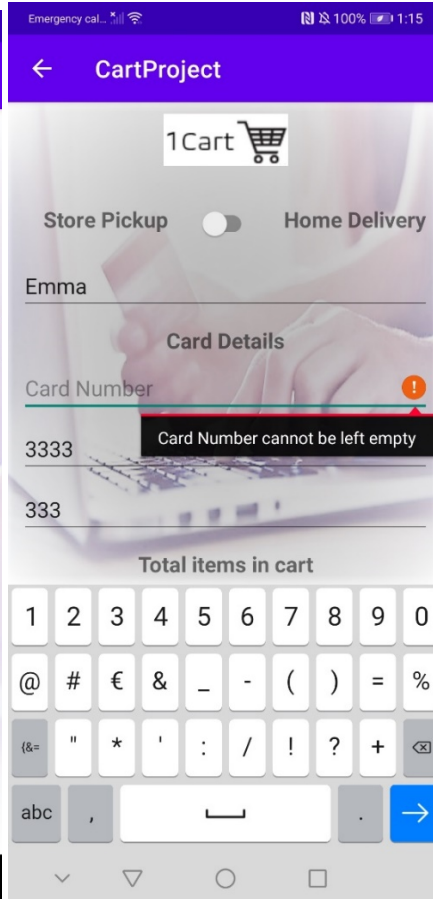
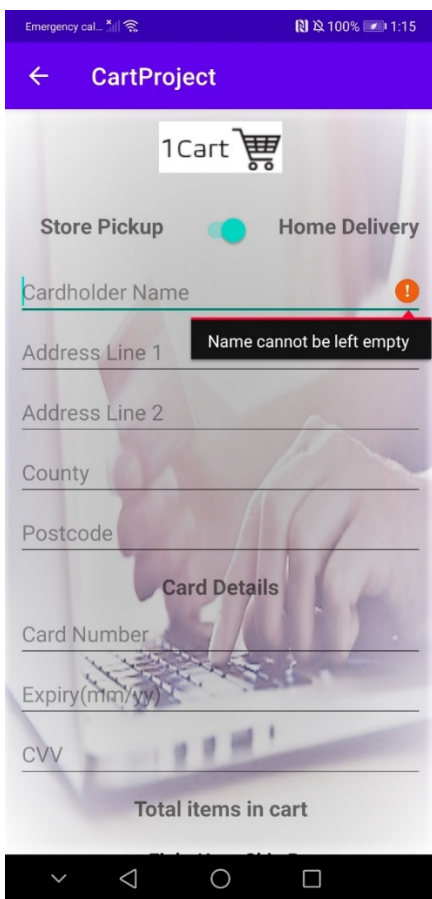
Cart and Payment

Over the last year, click and collect has become one of the only ways we can shop. I decided to add a Store Pickup feature to the cart. There is a switch button placed between store pickup and home delivery.

If the home delivery option is selected, the users address, and other relevant delivery information needs to be entered. The user then proceeds to enter their card details. The home delivery option also adds the relevant delivery charge if this item is ticked.

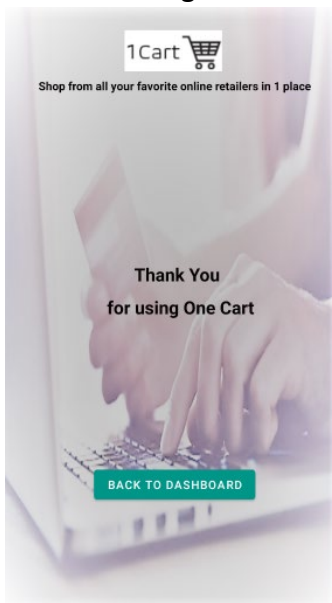
If the store pickup option is selected, the user does not have to enter an address, they just enter the relevant card details needed to pay for their items before they can be collected from their respective store. There is no delivery charge added for the collect in store option. At the bottom of this page, the user can see what items they have in their cart. It shows the item image, name, and quantity. The total for all products in the cart is also displayed here. Underneath the items in the cart there is a place order button. Errors will be displayed if the user has not entered any required details correctly. If all details have been input correctly, they will then be directed to the thank you screen.





Thank the customer screen.

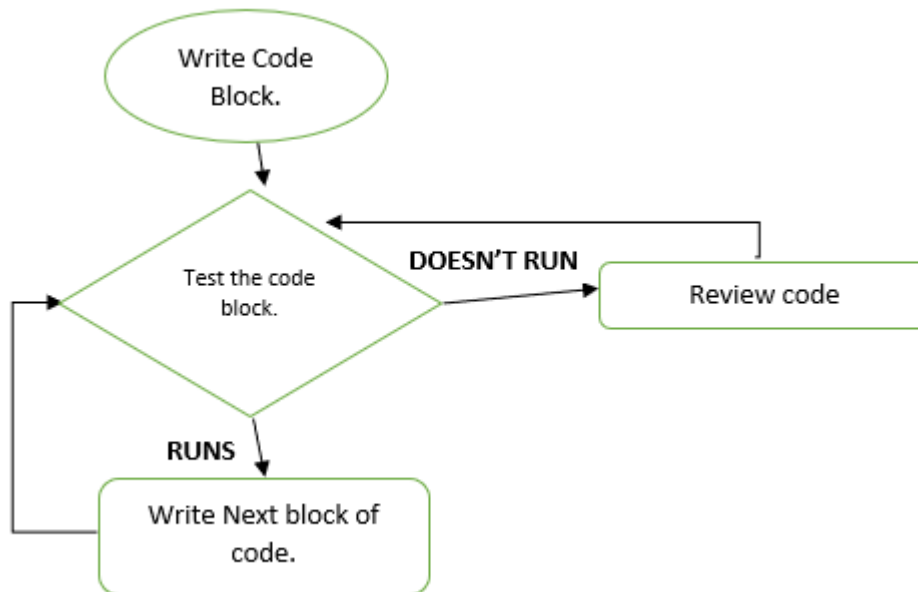
I decided to add a thank you screen that is displayed after the user has paid for their products. This shows appreciation to the customer and lets them know I am thankful that they chose to use my app. It also gives them the option to go back to the dashboard to browse through more stores if they wish to make more purchases.



4.0 Testing

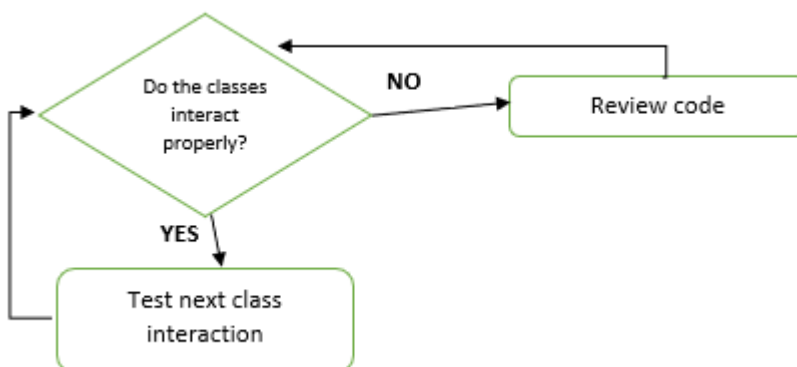
Unit Testing:

I tested the project every step of the way, code block by code block. Any time I added a piece of functionality, I made sure it worked before I moved on to the next step. All the validations I added in each of the classes were tested to ensure they appeared at the appropriate times and under the correct circumstances. The validations mentioned in the above sections were all added and tested to ensure user inputs were correct. I made one class at a time, and when that class worked, I then moved on to the next. This was my strategy throughout the development of the app.



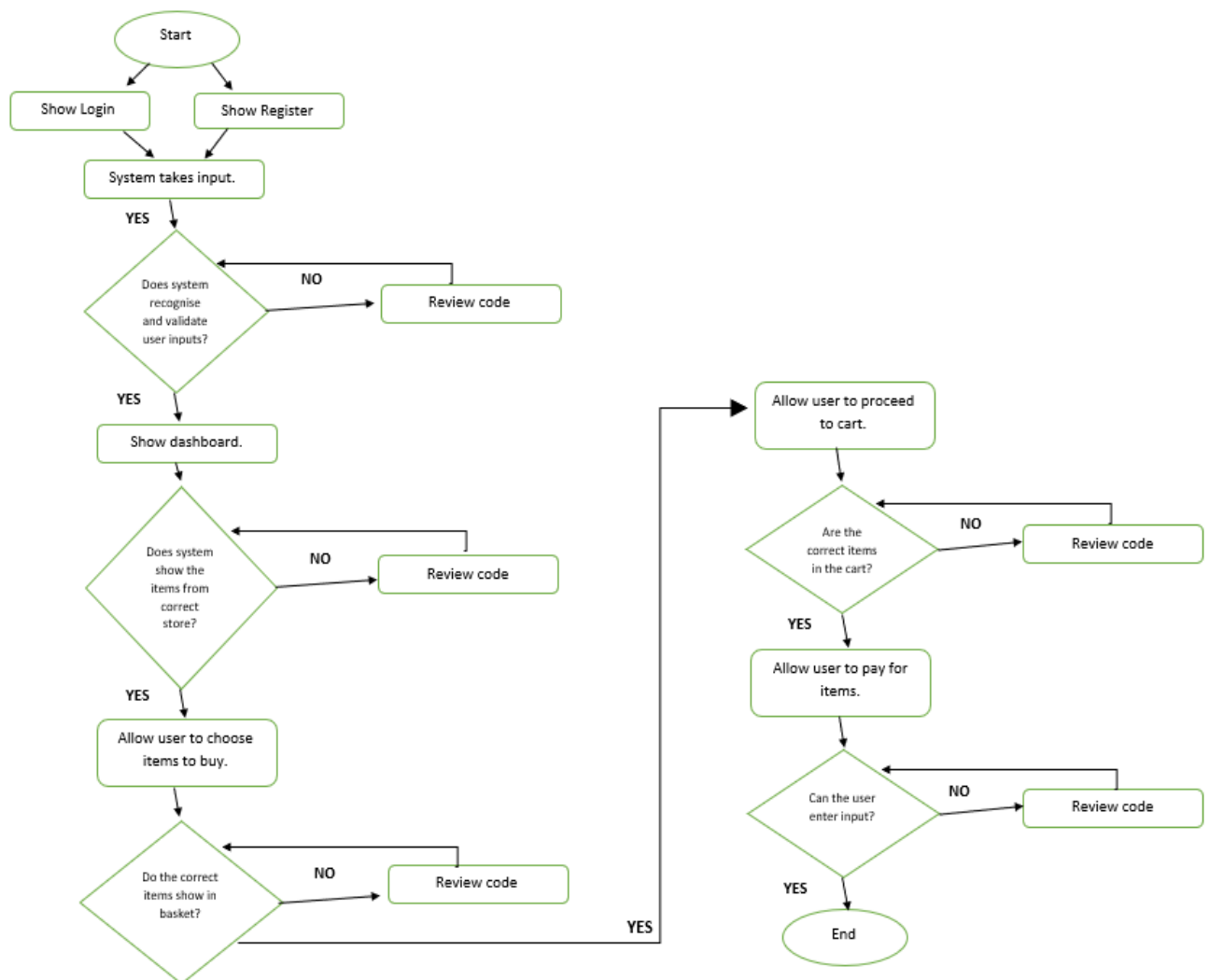
Integration Testing:

Once the unit tests were complete, I then started integration testing to ensure that the classes interacted with each other correctly. Examples of this were when the user added items to their cart, they should then be able to view their cart. The chosen items should be in there. If this did not work properly, I went back and figured out what went wrong and corrected it. Then once this worked, I could move on to test the next piece of functionality, for example, when the items are in the cart and the user wants to check out.



System Testing:

Once I had all the code working for the important functionalities, I then tested the system as a whole. Along the way I experienced many setbacks and bugs. As I fixed one problem, another arose. This was extremely frustrating and took hours and sometimes days to figure out how to fix. In the end, everything seems to be working how I planned. Below is an algorithm of how I tested the system when it was completed. Even though I did unit testing, I still encountered some small bugs when I had the code finished and moved through the finished system. Anytime I came across a functionality that did not work as it should have, I simply went back and reviewed that piece of code and made small changes to ensure it worked properly when I ran the app again.



Test Breakdown:

Test ID	Test Name	Description	Status	State
1	Register	This functionality should let a user choose a username and password. This username and password should match system criteria. If not, system should return appropriate error messages.	Complete	Working
2	Log In	This functionality should recognise the users previously registered username and password. The system should recognise if any inputs are incorrect and display appropriate error messages. If the users' details are correct, the system should give access to the 1Cart app.	Complete	Working
3	View Dashboard	This functionality should get the Store name, image, and description from the JSON file and display them on the dashboard. The user should be able to tap on any store and the system should redirect the user to the relative store products page.	Complete	Working
4	Choose Store	This functionality should be able to pull the individuals stores products and their prices from the JSON file and display them on the correct page according to which store the user chooses.	Complete	Working
5	Add Items to Cart	This functionality should allow the user to add their desired items to their cart by tapping on the add to cart button and choosing the quantity of the product they wish to buy. They should then be able to change the quantity of the item and the system should update this accordingly in the cart.	Complete	Working
6	View Cart	This functionality should allow the user to view the items that have previously added to their cart. This should include the item name, the quantity, price, and product image.	Complete	Working
7	Purchase Items	This functionality should be able to take required user input and recognise if the user has put incorrect input into any of the fields. The system should also recognise if the user has left any required fields empty. The system should then return appropriate error messages, so the user knows what has went wrong. If the user chooses to collect in store, the system should reconfigure the input screen to only take required payment information.	Complete	Working
8	Back to Dashboard	This functionality should show when the user has placed their item. The button on the final screen should lead the user back to the system dashboard where they can start the process again if they wish to do so.	Complete	Working

5.0 Evaluation

Evaluating the system as a whole: I feel the app works well and is fit for purpose. It took me a very long time and a lot of long, stressful days to get the app working. I think it runs smoothly from the Register screen to the Log in screen, then directing the user to the dashboard where they can choose their online store and items then purchase the items from different stores in the one shopping cart. I like the idea that they can choose whether they want to collect their items in their respective stores or get them delivered to their house. I am pleased with the outcome of the app and I hope my hard work has paid off.

6.0 Conclusions

The app development process was very difficult. I had not coded in over a year, so it was tough to get back into the swing of things and refresh my memory. I spent hundreds of

hours learning online through various websites. I restarted the project at least 15 times and made 3 different finished versions. I decided the one discussed above was the best way to deliver the idea. I decided not to use the data scraping technique because I wanted the user interfaces between stores to look the same for ease of use. This was also not possible when it came to storing the stores information and products in array form to be passed from class to class using the parcelable feature.

The main advantage of the 1Cart app is the convenience the user will experience when they are online shopping. This one stop shop for all their online store purchases will make online shopping much easier. It combines all the purchases from their online stores, and they are able to pay for them all in one transaction rather than having to enter their details numerous times on different sites. The main limitation of the app is that obviously, not all online shops are included in the user's dashboard, just a handful of popular sites. It would take a much larger team to make sure all online stores can be made available in the app and that all appropriate permissions can be granted. If the app were to be commercialised, I would need a team of skilled workers to develop it to its full capabilities.

7.0 Further Development or Research

The app could potentially include hundreds of online retailers all in the one place. If there was more development time, or the prospect of commercialisation, the app could include a scheduled payment functionality where the user can choose a time where the app could pay for their selected products, say at 10am on payday (the user would choose the specific date and time). The app could include functionality for the user to store all their virtual and physical receipts so they can easily find and access the ones they want at any time from their phone. This would obviously be a much larger job for a team of skilled developers, but again, I don't know of any other app on the market offering these two functionalities so I think they would be a good idea that would draw people into using this app over a stores individual one.

8.0 Appendices

8.1. Project Proposal (Original)

Project Proposal

1cart

07/11/2020

Business Information Systems

Academic Year 2020/2021

Emma Singleton

X18105718

X18105718@student.ncirl.ie

Contents

1.0	Objectives.....	34
2.0	Background	34
3.0	Technical Approach	37
4.0	Project Plan	37
5.0	Technical Details	38
6.0	Evaluation	38
7.0	Invention Disclosure Form	39
8.0	Bibliography.....	14

9.0 Objectives

My plan for this project is to ultimately create a one stop shopping basket that the user can pull all the items in their carts from their chosen websites into.

I plan to firstly make a login feature where the user can sign up, then sign into their account.

Then the app will display a dashboard with all of the supported websites from which you can cart your items from.

From this screen, the user should choose their desired website, browse that website, choose an item they like and add it to their cart. This will automatically add it to the “1cart” shopping basket. They can then continue shopping or view their cart.

I plan to be able to support the main cart feature by learning how to use data scraping which is a feature I’ve never used or looked into before. Data scraping is the process of channelling data from one website to another. It will be a challenging aspect of my project because it’s an aspect of software development I currently know nothing about, so if I can get it right, I will be very happy. I will be doing lots of research on this in the coming months and will hopefully be able to implement it in my Java coding successfully. It will eventually be the main feature of the application.

Once the user has chosen all the items from their different stores, they can come to their 1click cart, pay for their items all together there and then from the different retailers, or add some items to the “Buy later” cart. The payment would then be forwarded to the respective retailer.

I would also like to implement a feature of scheduled payments so the user can set up a time and date they want the system to automatically purchase the items without actually having to log in themselves, for example, on 9am Thursday when their wages come through. This final feature will ultimately come down to whether I have time to get it in at the end and whether I can learn online exactly how to do it.

10.0 Background

I, myself am a big online shopper. I love to browse websites for hours while I procrastinate other important tasks. I like to add loads of things to my shopping basket and promise myself I’ll come back on pay day and buy them. Payday arrives, and I am always left thinking “which app was this on? which app was that on?”. I search through my 50+ shopping apps and end up buying 1 full cart on one and leave myself short money of something else I wanted to order on another app.

This is so frustrating to me because then the next payday I have the same issues. First world problems, I know. As I was ordering one day. I thought to myself “I wish there was 1 place I could add all my items to and pay for them all together or see exactly how much the items from all my baskets would total too together. So, this led me to think that surely there was already something like this available in the app store. I did some searches and found an article about a company called “Keep”. Keep basically detailed itself as a place to discover and buy all the latest trends online. It offered a feature called “One Cart” which, ironically, I

had thought up myself as a name for my application. The “One Cart” feature was what I had described I would like to have in an app. Below is an excerpt from an article I found at:

<https://www.racked.com/2015/7/13/8950541/keep-app-layoffs>

Social shopping app Keep is scaling back. The company confirmed to Racked that it's getting rid of the app's universal cart feature and the staff is downsizing as a result of the decision. "Keep has stopped its experiment of a universal cart and has gone back to affiliate refer to retailers," Scott Kurnit, Keep's CEO, told Racked in an email. "In the process there are some people who are leaving the company." Kurnit declined to disclose the number of layoffs.

Keep's universal shopping cart debuted on the app last year. It promised to streamline user experience by allowing shoppers to browse and buy right in the app, instead of redirecting to the retailer's website to purchase an item discovered in Keep. "At a high level, it worked great, consumers loved it, but it was expensive to operate," Kurnit explains. "We'll be looking at ways to do much or all of the same thing at lower cost."

As we can see, this companies' app did not go to plan. I decided to search for the application on the Google Play Store myself to see the reviews. I have attached my findings in the next page. From the reviews of the app, we can see that it was actually pretty popular before its demise, which leads me to believe there is definitely a gap in the market for this type of idea. I didn't find any others of its nature while doing some research so it may well be a rarity of its kind. I don't plan on taking the app any further than its intended project life cycle of my final year software project, but it's nice to know that, if it all works out, my project is unique in nature.

About this app

Discover & buy the latest trends in fashion, home decor, beauty and design.

Shopping

Ratings and reviews



Rosaly Rosario

★★★★★ 28/12/2018

update: DELETING IT!!!! It's been months and still won't let me log in to my account! Continues to tell me there's an error on their ...

Was this review helpful?

Yes No

Jem W.

★★★★★ 21/11/2018

I used to love this, however I got a new phone and now it won't let me log in. When you go to reset your password it just comes up with an...

Jem W.

★★★★★ 21/11/2018

I used to love this, however I got a new phone and now it won't let me log in. When you go to reset your password it just comes up with an...

Was this review helpful?

Yes No

Amy T

★★★★★ 25/04/2019

Deleted app. It has not let me log in for months. What used to be a fun and helpful app is now useless!

Was this review helpful?

Yes No

[See all reviews](#)

Keep Shopping from Keep... 3.2★
Ratings and reviews

A Google user

★★★★★ 28/10/2018

It doesn't work! Can't log on and give me this little picture in another language....really!

Was this review helpful?

Yes No

Dan Knittle

★★★★★ 27/10/2018

App doesn't load. Constant spinning icon

Was this review helpful?

Yes No

S. B.

★★★★★ 05/01/2019

This was one of my favorite apps. Not sure why but it does not work anymore

Was this review helpful?

Yes No

Cassidy Flatt

★★★☆☆ 23/11/2018

keep shopping app



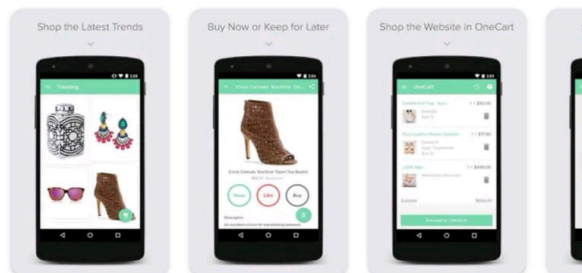
Keep Shopping from K...
Keep Shopping from Kee...

Install

3.2★
738 reviews

50K+
Downloads

3
PEGI 3



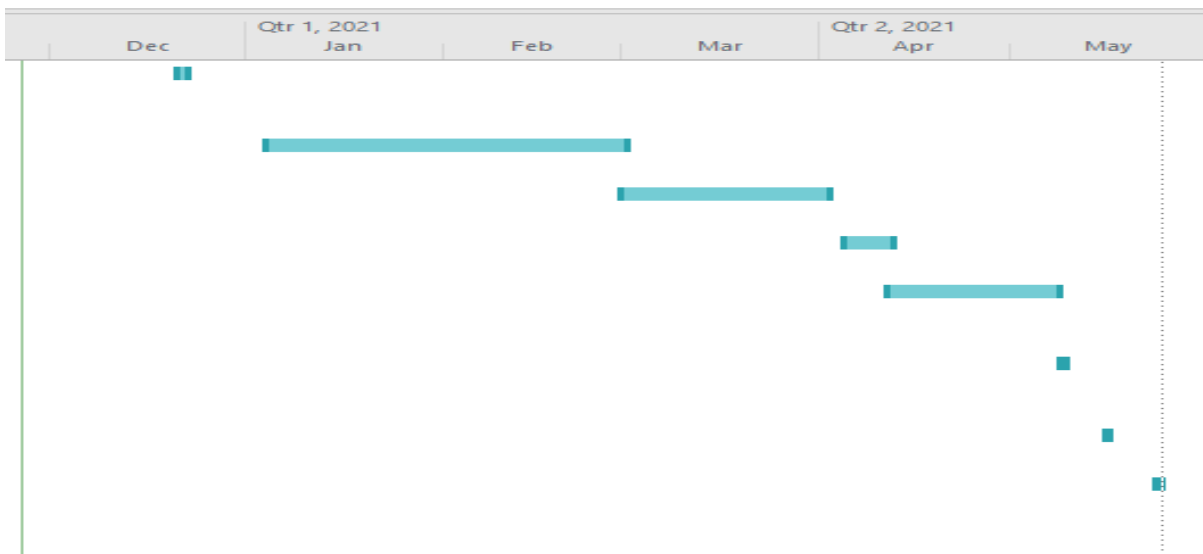
Discover & buy the latest trends in fashion, home decor, beauty and design.

11.0 Technical Approach

1. User Register
2. User Log in
3. System Dashboard
4. User chooses website they wish to browse.
5. User chooses product and adds to 1cart basket.
6. System directs them back to their 1cart basket.
7. User can see their overall total for all items in their cart from different websites.
8. User can choose to go back to the dashboard and browse other websites or pay for items.
9. If the user chooses to pay, they are redirected to the payment screen.
10. User enters their card details and display message lets the customer know they have paid for their items.

12.0 Project Plan

	Task Mode	Task Name	Duration	Start	Finish
✓		Prepare pitch video	2 days	Fri 16/10/20	Sun 18/10/20
✓		Milestone - Project Pitch upload	1 day	Sun 18/10/20	Sun 18/10/20
✓		Prepare Project Proposal	17 days	Sun 18/10/20	Sat 07/11/20
✓		Milestone - Project Proposal upload	1 day	Sun 08/11/20	Sun 08/11/20
✓		Planning phase	6 days	Mon 09/11/20	Sun 15/11/20
		Begin development code	26 days	Mon 16/11/20	Mon 21/12/20
		Prepare prototype, documents and video for mid point	11 days	Mon 07/12/20	Mon 21/12/20
		Milestone - Documentation & Video Presentation	2 days	Mon 21/12/20	Tue 22/12/20
		Milestone - Mid Point Implementation,	2 days	Mon 21/12/20	Tue 22/12/20
		Implementation of application	41 days	Mon 04/01/21	Mon 01/03/21
		Testing of application code	25 days	Mon 01/03/21	Fri 02/04/21
		Integration of application code	6 days	Mon 05/04/21	Mon 12/04/21
		Operations and maintenance of application	21 days	Mon 12/04/21	Sat 08/05/21
		Milestone - Final Implementation & Documentation	1 day	Sun 09/05/21	Sun 09/05/21
		Milestone - Video Presentation	1 day	Sun 16/05/21	Sun 16/05/21
		Milestone - Project Showcase	1 day	Mon 24/05/21	Mon 24/05/21



13.0 Technical Details

I will be using Android Studio to develop this application. I have briefly used this technology in past projects, but not in depth so it will be a new learning experience for me. I will be learning how to use the software to its full capacity.

I will be writing the application code using Java. This will include making classes.....

While researching some popular Android studio libraries, some helpful ones I came across and will be using while developing this app are:

Glide: This library is all about image loading. It applies automatic smart down-sampling and caching to minimise storage overhead and decode times. Its byte array recourses are reused, and it releases app recourses automatically where they are necessary.

Picasso: Like Glide, this is another image loading library in Android Studio. It simplifies displaying images from external websites. It supports complex image transformations, Image View recycling aswell as automatic caching to disk.

Retrofit: This library is about network communication. It is a type-safe HTTP client for Android and Java. It is said to be the most popular networking library in Android development. With this library, we can query parameters, manipulate endpoints, add a request body and choose request methods.

Dagger2: This library looks after Dependency Injection. This concept says that an object does not need to configure its dependencies. Dagger2 is a “fully static compile-time dependency injection framework for both Java and Android”. A feature of the new Dagger2 version includes the auto generation of subcomponents using a code generator.

Databinding: This library is in-built into the development support library.

SQLite: Android studio’s built-in database is SQLite. It is an opensource database that stores data to a text file on a device. One of SQLite’s well-known benefits among android mobile developers is its native support on Android. I have never used this database with Android Studio, on my last team project, my colleague in charge of the database aspect of the project paired it with Firebase, so this will be a first for me. I will be learning from scratch using online tutorials.

14.0 Evaluation

Upon research, I found the following integrated system libraries that will assist in the testing process:

Junit: This is one of the testing libraries that I found during research that is a library in Android Studio. It is a framework for unit testing. In this test, individual parts of the source code get tested. The framework has a set of Assert methods which check your expected result against the actual result. It makes use of heavy annotations.

Robolectric: This is another testing library available in Android Studio. The difference between this and Junit is that this one was created to help developers. It handles inflation of views, recourse loading aswell as lots of other things. It will allow our tests to do most of the same things we could carry out on a device that Android framework dependencies. This test simulates the Android SDK in the tests. We don't need additional mocking frameworks with this type.

15.0 Invention Disclosure Form

Please fill in the following sections if you think your idea is innovative:

1. Title of Invention

1cart

2. Inventors

<u>Name</u>	<u>School/Research Institute</u>	<u>Affiliation with Institute (i.e., department, student, staff, visitor)</u>	<u>Address, contact phone no., e-mail</u>	<u>% Contribution to the Invention</u>
Emma Singleton	National College of Ireland	Student	29 Temple view Row, Clare Hall, D13P2W0. 0852326847 X18105718@student.ncirl.ie	100%

3. Contribution to the Invention

Each contributor/potential inventor should write a paragraph relating to his/her contribution and include a signature and date at the end of the paragraph.

I, Emma Singleton, will be responsible for all contributions made to this project.

DocuSigned by:
emma singleton
40DCDF07723849B...

07/11/2020

4. Description of Invention

(Please highlight the novelty/patentable aspect. Attach extra sheets if necessary, including diagrams where appropriate). What is novel, the 'inventive step'? For more information on patents, please look at <http://www.patentsoffice.ie/en/patents.aspx>

A one stop shop for all the users online shopping baskets.

5. Why is this invention more advantageous than present technology?

What are its novel or unusual features? What problems does it solve? What are the problems associated with these technologies, products or processes? Explain how this invention overcomes these problems (*i.e.*, what are its advantages).

As I have outlined in my report above, there isn't, from what I can find, a rival app of the same nature on the market at the minute. It would be unique and innovative for online shoppers.

On researching the demise of the previous company who tried this app out, they say it was too costly to maintain their business. By trying to produce this application on my own with no overheads, may overcome that problem. Obviously in a setting where the product grew and a company was formed, it may well face the same challenge. This would be when you would have to take a look at more forms of income relating to the app.

6. What is the current stage of development / testing of the invention?

Pre-development stage.

Development due to start on 16/11/2020.

7. List the names of companies which you think would be interested in using, developing or marketing this invention

Google Play Store

App Store (Apple)

PEGI 3

8. Funding Partner(s)

Government Agency & Department	None
% Support	None
Contract/Grant No.	None
Contact Name	None
Phone No.	None

Address	None
---------	------

Industry or another Sponsor	None
% Support	None
Contract/Grant No.	None
Contact Name	None
Phone No.	None
Address	None

9. Where was the research carried out?

Google and Google Play store.

10. What is the potential commercial application of this invention?

Project- to have multiple baskets in the one place.

One stop shopping basket

- Login feature
- Dashboard
- Databases set up.
- Scraping of Data
- Payments transferred to different retailers.

11. Was there transfer of any materials/information to or from other institutions regarding this invention? No

12. Have any third parties any rights to this invention? -No

13. Are there any existing or planned disclosures regarding this invention?

Please give details.

Ethics form attached to submission.

14. Has any patent application been made? No

15. Is a model or prototype available? Has the invention been demonstrated practically?

No. The prototype will be available in due course and the finished model will be available for grading in May 2021.

I/we acknowledge that I/we have read, understood and agree with this form and the Institute's *Intellectual Property and Procedures* and that all the information provided in this disclosure is complete and correct.

I/we shall take all reasonable precautions to protect the integrity and confidentiality of the IP in question.

DocuSigned by:
emma singleton
40DCDF07723649B...

Inventor: Emma Singleton

Date 07/11/2020

Bibliography:

- Erica Adams (July 13, 2018) Shopping App Keep Has Laid Off Staff, Is Downsizing
Available at: <https://www.racked.com/2015/7/13/8950541/keep-app-layoffs>
Accessed on 07/11/2020.
- Idorenyin Obong (2018) Android Development: 15 Libraries you should be using.
Available at: <https://auth0.com/blog/android-development-15-libraries-you-should-be-using/>
Accessed on 07/11/2020.

15.1. Reflective Journals

October

The main goal for October was to think of an idea for the project. My aim was to produce an innovative piece of software that got approved by my supervisor and started getting a plan in place regarding the development process. I thought about the best way to approach the requirements phase and had a meeting with my supervisor about getting started.

November

This month I started thinking about how I will development my application. I also completed the project proposal and got it uploaded. I have made a project plan with a timeline and have begun gathering requirements, functional and non-functional. I have also been looking at videos of Android Studio and the level of difficulty in making projects. It seems to be okay and very user friendly, so I have decided to use this software to develop the app.

December

This month I started the development process. I made 5 or 6 different projects (all wrong with many errors) on Android studio before I started to get the hang of it. I watched lots of tutorials on YouTube and figured out how to collaborate many different types of coding skills to make my prototype. I got my prototype finished, made a documentation video, did the technical report, PowerPoint presentation and video report. I was way ahead of schedule and got all this finished by December 9th. This left me with 2 weeks to get another project completed for another module without the stress of having to juggle both. I will take a break now from the software project until January when I will get back to further development for the next semester.

January

The main goal for January was to start thinking about what comes next. The midpoint submission has the main bones of the app created and implemented, but now I am starting to think about the web scraping aspect of the project which will be the main selling point of the application. I have gotten in touch with Emer Thornbury who has recommended me having a 1:1 with a Data Analytics tutor which she will set up for me next week. I am hoping to get their opinion on the best way to go about the data scraping functionality as I have never come across or studied this in any way and can be a big task. Once I meet with the tutor next week, I will then decide what the best course of action is. I will then also start working on the payment functionality aspect of the project.

February

For the month of February, the main aim was to get some more of the design done for the app. I wanted to get a better image in my mind and on paper what the inside of the app would look like. I got a lot of new HTML done and decided it would be better for me to manually enter the products from the external websites rather than scrape them so I can make the app very consistent and user friendly. I want the app to have the same look and feel as the user goes from store to store. I want the overall app experience to be enjoyable and seamless and easy to navigate for any type of user so I feel that entering the products and details manually rather than scraping them will be a better direction to go in for the sake of the project design.

March

All through March I was extremely busy with other assignments, so I took this month to do a lot of research online about how different shopping apps look and have a think about how exactly I want my end product to look. I didn't do much in the way of coding this month, but I did a lot of brainstorming for my finished product and experimented with different shopping cart features from YouTube tutorials. I wanted to find the right tutorials that would help I'm when trying to code my cart and payment screen. I have found lots of online material that I will be putting into practise during the month of April and start getting ready to put all the pieces together and make my app complete.

April

This month I restarted my project from scratch. I wanted to include Firebase as an online database that will take the users details in when they register and validate these details when logging in. This required all new code. I then worked on the user interfaces throughout the app and developed the cart and payment systems.