CMPS 312 Mobile App Development
QuickMart App
Assignment 2
Deadline - Sunday October 22, 2024 @11:59PM

Instructions:

In this assignment, you will design and implement *QuickMart*, a grocery shopping app, using Flutter. The app will have five key screens:

1. **Product Screen**: This is the home screen that allows users to browse and search for products and add items to the shopping cart or favorites list.
2. **Product Details Screen:** Users can view detailed information about a specific product, such as its name, price, rating, and description. From this screen, users can also add the product to the cart or favorites.
3. **Cart Screen**: Users can update the quantity of cart items or remove items from the cart.
4. **Favorites Screen**: Users can view all products added to their favorites list.
5. **Shell Screen**: Users can navigate between the Product Screen, Cart Screen, and Favorites Screen. It is the screen that holds the bottom navigation bar which is common for all screens.

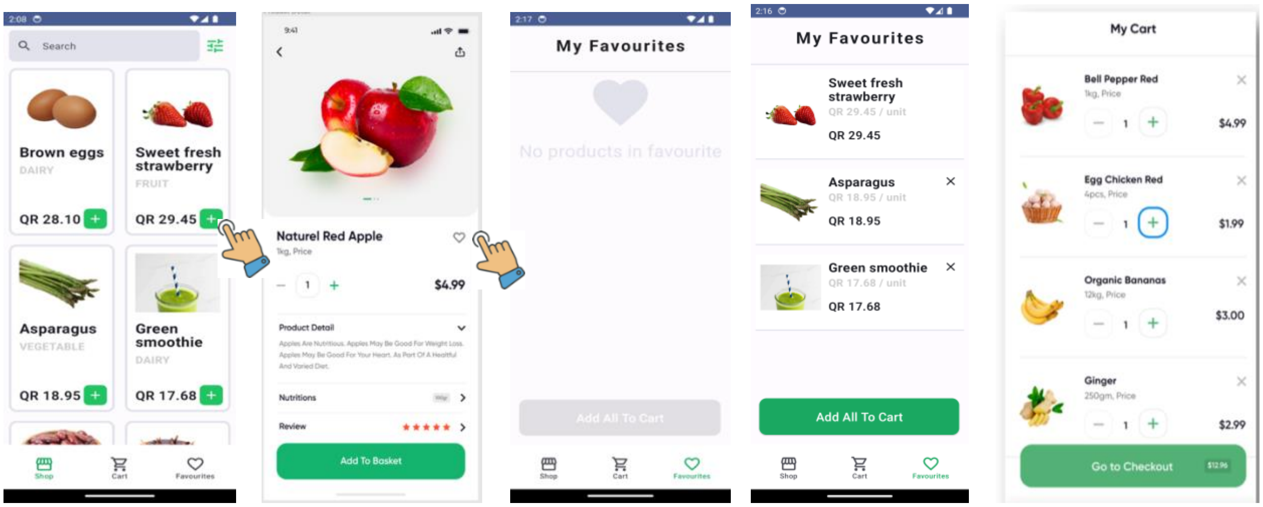


Figure 1 Quick Mart App

Provided Materials

You will be provided with a **base Flutter project called QuickMart** that includes the following:

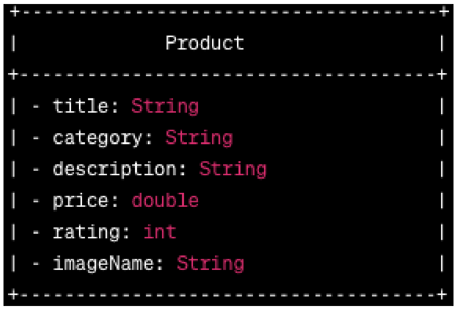
- All necessary dependencies such as go_router and flutter_riverpod already configured in the pubspec.yaml file , all you need to do is get the packages.
- Pre-configured project structure with directories such as assets, data, images, models, providers, screens, and routes.
- JSON files (**products.json** and **product-categories.json**) added to the **assets/data folder**.
- All the required images can be found under assets/images folder

Note: The major structure of the project has been provided to you, but it is your responsibility to handle and implement any missing elements or files.

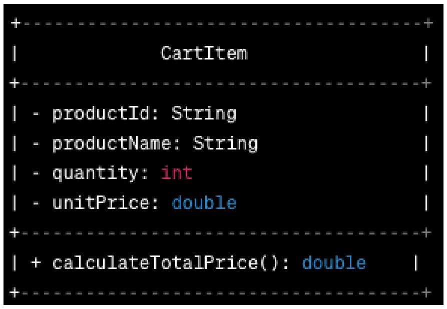
A. Data Model

Create two model classes:

- **Product**: Represents a product with attributes listed below.
- **CartItem**: Represents an item in the cart with attributes listed below.



```
+-----------------------------------+
|              Product              |
+-----------------------------------+
| - title: String                   |
| - category: String                |
| - description: String             |
| - price: double                   |
| - rating: int                     |
| - imageName: String               |
+-----------------------------------+
```



```
+-----------------------------------+
|              CartItem             |
+-----------------------------------+
| - productId: String               |
| - productName: String             |
| - quantity: int                   |
| - unitPrice: double               |
+-----------------------------------+
| + calculateTotalPrice(): double  |
+-----------------------------------+
```

B. Data Providers (State Management)

Manage the application state using the **Riverpod** library. Define notifiers/providers for managing product data, cart data, and favorites data.

1. **ProductProvider**:
   - Load product data from **products.json** into a list.
   - Provide functions for filtering products by name or category.
2. **CartProvider**:
   - Manage a list of items in the cart.
   - Implement functionality for adding products to the cart, updating item quantities, and calculating the total price of the cart.
   - Ensure that changes in the state (e.g., adding items to the cart) are reflected across the relevant screens.
3. **FavoritesProvider**:
   - Manage a list of favorite products.
   - Implement functionality for adding/removing products from the favorites list.

C. ProductScreen

- Design a screen that displays a list of products using Grid as shown in Figure 2. Each product should be displayed in a product card with the ability to add the product to the cart or to favorites by clicking a heart icon.
- Implement a search bar and a dropdown menu for filtering products by name or category.
- Ensure the screen updates the state when items are added to the cart or marked as favorites.
- When a product is clicked, navigate the user to the **ProductDetails Screen**.

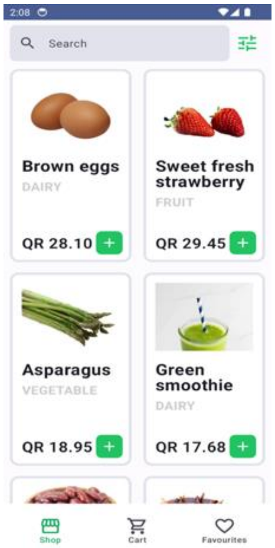


Figure 2 Product Screen

D. ProductDetailsScreen

- Design a screen that shows detailed information about the selected product, such as its name, price, category, and a description as shown in figure 3.
- Provide the ability to add the product to the cart or mark it as a favorite using a heart icon. If an product is already in the favorites, the hear should be **filled** green.
- Also, if a product was already added to the cart, the quantity in the details should reflect the same in the cart.
- Ensure navigation to the **ProductDetails Screen** works from the **ProductScreen** . Pressing the back button should return you to the previous screen.

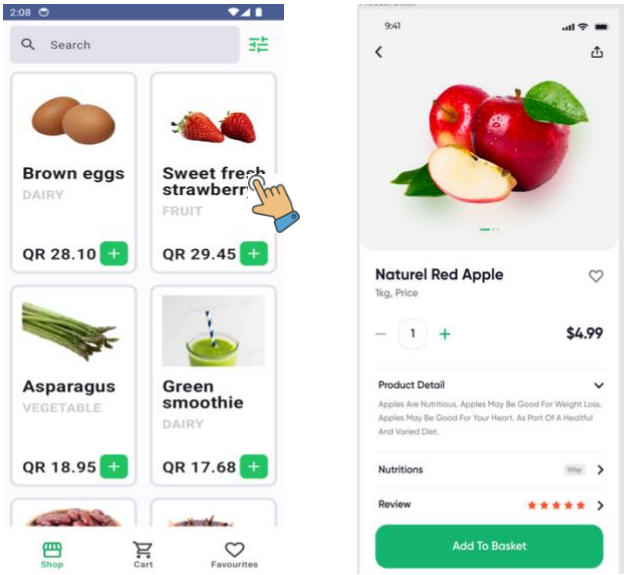


Figure 3 Details Screen

E. CartScreen

- Design a screen that shows all items in the cart, including the name, quantity, and total price of each item.
- Allow users to update the quantity of items or remove them from the cart.
- Show the **total price** of all items in the cart.
- The user should be able to add products to the cart from both the Product Screen and the Product Details Screen.

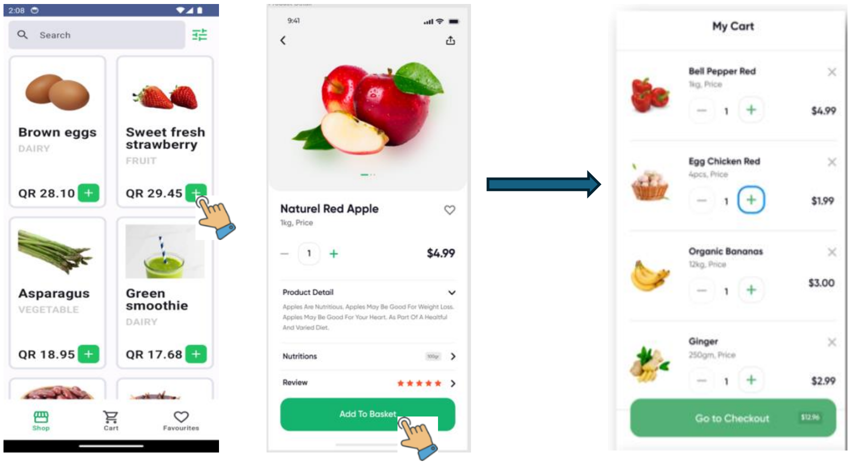


Figure 4 Cart Screen

F. FavoritesScreen

- Design a screen that shows all products added to the favorites list.
- Ensure users can view the product details (navigating to the **ProductDetails Screen**) and remove items from the favorites list if needed.

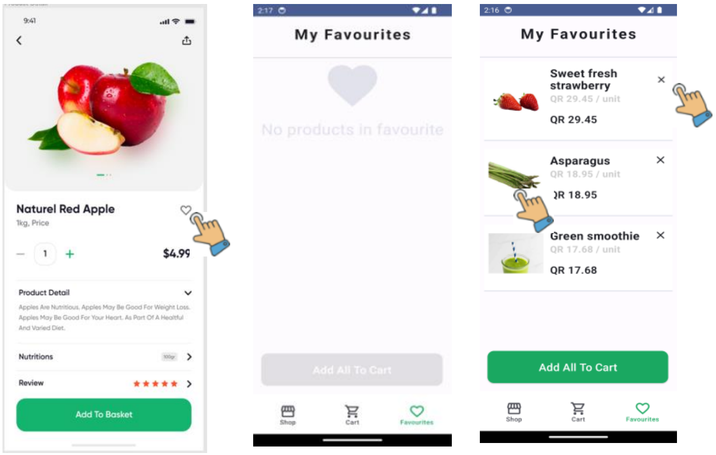
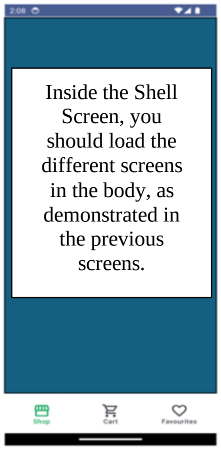


Figure 5 : Favorites Screen

G. ShellScreen

- Implement the main screen with a bottom navigation bar and an App Bar that allows users to switch between the Product Screen, Cart Screen, and Favorites Screen.
- Use GoRouter to handle navigation between the different screens.
- The ProductScreen should be the entry point of the app. Hint : See Lab 7



Inside the Shell Screen, you should load the different screens in the body, as demonstrated in the previous screens.

Deliverables

- Fill out the ***Lab-Testing-Grading-Sheet.docx*** and save it in the `***Assignments/Assignment2***` folder of your repository.
- Your submission should include:
  - o The Quickmart Flutter project code.
  - o Testing sheet that has all the screens.
- Sync your repository to push your work to GitHub.

Note Ensure your code adheres to best practices, is well-commented, and handles exceptions gracefully. Good luck!