

**Submitted By:**

Dana Barhoom (a00200)

**University:**

American University of Bahrain

**Degree:**

Bachelor of Science in Computer Engineering

**Module:**

Mobile Programming COSC 348

**Assignment:**

Mobile Programming Project

**Date:**

15/12/24

Table of Contents

[Project Overview: 3](#_Toc185472809)

[Objectives: 3](#_Toc185472810)

[Scope: 3](#_Toc185472811)

[Technical Specifications: 4](#_Toc185472812)

[Data Storage: 4](#_Toc185472813)

[Architecture: 4](#_Toc185472814)

[Libraries Used: 5](#_Toc185472815)

[Design Documentation: 5](#_Toc185472816)

[1. Login page: 5](#_Toc185472817)

[2. Home Screen: 6](#_Toc185472818)

[3. Item Details Screen: 8](#_Toc185472819)

[4. Products Page 9](#_Toc185472820)

[5. Favorites/Cart Page: 10](#_Toc185472821)

[6. Profile screen: 12](#_Toc185472822)

[Implementation Details 13](#_Toc185472823)

[1. Adding Items to Cart 13](#_Toc185472824)

[2. Edit - Incrementing and Decrementing: 13](#_Toc185472825)

[3. Deletion from the Cart 13](#_Toc185472826)

[4. Saving Data 14](#_Toc185472827)

[5. Loading Data 14](#_Toc185472828)

[6. Navigation Links 14](#_Toc185472829)

[7. Filtering Products: 15](#_Toc185472830)

[8. Editing Text 15](#_Toc185472831)

[Testing and Validation 16](#_Toc185472832)

[Unit Testing: 16](#_Toc185472833)

[User Testing: 16](#_Toc185472834)

[Challenges and Solutions 17](#_Toc185472835)

[Future Enhancements 17](#_Toc185472836)

# **Project Overview:**

* In this project, it is an implementation of swift coding langauge using Xcode. It is a simple clothing app shop with a list of retail clothing products. The app is made to help the user save their favorite product from the shop onto the shopping cart and buy it if they want to for later. The app gives the ability for the users to do basic tasks like adding clothes, deleting a clothing item and being able to view them in a listed way and scroll over them.
* The purpose of completing this projet is to ensure the understanding swift coding language and grasped by creating a simple and functioning app for the user to save clothing items. This shows the understanding of Swift programming and the SwiftUI framework.

## **Objectives:**

The main objective of creating this project is to create an app that makes it easy to manage clothing items, like adding the item, being able to view it or deleting them. This is to practice and enhnace SwiftUI coding ability, in addition to showcasing what was learned in the course in an organized user friendly project. The goal of this project, was to demonstrate all that was learned including the basics of adding, viewing, and deleting items in a simple way.

## **Scope**:

* The app that was created has the ablity to preform a range of services for the user. It can log the user in, display a list of all the clothings with their name, price, and a small image showing how the product looks like. It also has the main ability to add the items to cart, where it also will show a snapshot of their information such as their name, price, in addition to being able to remove the products that are no longer wanted.
* In additon, the items added to cart can be viewed again since they are saved.

# **Technical Specifications:**

The techincal aspects and details that is present in the project is first the use of a apple-based device such as a Macbook to complete the project. The Programming Language that was used to complete the project is Swift including the UI framework which is SwiftUI. The integrated development enviroment tool that was used to complete the project is Xcode for Mac.

## Data Storage:

* The data storage method that was used in this project is the Swift's State and Binding properties that has the ability to handle the data present in the app. These methods can be seen throughout the app coding components.
* The app currently has the ability to save the products on the cart itself for future lookback, but it doesn’t permanently keep their amount in the cart to allow other customers to buy the product.

## Architecture:

* In terms of architecture, I used a **simple MVVM** architecture**. This is Model, View and ViewModel** architecture. This architecture was used since it simplifies the working of the app where the **Model aspect would** represents the data, like products and cart items. The **View** apsect shows the screens the user sees, such as Home, Cart, and Profile and **ViewModel** manages the logic, like adding items to the cart or saving data.
* This would keep the code clean and organized so that the data and user interface are separate.

## Libraries Used:

The libraries that were used were SwiftUI and the Foundation inbuilt Apple libraries that were in Xcode. There was no additional libraries used to create this app because the app is made using simple swift coding language.

# **Design Documentation:**

These are wireframes of what each screen would look like. Wire frames can be seen as rough sketch versions of what the app looks like.

## 1. Login page:

* The login page is a simple login page where the user saves his/her profile by entering in their name and email address.
* These details are then saved onto the profile page.
* The users can interact with this screen by entering in their email and name, and then clicking on login.
* This would then save their details and place them in the profile screen.

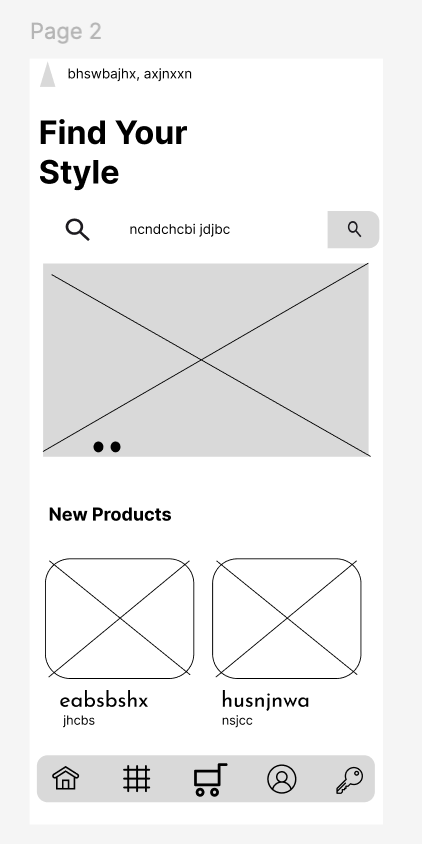
A screenshot of a login form

Description automatically generatedA screen shot of a phone

Description automatically generated

## 2. Home Screen:

* This is the first or main screen where the focus of this page is highlights of product’s picture section where some items are previewed in an animated manner, where the pictures are displayed on a bigger box, to promote some of the new products present in the store.
* Under this, at the bottom of the screen the user can see also the previous highlighted clothing items where they can scroll horizontally to view them. The user can also click on the grid icon in the corner next to new products, to view the list of products. If the user clicked on any of the items, it would take them to the item detail page.
* Each of the items at the buttom of the page has a small picture, the name of the item, and the price as shown in the wirefames below.
* You can search for an item using the search feature on the homepage. If you type in letters like j for jacket, or any letter, that letter or only jackets will show in the product section of the homepage.

A screenshot of a phone

Description automatically generated

## 3. Item Details Screen:

* This page’s purpose is to show a detailed information about the item, how it looks, to view the colors available, to see it’s rating, the price and the ability to add the product to cart if not done already.
* The button named nxjnsxjhc dhc nsnxnjk is there for adding the product to the cart.
* After that the user can go to the cart to view their product and pay for it if they wish to buy it.
* Users have the ability to interact with the button that adds the product to cart and then they can make the decision to buy it after reviewing it’s details.

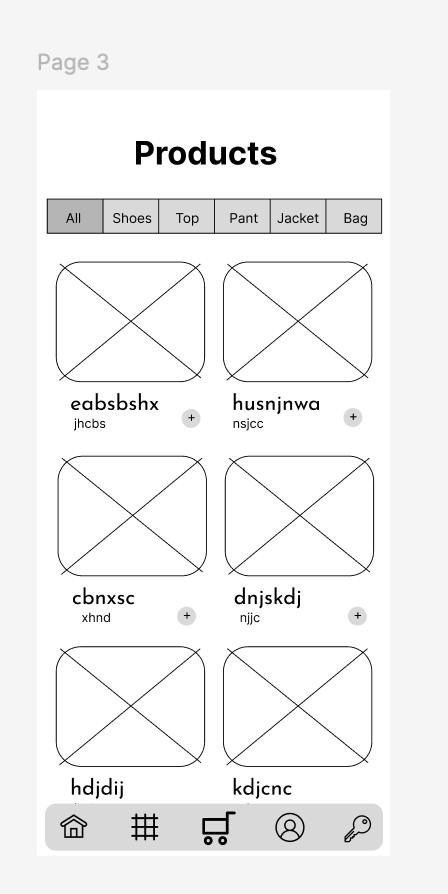
A screenshot of a phone

Description automatically generatedA phone with a price list

Description automatically generated

## 4. Products Page

* For this page, the user can navigate to it either by clicking on the grid icon which takes them to the same page or by clicking on the grid icon in the homepage next to ‘new products’. This page shows all the available products to pick the user’s favorite and add them to cart. The display shows a grid of products with an option for the user to add their favorites to cart to view them later. They can press on the product and then they will be taken again to view the item detail screen which is the description of the product.
* Users can intercat with this page by viewing products in a grid layout display.
* Each product has a + button, which is a quick add method that is used to add the product they desire to their favorites cart.
* The product can be filtered out per category, if they want to see only specific items, which is shown on the top of the screen. These include All, Shoes, Top, Pant, Jacket, and Bag. This would allow the users to filter the list of displayed items.
* The items are vertically scrollable to view all the products available.

A screenshot of a phone

Description automatically generated

## 5. Favorites/Cart Page:

* The purpose of this screen is to display a list of the items that were added from either the item details page or the products page which were added as favorites or to the cart.
* Items are listed vertically.
* The user can interact and edit the quantity of the item added in this page using the options for increasing + or decreasing -, the quantity of the item. The user can also delete the entire product using the trash icon from the list.
* Users can view the product’s small image, it’s names, the quantity and price of the product.
* The total for their selected items can also be shown on this page to show how much they would have to pay to get their favorite items.
* There is a button ("nxjnsxjhc dhc nsnxnjk") at the bottom, which is there for if the user desires to proceed to buying the item through ApplePay since their address is saved on their profile page which will be discussed later on.

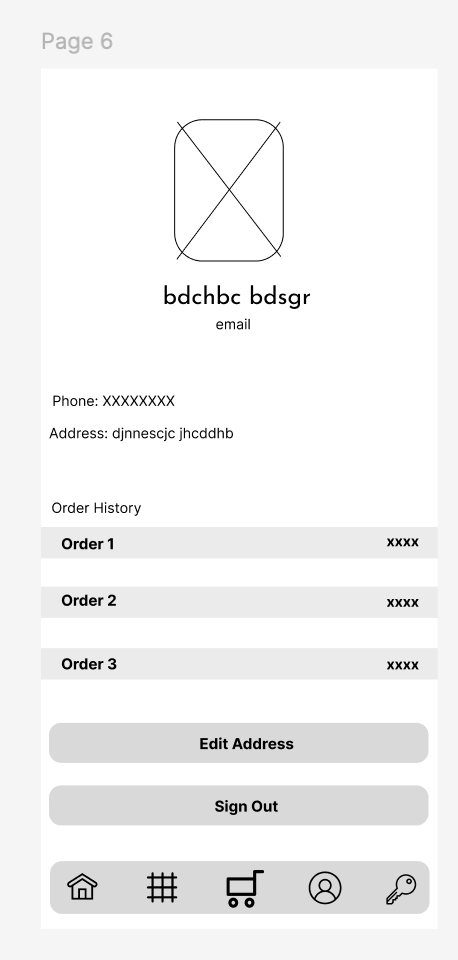
A screenshot of a phone

Description automatically generatedA screen shot of a phone

Description automatically generated

## 6. Profile screen:

* This screen contains the details entered in the login page at the top, with the ability for the user to see them at the top.
* This screen also contains the user’s address, and phone number that is their to help them when they need to purchase an item without having to enter their address again.
* It also contains the history of their past orders.
* The user can interact with this screen by clicking on edit address and they can edit their current address if they wish to change it. They can also sign out of the page by clicking on sign out. They can sign back in by going on to the login page and adding their information.

A screenshot of a phone

Description automatically generated

# **Implementation Details**

The building of the app was done using swiftUI.

## 1. Adding Items to Cart

* The addItemToCart() method is responsible for adding items to cart, where if an item is available in cart this method uses the quantities dictionary where it will check whether a product by it’s given id is available in the cart.
* If the product is present in cart, its quantity is incremented by 1.
* If it does not exist in the cart, the item will get joined to the products array. Then, the total will be recalculated after the product is added to the cart. The recalculateTotal() method is called to recalculate the total based on the new products present in the cart.

## 2. Edit - Incrementing and Decrementing:

* An Item is incremented using the increment(product:) method which would increase the number of the product where it calls the addToCart(product:) method, which in return would increment the quantity of an already existing product or adds a product if it does not exist.
* Reducing an item quanitity, where this method decreases the quantity of a product in the cart by calling removeFromCart(product:). This would then decrement the quantity of the item.

## 3. Deletion from the Cart

* Through the use of removeFromCart(product:) method, a product can be deleted compelety from cart.
* The product can be removed from the array of present products through its index.
* Therefore the product’s entry that is present in the quanitities dictionary is removed by setting it to nil.
* The total price is the recalculated through the recalculateTotal() method that is called when this happens.

## **4. Saving Data**

* **After editing the quantity, adding the product to cart, and the total price is shown, the** data is saved whenever there is a change in the products or quantities. This is done via the didset property observer or when the total changes.
* **This is implement through the** saveCart(), where it converts the products and quantities into a JSON format. After this, the data is saved to UserDefaults using the cartKey and quantitiesKey.
* The saveTotal() would save the total amount to the UserDefaults as an integer.

## **5. Loading Data**

* When CartManager starts, the **loadCart()** method runs. It gets the saved data for products, quantities, and total from UserDefaults. It would also updates the total price to make sure it’s correct.

## 6. Navigation Links

* NavigationLink are used throughout the project for navigating between screens to provide the user-friendly interface.
* An example of a navigation link is the list of products that is displayed as a grid. Each of the product is clickable. After clicking on a product, the user is then taken to a detailed product view (ProductsDetails), where more information about the product is shown.
* The NavigationLink is used to create this change in screens. Each product card is present inside the NavigationLink, which then navigates to the ProductsDetails view. This would pass the item chosen as a parameter.

## **7. Filtering Products:**

* The app allows the user to filter products based on their type such as Shoes, Tops, Pants, Bag, Jacket. The selectedType state variable is the varible responsible for keeping track of the desired filter by the user.
* When the user clicks on the filter option from the segmented picker, it would update the selectedType state, which in return causes the filtering of the products in the allproducts page.
* The Picker is what was chosen to be used for displaying the filtering options. It would bind the selectedType variable to the selected filter. Then the filteredProducts property is used to filter the productList array based on the type wanted to be viewed.

## 8. Editing Text

* In the **ProfileView,** the user has the ability to edit their address using a TextField. When the Edit Details button is clicked, the screen changes slightly to editing the text. This would allow the user to update their address and phone number.
* The address is stored in a @State variable, and thus any changes that were made when the button is clicked is then updated. When the user presses on save, the content is updated.
* If the user pressed on Edit Details again, it would then switch to editing the content, keeping the display of the current address. The text of the button itself changes according to the user if he/she is editing or viewing the details.

# **Testing and Validation**

The application was tested using the following approaches in Swift with Xcode:

## Unit Testing:

I did unit tests for critical functions like adding, editing, and deleting items in the cart to ensure they work correctly.

For example, I checked if adding several items multiple times would update the cart correctly and if deleting items would reduce the total count. In addition to adding one item several times to see if the quantity gets incremented. This was done on Xcode.

## User Testing:

I manually tested the app by putting myself as the user and preforming these actions such as

* + - Adding one of the items to the cart
    - Editing the item’s quantity
    - Deleting an item
    - Checking if the cart would update and change the total price accordingly.
    - In addition, testing the navigation links between screens to ensure that the data present in the cart items and user profile are there and the app switches between screens smoothly.

# **Challenges and Solutions**

**Challenge**:

One major challenge was **saving and keeping the cart items** when the user leaves or closes the app. Initially, the items in the cart would disappear because the data wasn’t being saved. This issue kept appearing unfortunetly, which is why I had to seek methods to solve it.

**Solution**:

* To solve this issue, I used **UserDefaults** to save and load the cart data, where before the user leaves the app, the cart data is stored in UserDefaults.
* In addition, when the app opens again, the data is loaded back into the cart.
* In addition, I resolved the issue through learning how to save and load data using JSONEncoder and JSONDecoder.

# **Future Enhancements**

For future work, if I did this app again, I would want to

1. **Add notifications** section to notify users if they leave items in their cart without checking out. This will help remind users to complete purchases they want to buy.
2. **Dark mode feature where** this would allow users to switch to dark mode for a more comfortable viewing experience, especially at night.

By solving these challenges and improving the app, this would in return enhnace the user experience to make it even better.