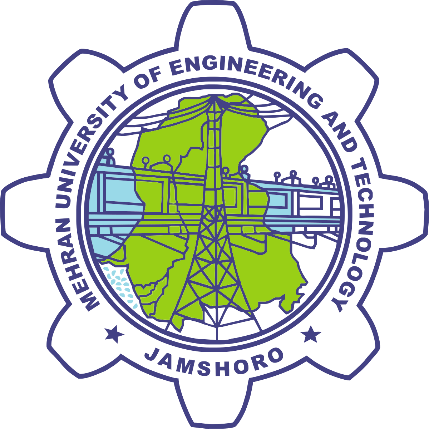
**MEHRAN UNIVERSITY OF ENGINEERING AND TECHNOLOGY**



|  |  |
| --- | --- |
| Roll No | 22SW057 |
| Section | III |
| MAD | CEP |
| Project | Clothing Order App |

## Project Report: Clothing Order App

**1. Real World Problem Identification**

Online shopping is one of the most popular trends today, but many clothing platforms are either too complex, cluttered, or lack essential personalization and order management features.  
Customers often face issues such as:

* Difficulty browsing organized product categories.
* Lack of wishlist or cart management.
* Complicated checkout process.
* Inconsistent user experience across devices.

There was a need for a **simple, user-friendly clothing order app** that provides a smooth shopping experience with responsive design and efficient data storage.

**2. Proposed Solution**

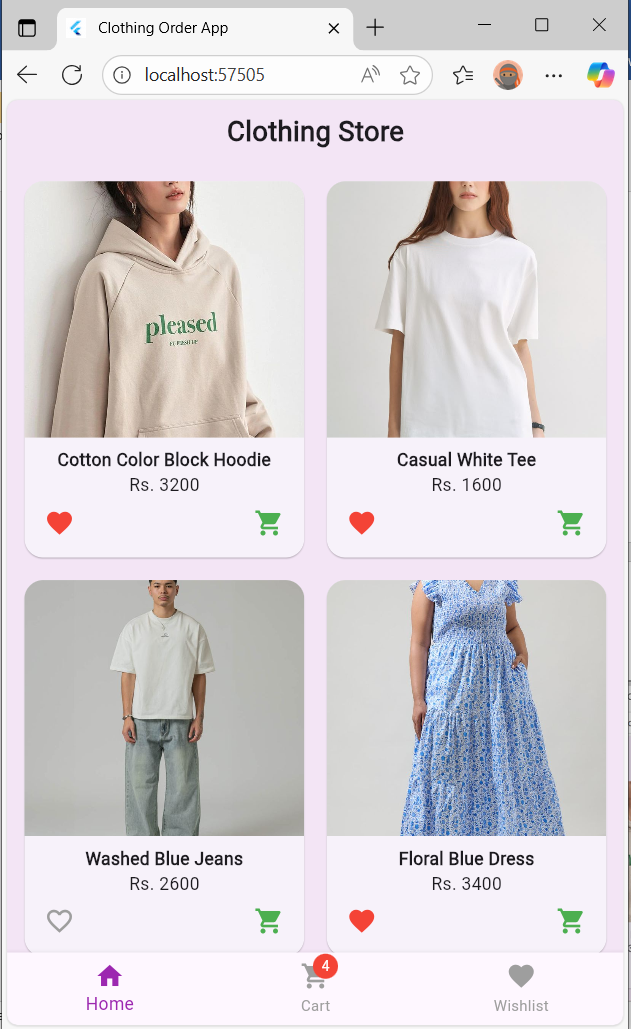
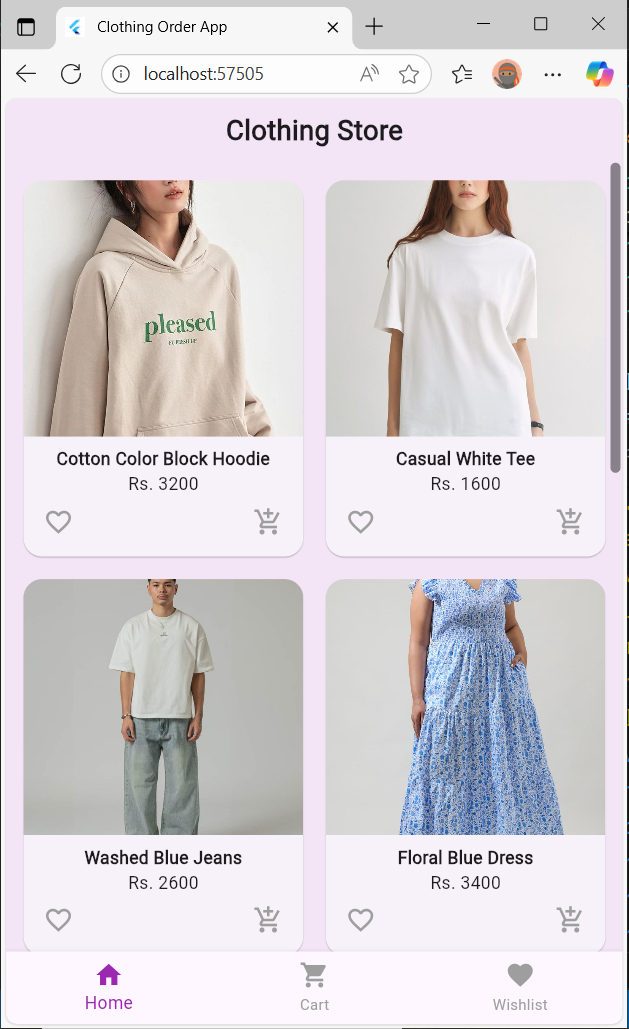
The **Clothing Order App** was developed to solve these problems by providing:

* A clean and minimal shopping interface.
* Features like adding products to **cart** and **wishlist**.
* A working **checkout system** with order tracking.
* **Hive database** integration for local storage (lightweight and offline-friendly).
* Fully **responsive UI** that adjusts to mobile, tablet, and desktop screens.

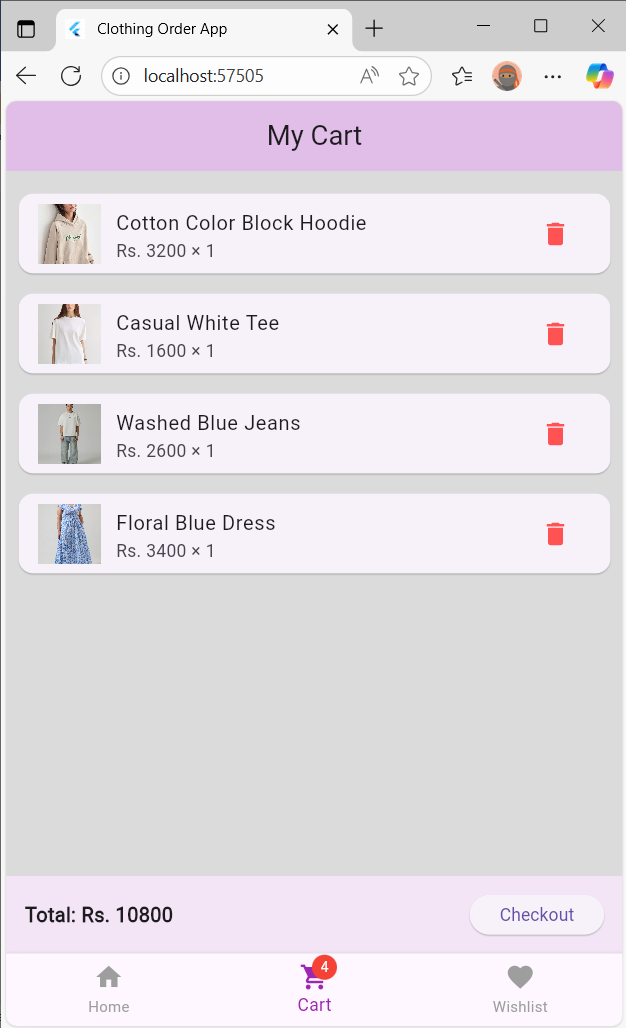
This makes the app ideal for small online clothing stores that want a functional, modern mobile solution.

**3. Responsive User Interfaces (Screenshots)**

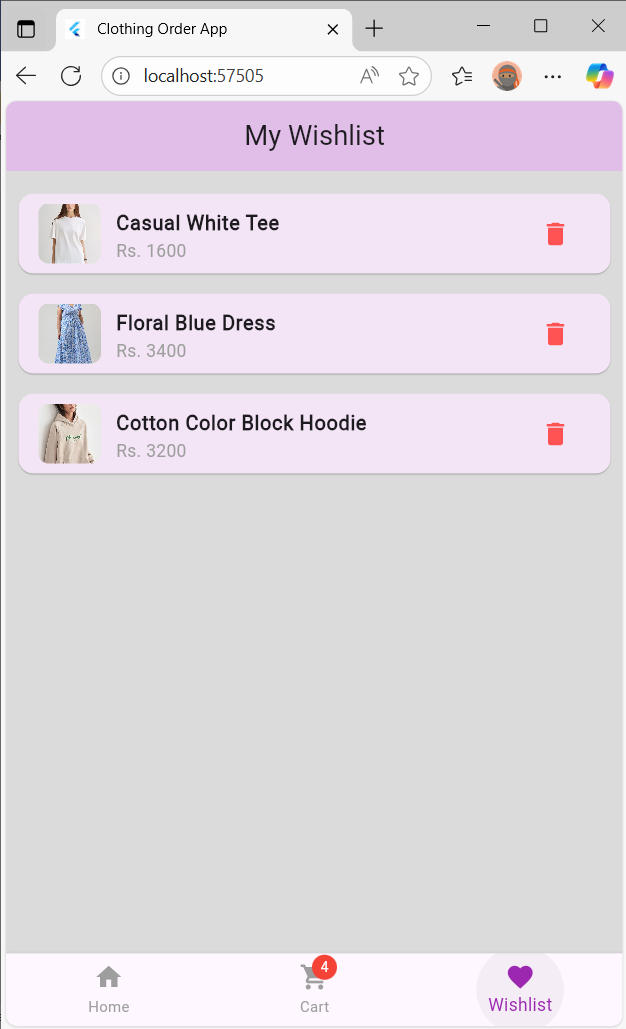
**Home Screen:**



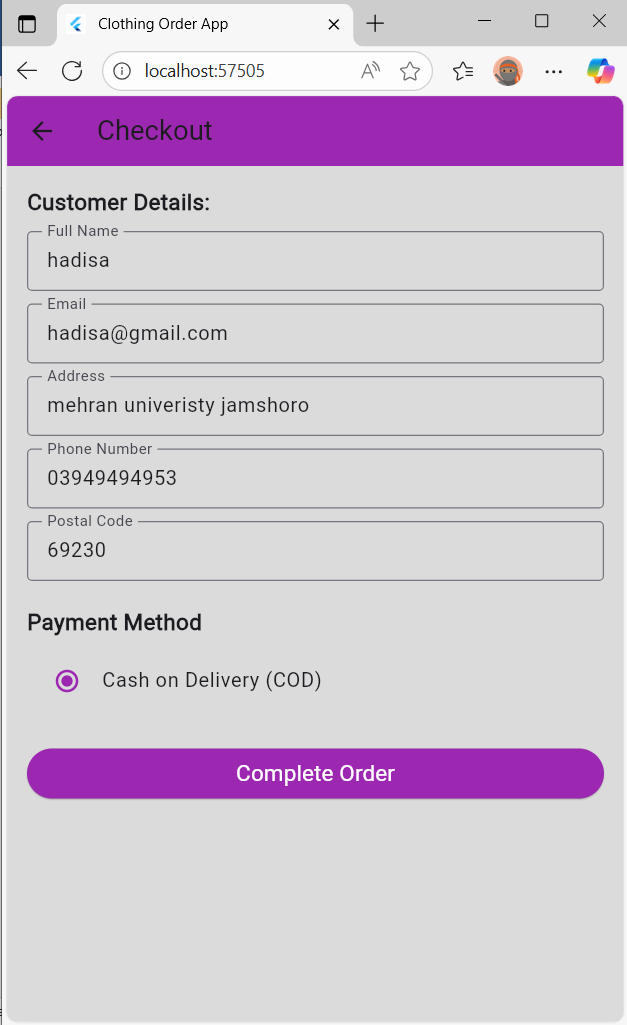
**Cart Screen:**

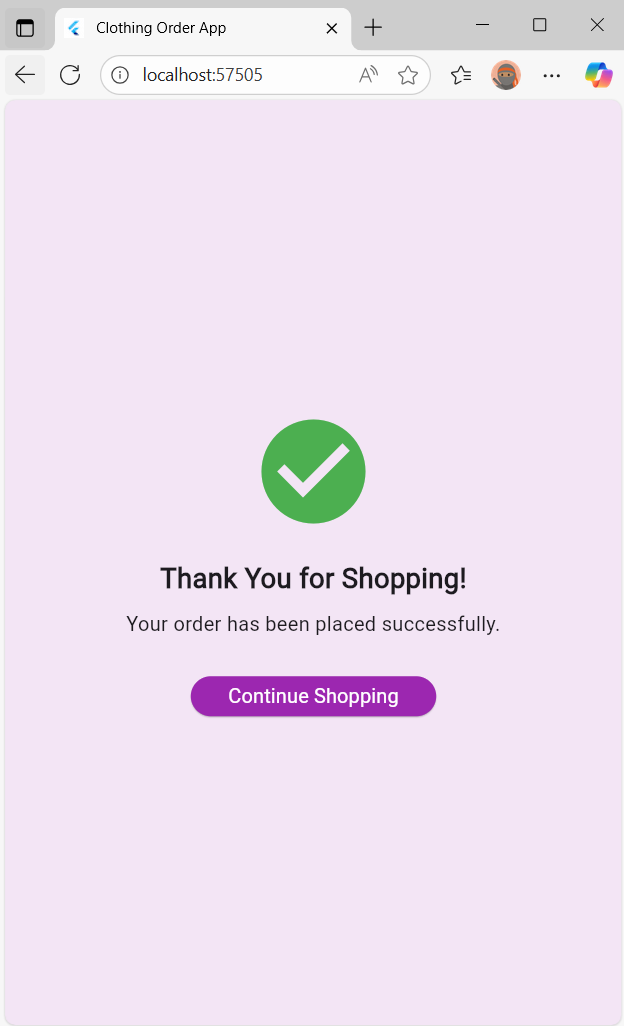


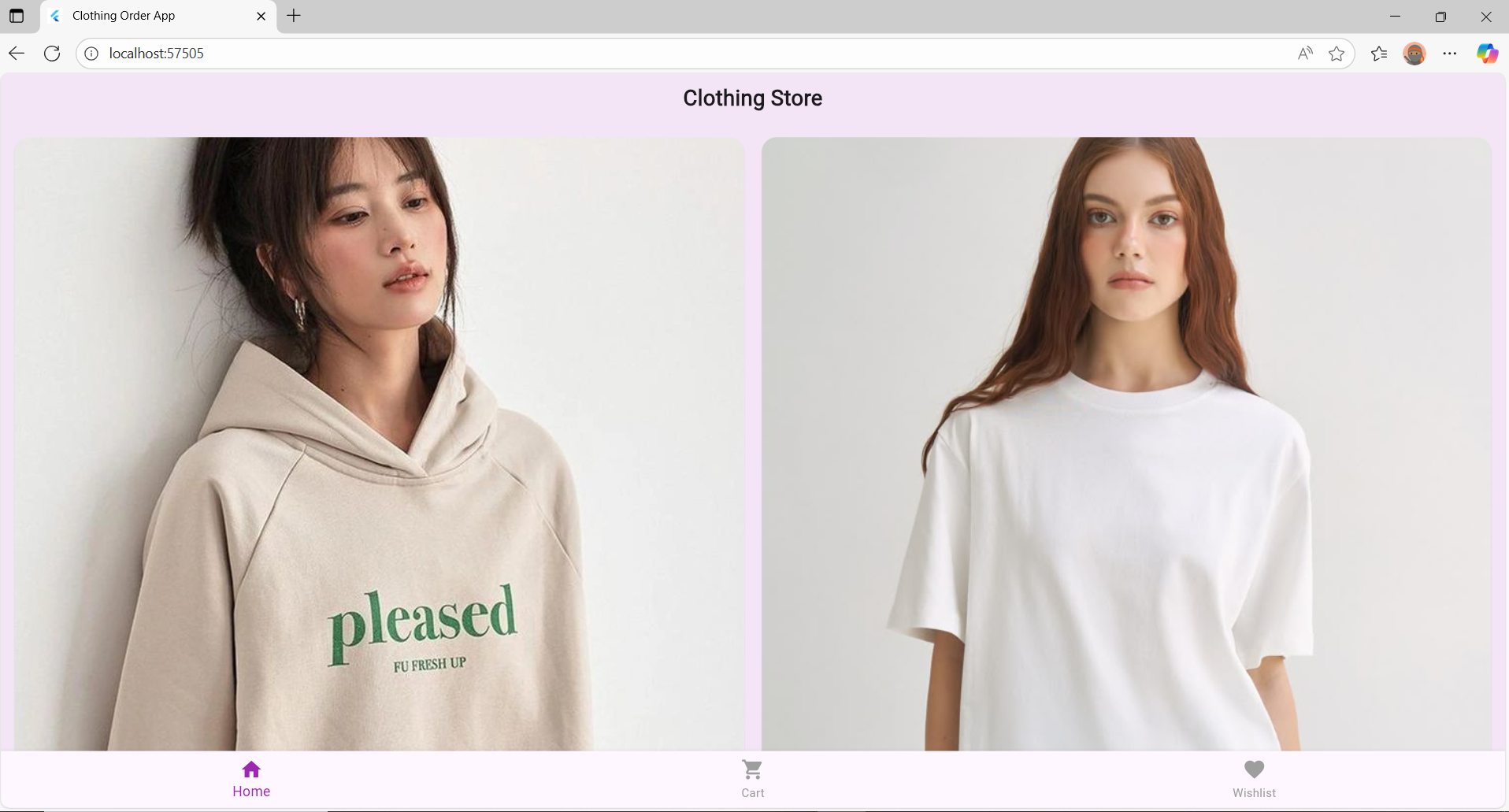
**Wishlist Screen:**

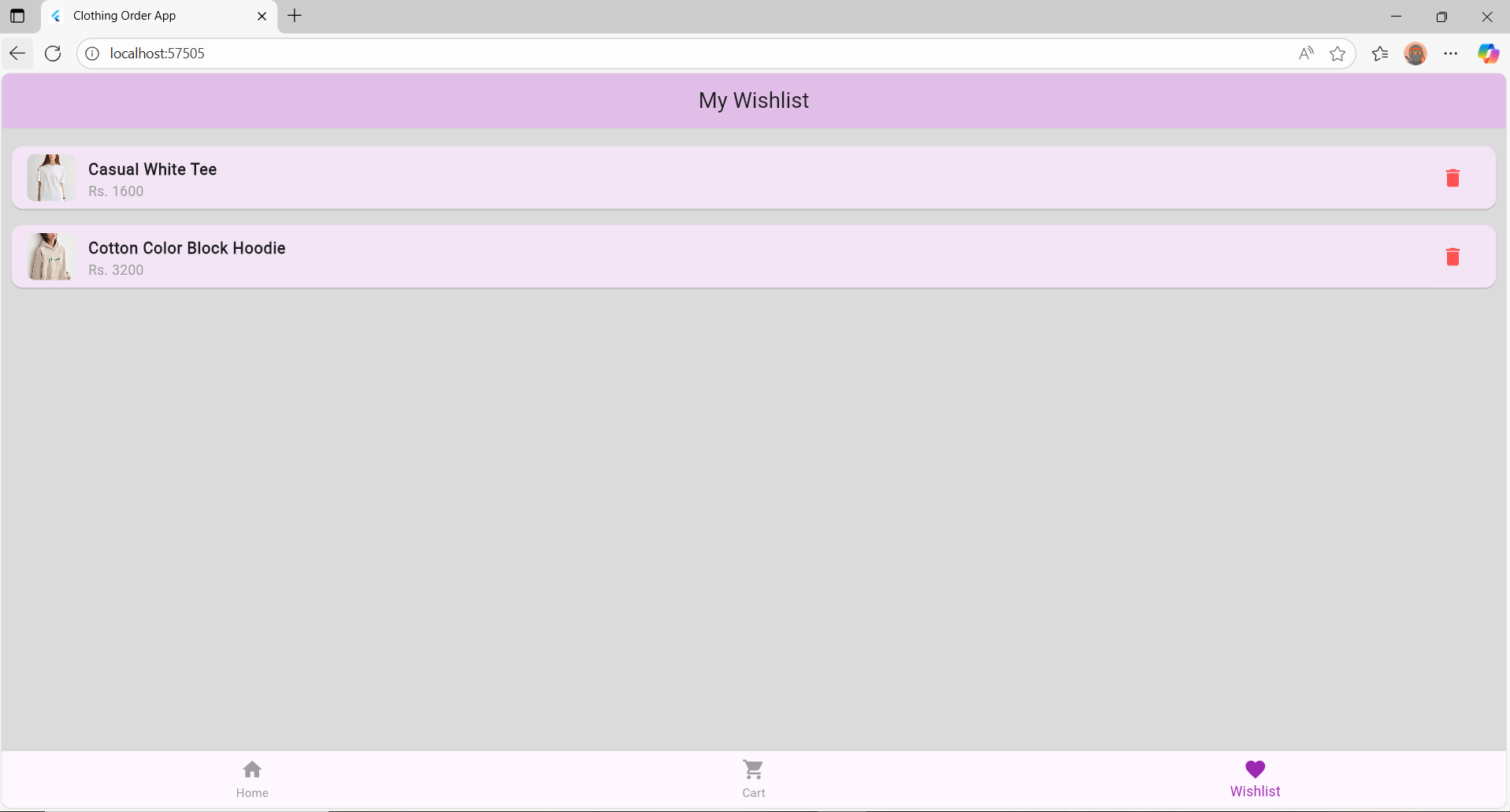


**Order screen:**









**4. Data Storage**

**Database Used:** Hive

**Justification:**

* Hive is a **lightweight, NoSQL database** optimized for Flutter apps.
* It stores data **locally**, allowing offline functionality.
* It’s **fast** and doesn’t require a separate server setup.
* Perfect for small to medium applications like clothing apps.
* Used here to store **Products, Cart items, Orders**, and **Wishlist** efficiently.

Example:

* product\_provider.dart handles loading products from Hive.
* cart\_provider.dart manages adding/removing items using Hive boxes.

### ****5. APIs / Packages / Plug-ins****

**Packages Used:**

* provider → for state management between UI and logic.
* hive and hive\_flutter → for local database.
* path\_provider → to locate device directories for Hive storage.

**Justification:**  
These packages simplify app architecture and make data management and UI updates efficient.  
For example, provider ensures that any cart or wishlist change updates the UI instantly.

**6. Issues and Bugs Encountered and Resolved**

**Error.1**

**Issue:** Navigation not working properly  
**Cause:** Missing routes or incorrect navigation setup.  
**Solution:** Used Navigator.push properly inside GestureDetector to navigate between screens.

**Error.2**

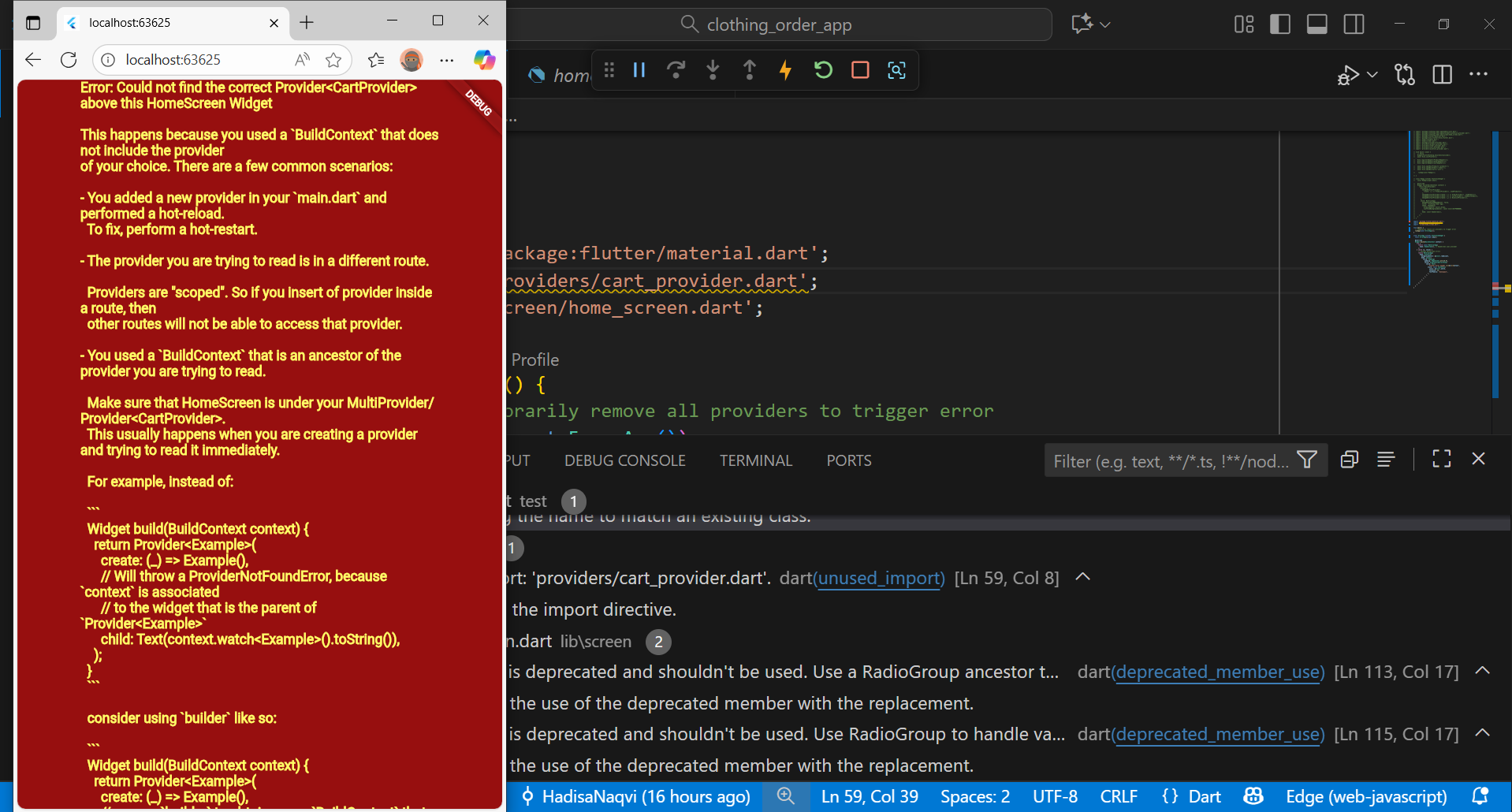
**Issue:** Image not displaying in Product Detail Screen  
**Cause:** The property name used (imageUrl) did not match the one defined in the model.  
**Solution:** Updated the model and used the correct field name for the image.

**Error.3**

**Issue:** “git not recognized” error during deployment  
**Cause:** Git was not installed or not added to the system PATH.  
**Solution:** Installed Git and added it to the environment variables.

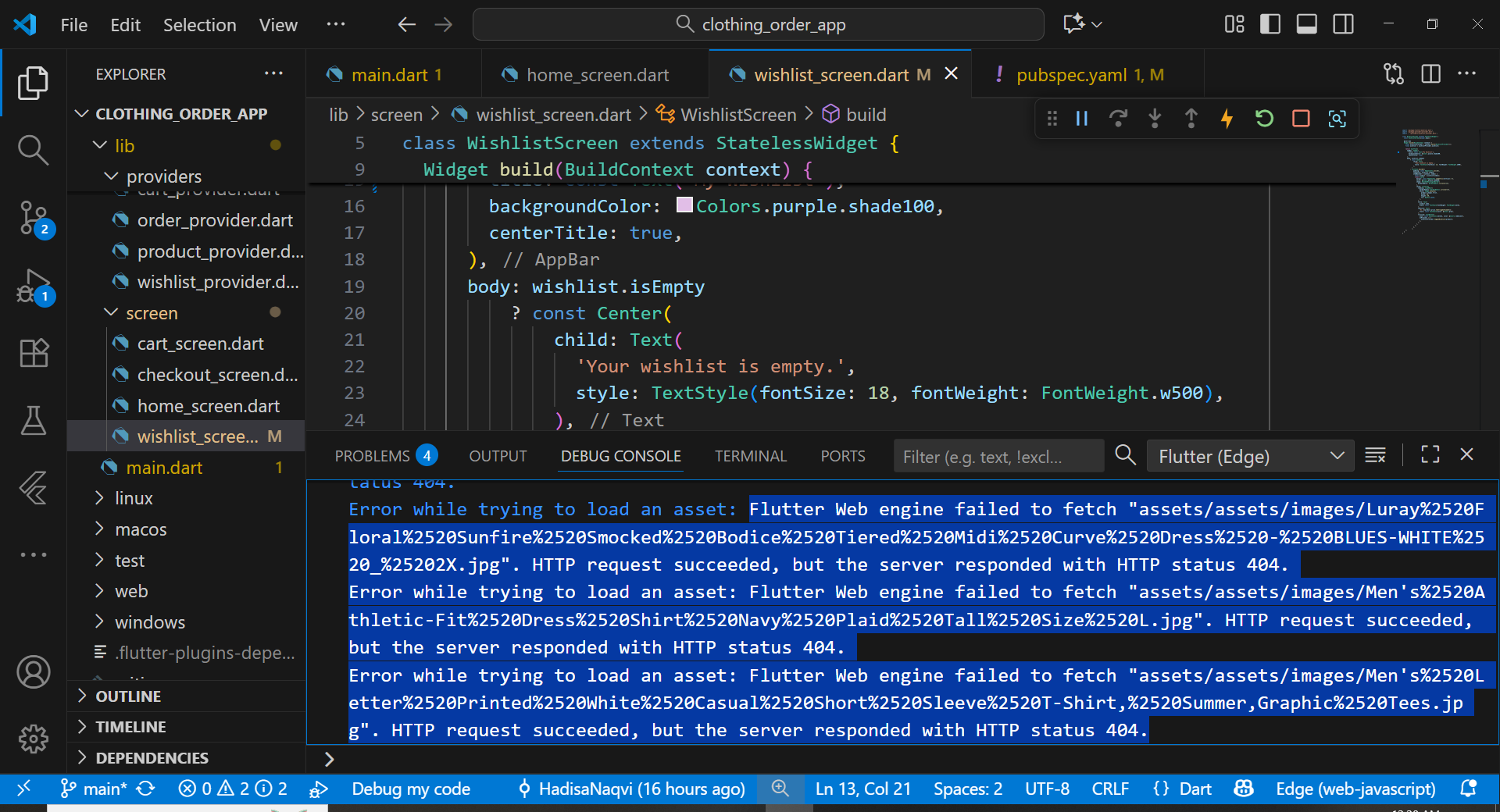
**Error.4**

A red screen appeared because the providers (like CartProvider) were accessed before initialization; fixing it required properly adding all providers inside the MultiProvider before the MaterialApp.



**Error.5**

**Assets were not defined:**



**Error.6**

**Hive package was missing:**

