Shoply Mobile Application

Submitted by: Jha Kunalkumar Dilipkumar

Role Applied: Application Developer Intern

Date: November 27, 2025

1. Executive Summary

Shoply is a high-fidelity mobile application prototype designed to simulate a modern e-commerce shopping experience. Built using the **Flutter** framework and **Dart** programming language, the project demonstrates core competencies in cross-platform mobile development, including responsive UI design, efficient state management, and complex navigation transitions. The application focuses on delivering a seamless user experience (UX) through the use of Material Design 3 standards and optimized rendering performance.

2. Technical Architecture & Tech Stack

- **Framework:** Flutter (Stable Channel)
- **Language:** Dart 3.0+
- **Design System:** Material Design 3 (Material You)
- **State Management:** Ephemeral State (StatefulWidget, setState)
- **Architecture Pattern:** Component-Based Architecture (Separation of Data Models, UI Screens, and Logic).

3. Key Features & Implementation Details

A. Dynamic Data Filtering Logic

The application avoids static lists by implementing a real-time filtering system.

- **Logic:** A custom filtering algorithm sorts the product dataset based on the user's selected category (e.g., Tech, Fashion, Home).
- **Implementation:** Utilized Dart's functional programming method .where() to return a new list whenever the state changes.
- **UX Benefit:** This allows for instant content updates without reloading the page, ensuring a snappy interface.

B. Optimized Grid Layout System

To display products effectively on mobile screens, I implemented a responsive grid system rather than a standard list.

- **Widget:** GridView.builder with SliverGridDelegateWithFixedCrossAxisCount.

- **Performance:** The builder pattern ensures that off-screen grid items are lazy-loaded, conserving memory resources on the device.
- **Design:** Configured a generic Aspect Ratio of 3:4 for product cards to ensure consistent image presentation across different device sizes.

C. Shared Element Transitions (Hero Animations)

To demonstrate advanced UI polish, I implemented Hero animations.

- **Function:** When a user clicks a product in the grid, the image "floats" and expands into the Detail Screen.
- **Technical Detail:** This was achieved by wrapping the image assets in Hero widgets with unique tag IDs tied to the Product ID. This maintains visual continuity and spatial awareness for the user.

D. Robust Data Modeling

Instead of using raw maps or untyped data, I engineered a strong data layer using a custom Product class. This ensures type safety throughout the application, preventing null errors and making the codebase scalable for future backend integration.

4. Code Snippets

Data Model Definition:

Dart
```
class Product {
  final String id;
  final String title;
  final double price;
  final IconData icon;
  final String category;

  Product({required this.id, required this.title, ...});
}
```

Category Filtering Algorithm:

Dart
```
List<Product> get _filteredProducts {
  if (_selectedCategory == 'All') return _allProducts;
  return _allProducts.where((p) => p.category == _selectedCategory).toList();
}
```

5. Future Roadmap (Scalability)

While this prototype operates on local data for demonstration purposes, the architecture is designed to support future scaling:

1. **Backend Integration:** Replace the local _allProducts list with an HTTP GET request to a REST API or Firebase Cloud Firestore.
2. **State Management:** Upgrade from setState to **Provider** or **Bloc** for global cart management across multiple screens.
3. **User Authentication:** Implement Firebase Auth to allow users to save their cart and order history.

6. Conclusion

The **Shoply** project validates my ability to build functional, aesthetically pleasing, and technically sound mobile applications. It demonstrates not just coding ability, but an understanding of User Experience, Memory Management, and Software Architecture principles required for the Application Developer Intern role.