Mobile Application Development (7421ICT)

Trimester 1, 2024

Assignment 2

Fake Store

Milestone 2

Student Name: Raghav Kohli

Student ID: s5326153

Student Email: raghav.kohli@griffithuni.edu.au

Course Code: 7421ICT

Lab: Gold Coast, Tuesday 6pm-8pm

Lab Tutor: Larry Wen

Table of Contents

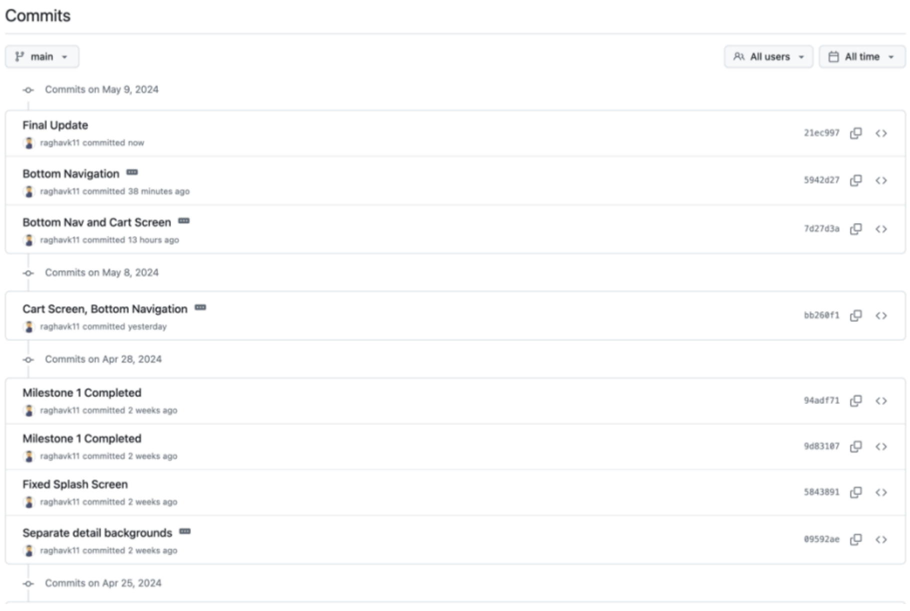
Project Overview

In this milestone of the "Fake Store" mobile application project, the primary objective is to enhance the application's state management and shopping cart functionality using Redux with reduxjs/toolkit. The focus is on providing a seamless user experience by implementing a bottom tab navigation, allowing users to easily switch between the product categories and the shopping cart screens. The shopping cart screen will display the cart's contents, enabling users to adjust product quantities and view the total item count and cost.

GitHub Repository

Repository URL: https://github.com/raghavk11/Ass2_7421ICT.git

Commit History


Commits

Commits on May 9, 2024

Final Update
raghavk11 committed now
21ec997

Bottom Navigation
raghavk11 committed 38 minutes ago
5942d27

Bottom Nav and Cart Screen
raghavk11 committed 13 hours ago
7d27d3a

Commits on May 8, 2024

Cart Screen, Bottom Navigation
raghavk11 committed yesterday
bb260f1

Commits on Apr 28, 2024

Milestone 1 Completed
raghavk11 committed 2 weeks ago
94adf71

Milestone 1 Completed
raghavk11 committed 2 weeks ago
9d83107

Fixed Splash Screen
raghavk11 committed 2 weeks ago
5843891

Separate detail backgrounds
raghavk11 committed 2 weeks ago
09592ae

Commits on Apr 25, 2024

State Management

For this milestone, Redux with reduxjs/toolkit has been utilized to manage the state of the shopping cart. The shopping cart slice has been created to handle the necessary actions and reducers, ensuring efficient state management and seamless updates to the user interface.

Bottom Tab Navigation

A bottom tab navigation has been implemented with two tabs: Products and Shopping Cart. The Products tab links back to the original product category screen, while the Shopping Cart tab opens a new screen displaying the contents of the shopping cart. A badge on the Shopping Cart tab indicates the total number of items in the cart, leveraging Redux state for accurate count management and display.

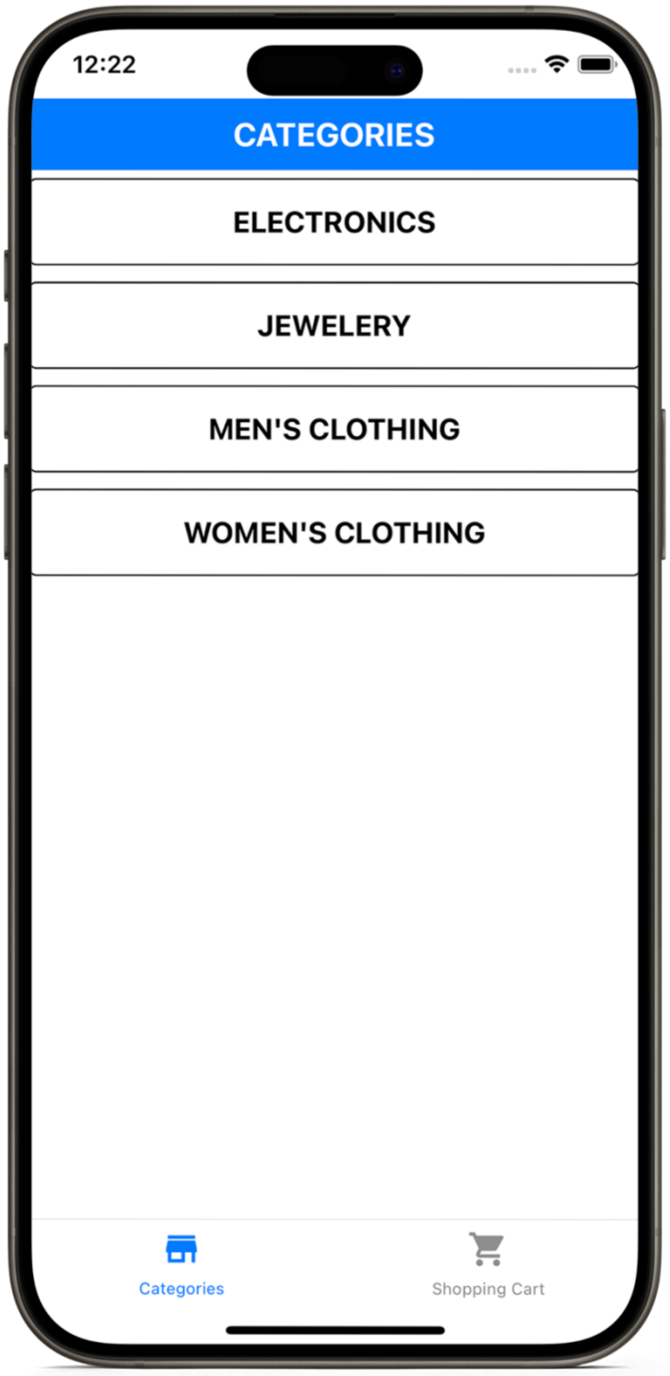
Shopping Cart Screen

The shopping cart screen has been designed to handle various scenarios:

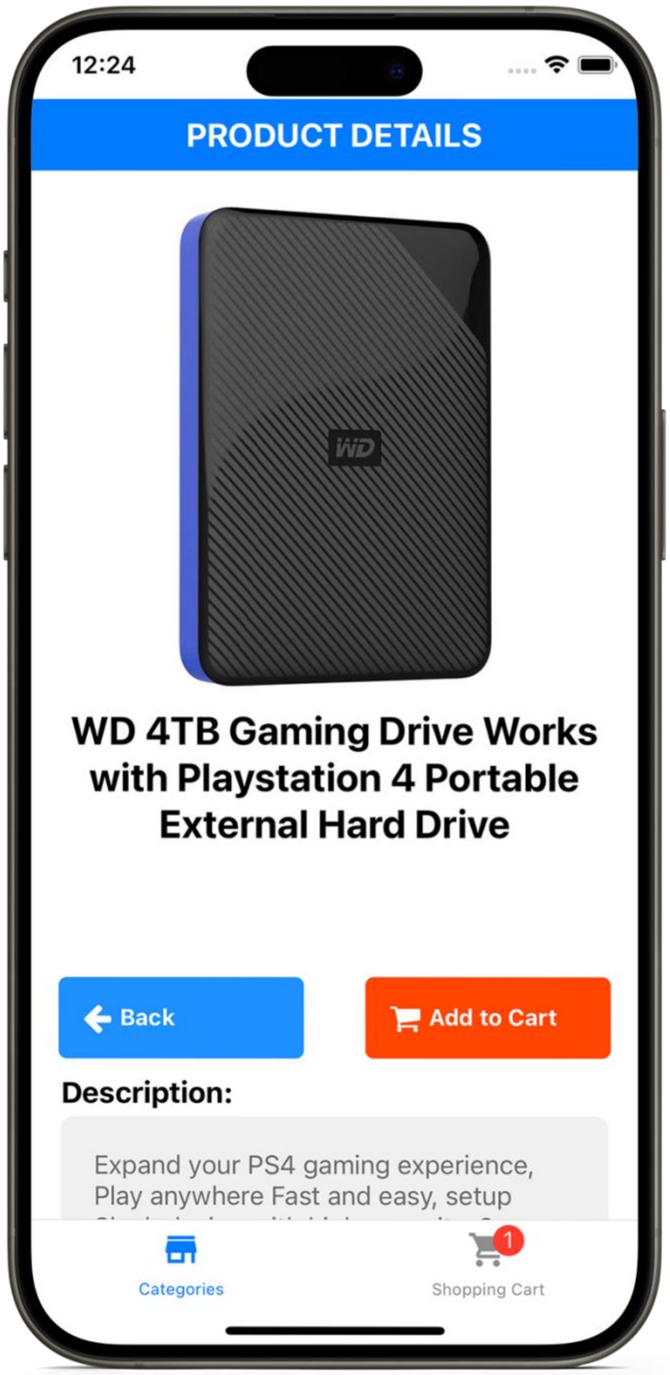
- When the cart is empty, a message "Your shopping cart is empty" is displayed.
- When items are present, a FlatList is used to display the products in the cart.
- Increase and decrease buttons are provided for each product, allowing users to adjust quantities directly from the cart screen. These adjustments update the Redux state and are immediately reflected in the user interface.
- If a product's quantity drops to zero, it is automatically removed from the cart.
- When the cart is not empty, the total number of items and the total cost of the products are displayed in the top section of the screen.

Screenshots

Categories Screen with Product Categories and Tab Navigation:



Product Detail Screen with One Product Added to Shopping Cart:

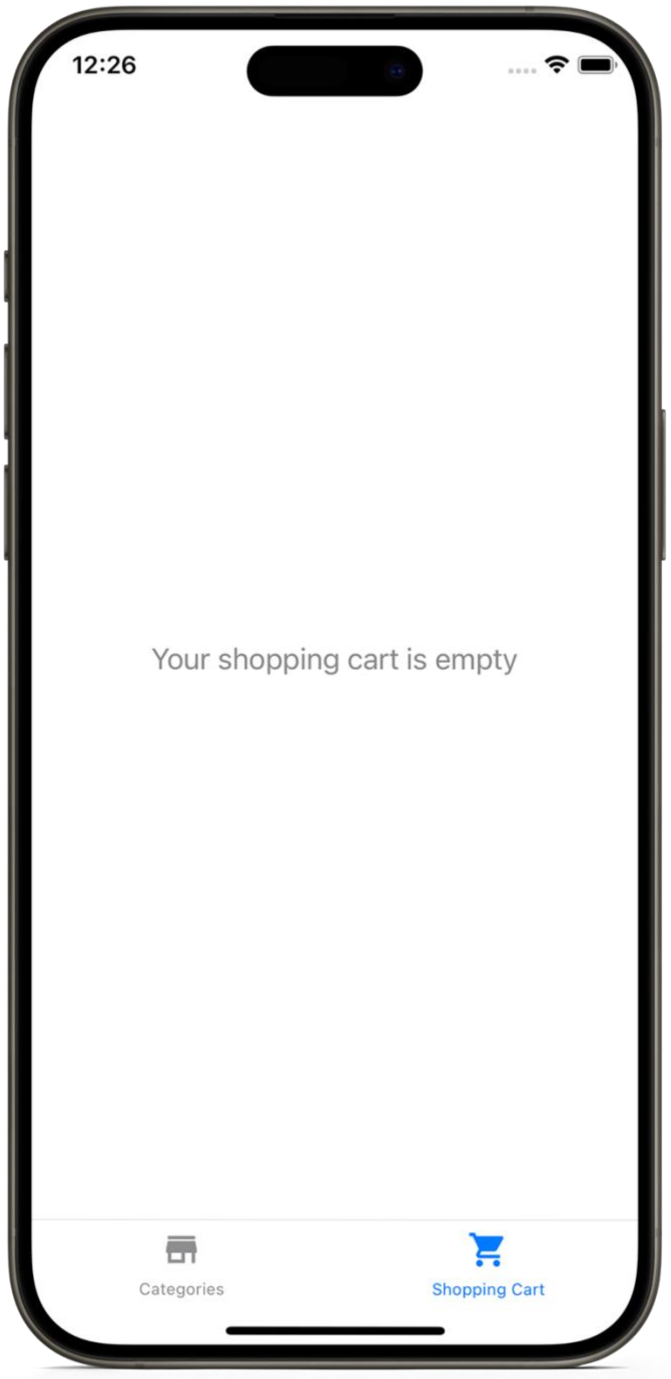


WD 4TB Gaming Drive Works with Playstation 4 Portable External Hard Drive

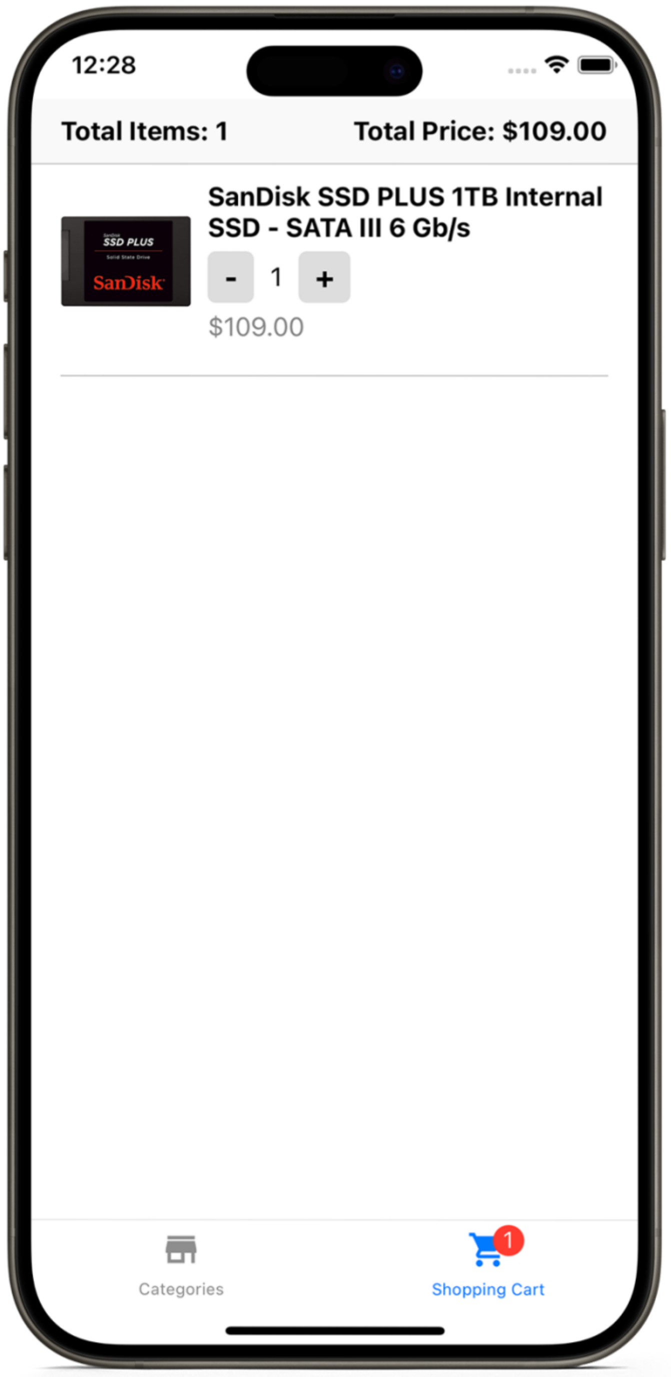
Description:

Expand your PS4 gaming experience, Play anywhere Fast and easy, setup

Shopping Cart Screen - Empty Cart:



Shopping Cart Screen - One Product in Cart:

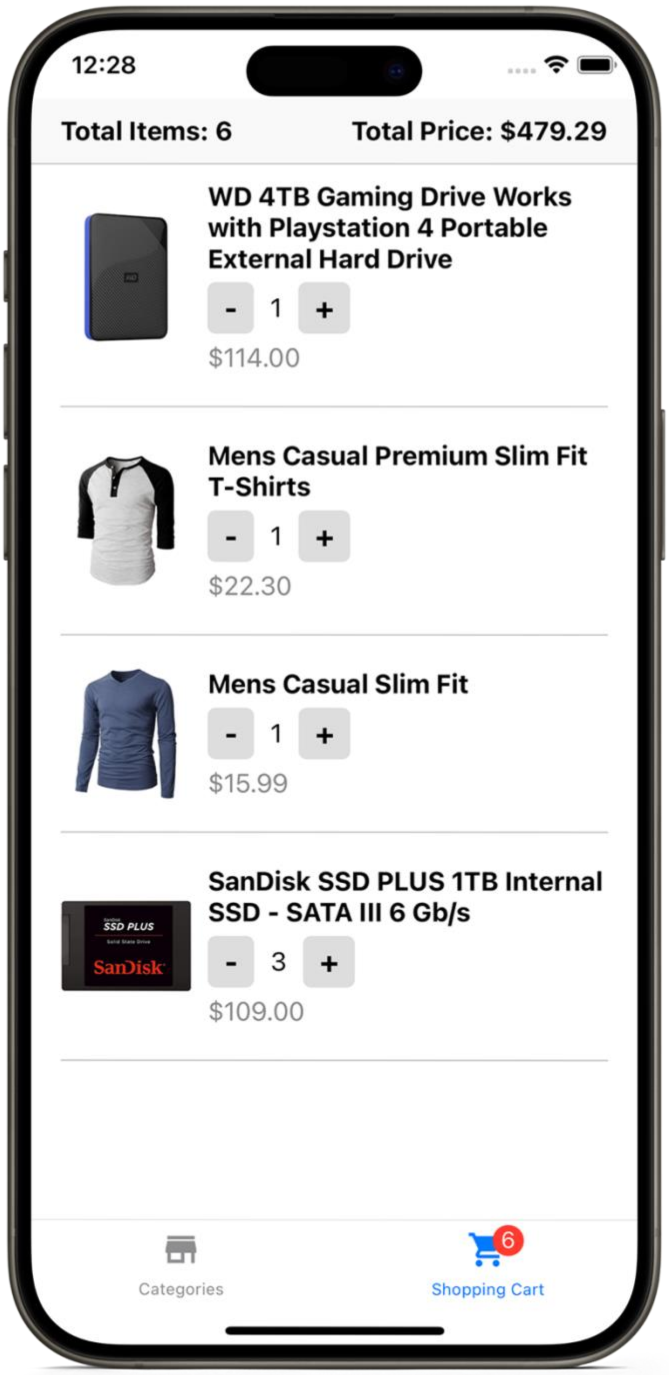

12:28

Total Items: 1 Total Price: $109.00

SanDisk SSD PLUS 1TB Internal SSD - SATA III 6 Gb/s

$109.00

Shopping Cart Screen - Multiple Products in Cart:



12:28

Total Items: 6 Total Price: $479.29

WD 4TB Gaming Drive Works with Playstation 4 Portable External Hard Drive
- 1 +
$114.00

Mens Casual Premium Slim Fit T-Shirts
- 1 +
$22.30

Mens Casual Slim Fit
- 1 +
$15.99

SanDisk SSD PLUS 1TB Internal SSD - SATA III 6 Gb/s
- 3 +
$109.00

Source Code Snippets

Redux Shopping Cart Slice (CartSlice.js):

```
import { createSlice } from '@reduxjs/toolkit';

const cartSlice = createSlice({
  name: 'cart',
  initialState: {
    items: [],
  },
  reducers: {
    addItem: (state, action) => {
      const item = action.payload;
      const existingItem = state.items.find((i) => i.id === item.id);
      if (existingItem) {
        existingItem.quantity += 1;
      } else {
        state.items.push({ ...item, quantity: 1 });
      }
    },
    removeItem: (state, action) => {
      const itemId = action.payload;
      state.items = state.items.filter((item) => item.id !== itemId);
    },
    incrementQuantity: (state, action) => {
      const itemId = action.payload;
      const item = state.items.find((i) => i.id === itemId);
      if (item) {
        item.quantity += 1;
      }
    },
    decrementQuantity: (state, action) => {
      const itemId = action.payload;
      const item = state.items.find((i) => i.id === itemId);
      if (item) {
        if (item.quantity === 1) {
          state.items = state.items.filter((i) => i.id !== itemId);
        } else {
          item.quantity -= 1;
        }
      }
    },
  },
});

export const { addItem, removeItem, incrementQuantity, decrementQuantity } = cartSlice.actions;

export const selectCartItems = (state) => state.cart.items;

export const selectCartItemsCount = (state) => {
  return state.cart.items.reduce((total, item) => total + item.quantity, 0);
};

export default cartSlice.reducer;
```

This code snippet demonstrates the implementation of the shopping cart slice using Redux Toolkit. Let's break it down:

1. The cartSlice is created using the createSlice function from Redux Toolkit. It has a name of 'cart' and an initial state of an empty items array.
2. Inside the reducers object, four reducer functions are defined:
    - addItem: Adds an item to the cart. If the item already exists, it increments the quantity; otherwise, it adds a new item with a quantity of 1.
    - removeItem: Removes an item from the cart based on its ID.
    - incrementQuantity: Increments the quantity of an item in the cart by 1.
    - decrementQuantity: Decrements the quantity of an item in the cart by 1. If the quantity becomes 0, the item is removed from the cart.
3. The action creators (addItem, removeItem, incrementQuantity, decrementQuantity) are exported using the cartSlice.actions object. These action creators can be used to dispatch actions to modify the cart state.
4. Two selector functions are exported:
    - selectCartItems: Selects the items array from the cart state.
    - selectCartItemsCount: Calculates the total count of items in the cart by summing up the quantities of all items.

5. Finally, the cartSlice.reducer is exported as the default export, which will be used to handle state updates related to the cart in the Redux store.

This code snippet showcases the key aspects of the shopping cart slice, including the initial state, reducer functions, action creators, and selector functions. It provides a clear understanding of how the cart state is managed using Redux Toolkit.

Reflection

Throughout the development process of Milestone 2, I have gained valuable experience in integrating Redux with reduxjs/toolkit for state management in a React Native application. Implementing the bottom tab navigation and the shopping cart functionality has deepened my understanding of how to structure and manage complex application states effectively.

One of the challenges I encountered was ensuring that the shopping cart state was accurately updated and reflected in the user interface in real-time. By leveraging Redux, I was able to overcome this challenge and provide a seamless user experience.

This milestone has also reinforced the importance of proper code organization and modularization when working with Redux. Breaking down the shopping cart functionality into separate actions and reducers has made the code more maintainable and easier to reason about.

Conclusion

In conclusion, Milestone 2 of the "Fake Store" project has successfully integrated Redux with reduxjs/toolkit for efficient state management, particularly focusing on the shopping cart functionality. The implementation of the bottom tab navigation and the shopping cart screen has enhanced the user experience by providing intuitive navigation and real-time updates to the cart's contents.

The project has demonstrated proficiency in utilizing Redux for managing complex application states, handling user interactions, and ensuring data consistency across the application. The skills and knowledge gained from this milestone will be invaluable for future development efforts in building robust and scalable mobile applications.