GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY

Mobile Computing - IT3093

Assignment Report

BSc in IT Degree Program (Intake 39)

Faculty of Computing Department of Information Technology	
Group Details Group No: 13	
Student No:	**Student Name**
M/IT/22/002	NMK Wickramarathana
M/IT/22/004	LL Kumarasinghe
D/BIT/22/0001	WMLH Weerasinghe
D/BIT/22/0002	DTN Wijesooriya
D/BIT/22/0004	KKLD Karunarathna

Contents

1.INTRODUCTION

Creating a simple e-commerce app in Flutter involves integrating several key features to ensure a seamless and efficient shopping experience. Essential functionalities include user authentication, allowing users to sign up, log in, and manage their profiles. The app will feature product listings categorized by types such as shoes, clothes, and watches, with a search function to easily find specific items. Detailed product pages will provide comprehensive information, including images, descriptions, prices, and options to add items to the cart or favorites. A shopping cart will enable users to review and modify their selected items, and proceed through a streamlined checkout process. Order management functionalities will allow users to place orders, view order history, and cancel orders if needed.

Interaction with stores will be facilitated through features that show store locations on a map and provide contact options. Notifications will keep users informed about order status and promotions, while integrating secure payment gateways ensures smooth transactions. Additional features like a discount calculator and a favorites panel will enhance the user experience. To build the front-end, Flutter will be used for its ability to create visually appealing and responsive user interfaces. Firebase will serve as the back-end, providing robust solutions for user authentication, data storage, and real-time database needs.

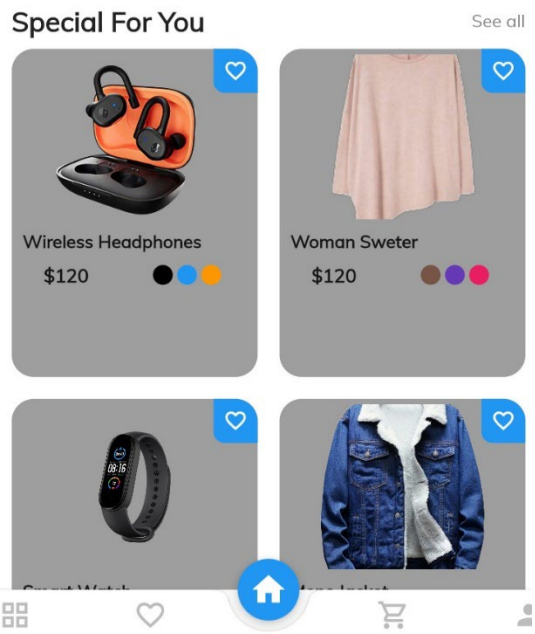
Furthermore, the app will focus on performance optimization to ensure quick load times and a responsive user interface. Implementing efficient state management techniques in Flutter will help manage the app's state effectively, providing a smooth and consistent user experience. The design will follow best practices in UX/UI to create an intuitive and engaging interface that makes shopping enjoyable. This combination of features and technologies forms a robust foundation for a basic e-commerce app in Flutter, making it a powerful tool for both users and businesses looking to engage in online retail.

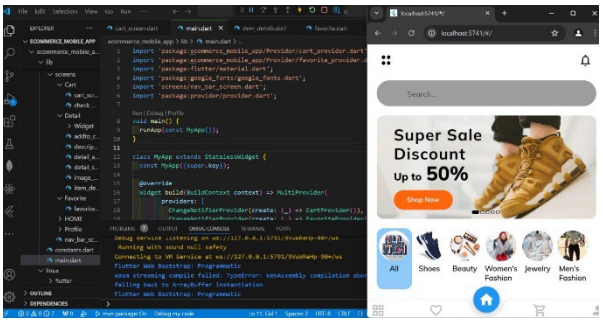
2. API INTEGRATION PROCESS

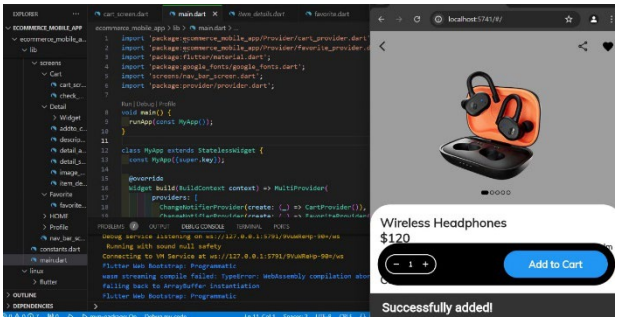
To integrate various APIs for user authentication, product data management, payment processing, and location services in your Flutter project, follow these steps:

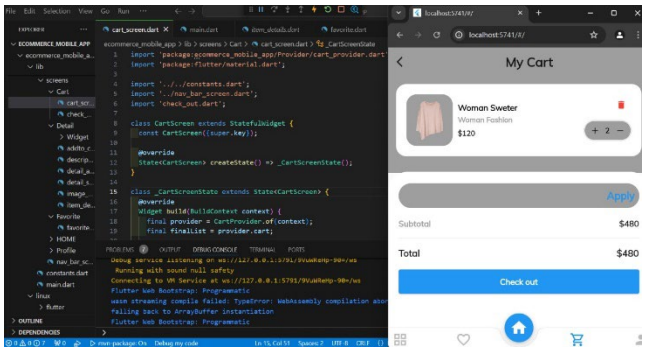
1. **User Authentication with Firebase:** Start by creating a Firebase project and adding the Firebase SDK to your Flutter project by including the firebase core and firebase AUTH packages in the pubspec.yaml file. Initialize Firebase in the main. Dart file, and implement authentication functions for sign-up, login, and logout using Firebase Authentication methods. This setup allows users to create accounts, log in, and manage their profiles securely.
2. **Product Data Management with Firestore:** Utilize Firebase Firestore for managing product data. Add the cloud_firestore package to your pubspec.yaml file. Implement CRUD operations for products by creating functions to add, retrieve, update, and delete product data in Firestore. This configuration helps manage the product inventory, making it easier to display products to users and handle updates efficiently.
3. **Payment Integration with Stripe:** For payment processing, use Stripe. Create a Stripe account and obtain your API keys from the Stripe Dashboard. Add the stripe_payment package to your project and initialize Stripe in your Flutter app with your publishable key. Implement payment functions to handle payment processing, allowing users to enter payment details and complete transactions securely. This ensures smooth and secure payment handling in your app.

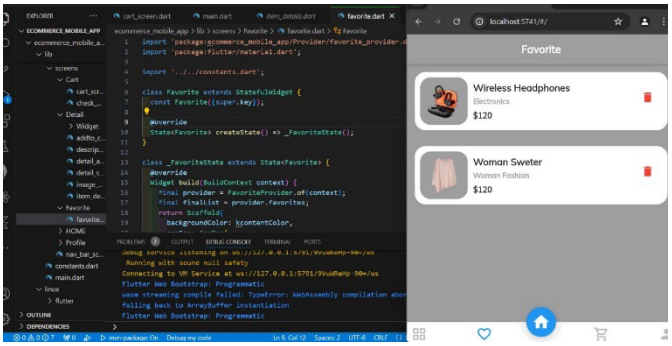
4. SAMPLE REQUESTS & RESPONSES











5. KEY IMPLEMENTATION DETAILS

1. **User Authentication**
    - **Sign Up:** Allow users to create an account.
    - **Log In:** Enable users to access their accounts.
    - **Profile Management:** Let users update their profiles and view their account details.

2. **Product Listing**
    - **Categories:** Display products categorized by types (shoes, clothes, watches, etc.).
    - **Search:** Provide a search function to find specific items.

3. **Product Details**
    - **Detailed Information:** Show comprehensive details about each product, including images, descriptions, prices, and sizes.
    - **Add to Cart/Favorites:** Allow users to add products to their shopping cart or mark them as favorites.

4. **Shopping Cart**
    - **Review Items:** Let users review their selected items.
    - **Modify Cart:** Enable adding, removing, or updating item quantities.
    - **Checkout Process:** Streamline the checkout process for order placement.

5. **Order Management**
    - **Place Orders:** Allow users to place orders from their cart.
    - **Order History:** Provide a history of past orders.
    - **Order Cancellation:** Enable users to cancel orders if necessary.

6. **Interaction Features**
    - **Location Services:** Show store locations on a map.
    - **Contact Options:** Provide contact details for customer support.

7. **Notifications**
    - **Order Status:** Inform users about the status of their orders.
    - **Promotions:** Notify users about ongoing promotions and discounts.

8. **Secure Payment Gateway**
    - **Online Payments:** Integrate secure payment methods for smooth transactions.

9. **Additional Features**
    - **Discount Calculator:** Provide a tool to calculate discounts on products.
    - **Favorites Panel:** Allow users to manage their favorite items.

6. ESSAY.

Creating an e-commerce app in Flutter for selling items such as shoes, clothes, watches, and more, involved integrating numerous key features to provide a seamless and efficient shopping experience. This end-to-end process offered extensive hands-on experience in mobile app development, API integrations, and enhancing user experiences. This reflective essay discusses the challenges encountered, insights gained, and lessons learned throughout the application development process.

Challenges

I. Understanding API Documentation:

Understanding the API documentation for the various services integrated into the app was the first challenge. Despite the comprehensive documentation available, it took a significant effort to read and comprehend the details due to the volume of information and the intricacies of the operations required.

II. API Integration:

Integrating multiple APIs into a single application required careful handling of asynchronous data collection and processing. One major issue was ensuring that calling APIs did not cause the application to lag or the user interface to become unresponsive. Efficient data management and optimization were crucial to maintain a smooth user experience.

III. Handling Permissions:

Proper handling of user permissions was essential for accessing certain features, such as geolocation services. Careful planning and execution were needed to ensure the app requested and managed permissions effectively, and to handle scenarios where permissions were denied gracefully.

IV. State Management:

Managing the application's state while continuously fetching and updating data was challenging. Selecting an appropriate state management solution, such as Provider, Bloc, or Riverpod, was crucial to maintain the application's scalability and performance. Efficient state management ensured that the app remained responsive and reliable.

V. Error Handling:

Developing robust error handling mechanisms was another significant challenge. A seamless user experience required the smooth management of potential issues such as network failures, API rate limits, and incorrect responses. Implementing comprehensive error handling ensured the app could gracefully handle unexpected situations.

Insights

I. Importance of Thorough Planning:

Thorough planning and a clear understanding of how various components interact can save time and prevent significant problems later in the development process. Detailed design and planning helped in anticipating potential issues and devising effective solutions in advance.

II. API Performance and Limitations:

Understanding the capabilities and limitations of the APIs used was crucial. For example, some APIs, like the Google Maps API, have rate limits that can lead to service interruptions if exceeded. Utilizing caching techniques and minimizing unnecessary API requests helped mitigate these issues and improve performance.

III. User-Centric Design:

Designing with the end user in mind was essential for creating an engaging and intuitive app. Integrating features such as a discount calculator, favorites panel, and secure payment gateways significantly enhanced the user experience. Focusing on user-centric design ensured that the app met the needs and expectations of its users.

IV. Asynchronous Programming:

Gaining a solid understanding of asynchronous programming in Dart was crucial. Utilizing Streams, Futures, and async/await effectively improved the efficiency of UI updates and data fetching. Mastering asynchronous programming techniques enabled the development of a responsive and smooth user interface.

V. Continuous Testing and Feedback:

Continuous testing and debugging were essential to identify and resolve issues early in the development process. Actively seeking user feedback and incorporating it into future updates helped in refining the app and ensuring it met user expectations. Regular testing and feedback loops contributed to the overall quality and reliability of the app.

Lessons Learned

1. **The importance of documentation**

Maintaining extensive documentation during the development phase was quite helpful. It was useful not only for monitoring advancement but also for addressing problems and integrating new team members.

2. **Testing and Debugging:**

To guarantee the dependability and quality of the programmer, extensive testing, comprising unit tests, integration tests, and user acceptability testing, had to be put into place. It took a thorough approach to debug API problems, and it was frequently necessary to examine network traffic and API answers.

3. **Community and Resources:**

It was advantageous to make use of libraries, forums, and community resources. There are active communities for both Flutter and Dart, and many problems that arise have previously been explored and fixed by others.

4. **Testing and Debugging:**

To guarantee the dependability and quality of the program, extensive testing, comprising unit tests, integration tests, and user acceptability testing, had to be put into place. It took a thorough approach to debug API problems, and it was frequently necessary to examine network traffic and API answers.