**Mobile Application Development (3701ICT/7421ICT)**

Trimester 1, 2024

Assignment 2

**Fake Store**

Milestone 2

Student Name:Maisi Hao

Student ID:s5049158

Student Email:maisi.hao@griffithuni.edu.au

Course Code: [3701ICT or 7421ICT]

Lab:NA

Lab Tutor:

Table of Contents

[Project Overview (Optional) 3](#_Toc165465033)

[GitHub Repository 3](#_Toc165465034)

[Screenshots 3](#_Toc165465035)

[Source Code Snippets 3](#_Toc165465036)

[Reflection (Optional) 3](#_Toc165465037)

[Conclusion (Optional) 3](#_Toc165465038)

## Project Overview (Optional)

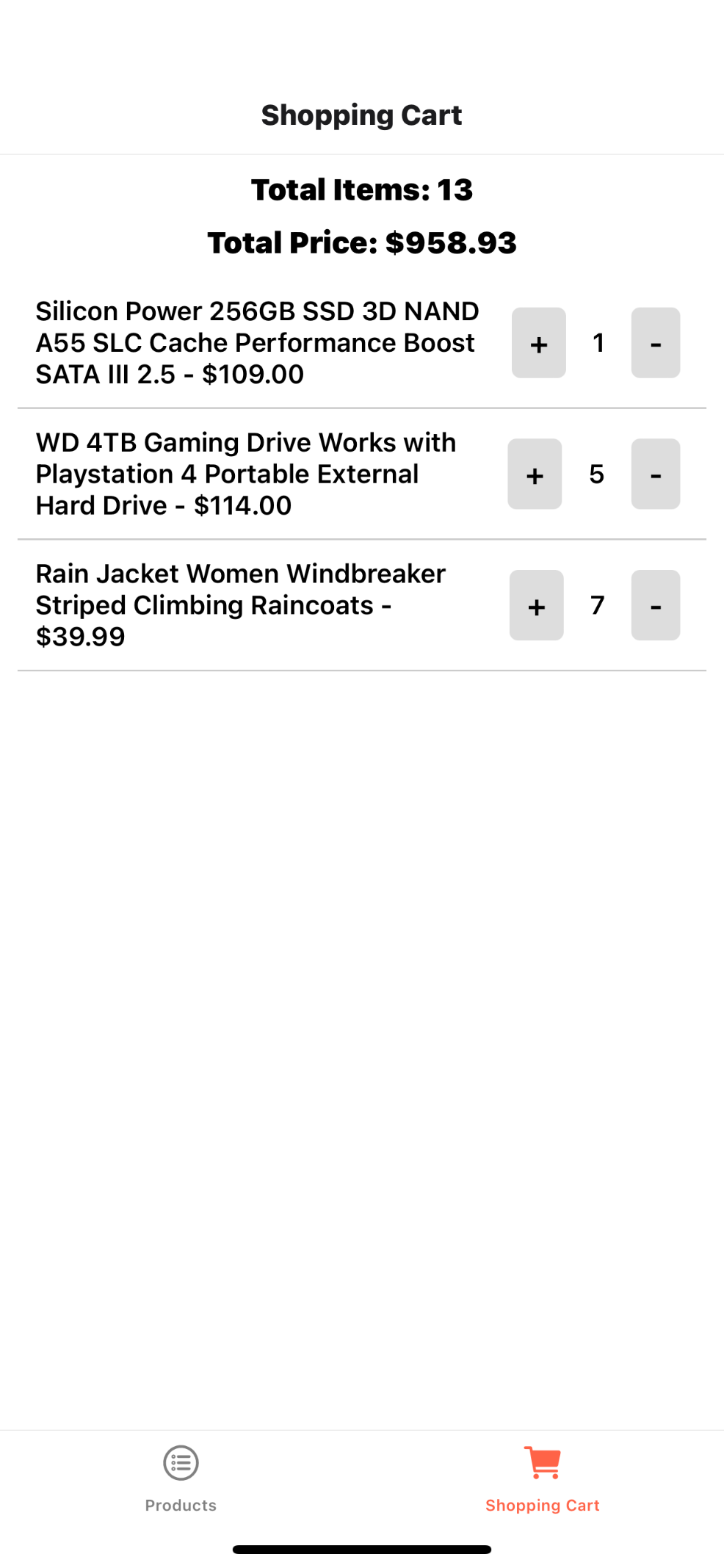
Begin with a concise introduction to the project. Detail its objectives and your approach to achieving the goals of this milestone.

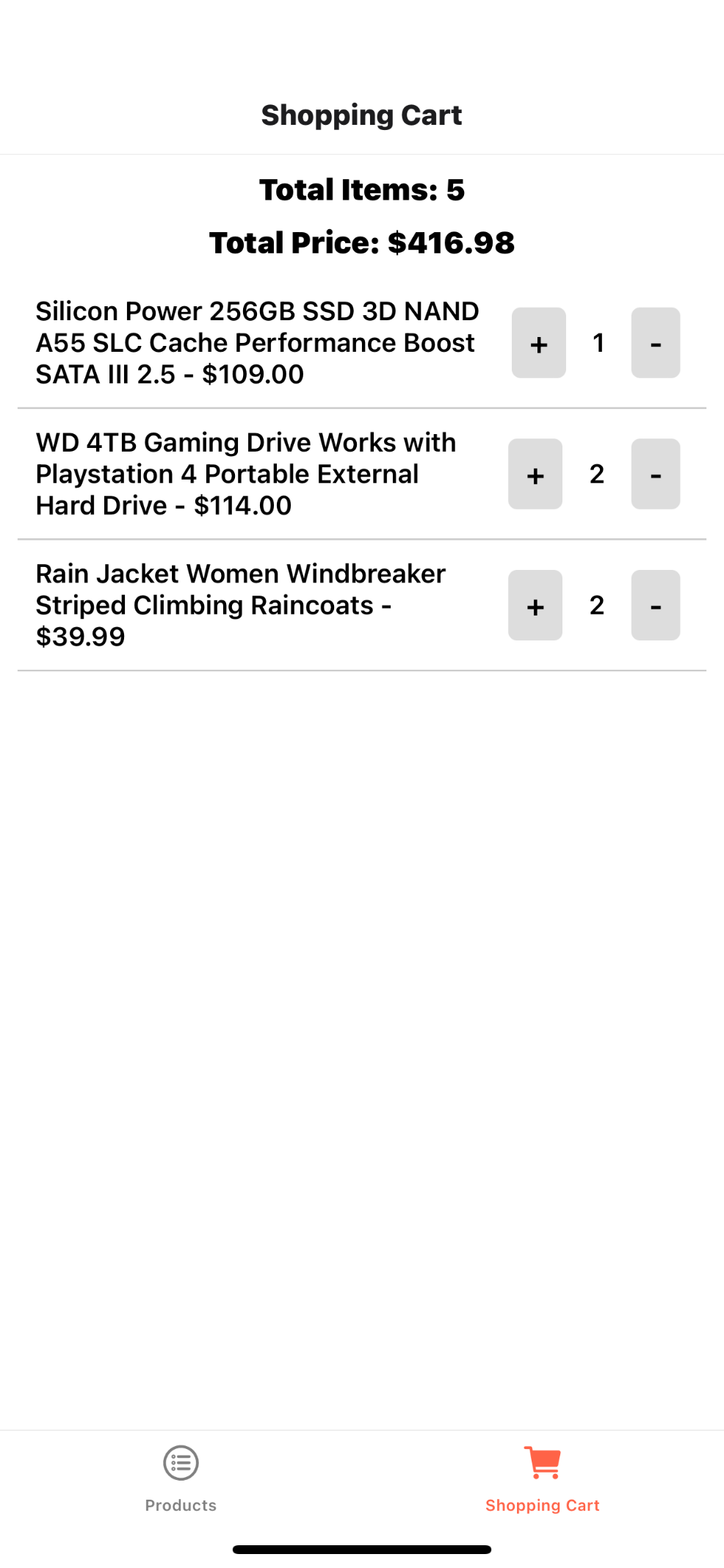
## GitHub Repository

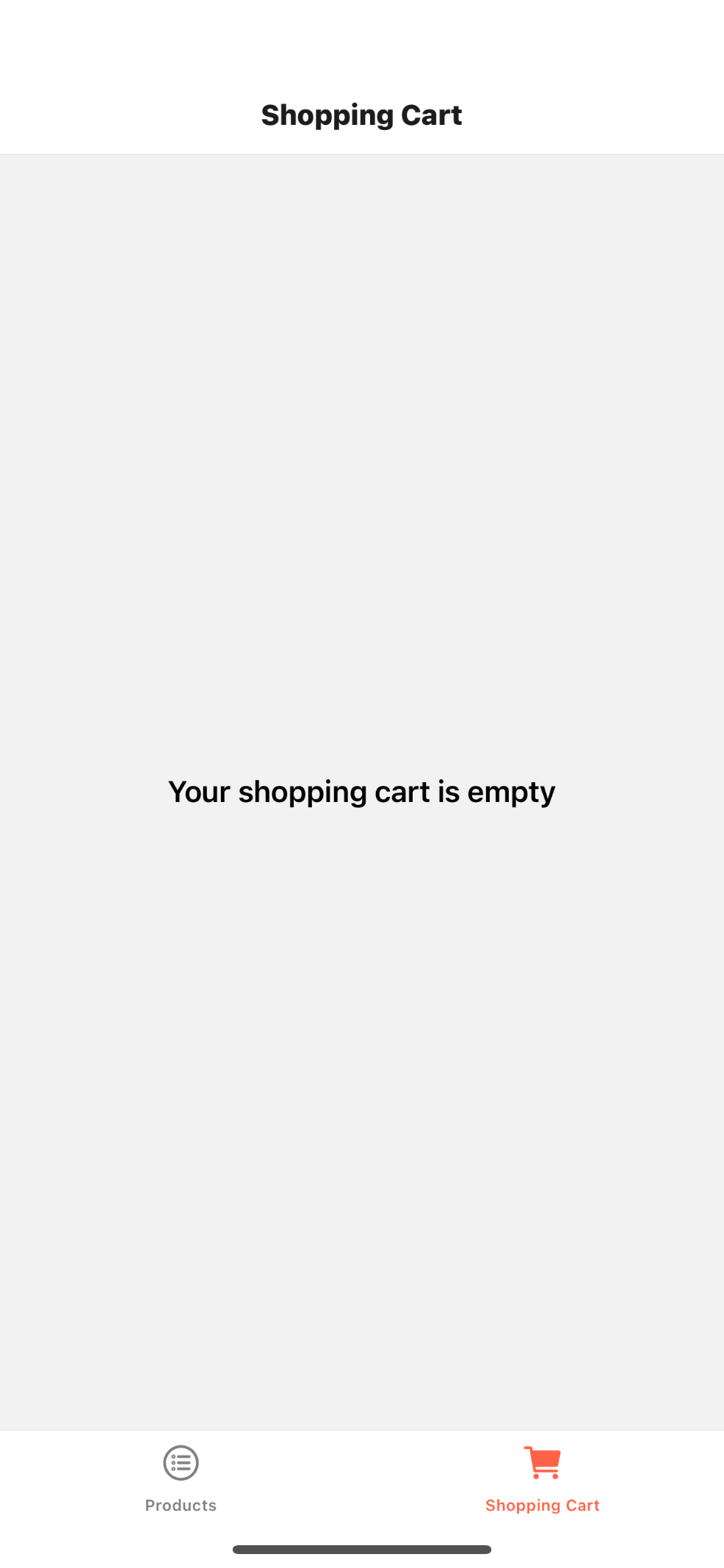
https://github.com/929583249/7421ICT\_Assignment2\_Milestone2.git

## Screenshots

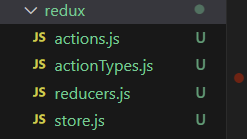








## Source Code Snippets



In my React Native application, I use Redux to manage the shopping cart functionality centrally. This setup involves handling the state related to shopping cart operations such as adding items, removing them, and adjusting their quantities. Let me explain how each part of my Redux implementation works and contributes to the app's functionality:

Store

I set up the Redux store as the central repository for all the state managed across my application. It's created with createStore from Redux, which takes a reducer—in my case, the cartReducer. This store allows the state to be accessible throughout my app via the Provider component, which wraps my app's component tree.

javascript

Copy code

const store = createStore(cartReducer);

Reducers

A reducer in Redux is a function that decides how the state should change in response to actions sent to the store. My cartReducer manages the state of the shopping cart. It handles actions such as adding items to the cart, removing items, and adjusting their quantities.

Here's what the state managed by cartReducer includes:

items: An array that stores each product added to the cart as an object.

totalQuantity: The total number of items in the cart.

totalPrice: The total price of all the items in the cart.

The reducer listens for actions like ADD\_TO\_CART, REMOVE\_FROM\_CART, INCREASE\_QUANTITY, and DECREASE\_QUANTITY and adjusts the cart's state based on these actions.

javascript

Copy code

const cartReducer = (state = initialState, action) => {

switch (action.type) {

case ADD\_TO\_CART:

// logic to add item or increase quantity

case REMOVE\_FROM\_CART:

// logic to remove item

case INCREASE\_QUANTITY:

// logic to increase item quantity

case DECREASE\_QUANTITY:

// logic to decrease item quantity

default:

return state;

}

};

Actions

Actions in Redux are payloads of information that send data from my application to the store. They are the only source of information for the store. I send them to the store using store.dispatch(). The actions I've defined, such as addToCart, removeFromCart, increaseQuantity, and decreaseQuantity, help modify the state based on user interactions.

For instance, my addToCart action creates a payload with a type of ADD\_TO\_CART and includes the product object as its payload. This action instructs the reducer on how to update the state: either by adding a new product to the cart or increasing the quantity of an existing product.

javascript

Copy code

export const addToCart = (product) => ({

type: ADD\_TO\_CART,

payload: product

});

How Redux Works in My App

Action Dispatch: When I interact with the UI (e.g., when I press "Add to Cart"), my app dispatches an action.

Reducer Processing: The reducer receives the current state and the dispatched action, decides how the state should change, and returns the new state.

Update UI: Through React Redux, using the useSelector hook, my components subscribe to Redux state updates. When the state changes, my components automatically re-render with the new data.

## Reflection (Optional)

Reflect on your experiences throughout the learning process and any feedback you may have about this particular milestone.

## Conclusion (Optional)

Conclude with a summary of what you have achieved in this milestone and any additional remarks.