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: 01	
Student No:	**Student Name:**
D/BIT/22/0043	RMS Laknath
D/BIT/22/0036	TDCS Gunawardana
D/BIT/22/0038	NW Kumarasinghe
D/BIT/22/0055 (D/BIS/22/0019)	HAH Sathsarani
D/BIT/22/0039	RAKK Ranasinghe

CONTENTS

1. INTRODUCTION.

Creating a simple food ordering app in Flutter involves integrating several essential features to offer a seamless user experience. Key functionalities include:

- **User Authentication**: Allowing users to sign up, log in, and manage their profiles securely.
- **Restaurant Listings**: Displaying restaurants by categories, enabling users to search and filter options.
- **Detailed Restaurant Pages**: Providing comprehensive information about each restaurant, including menu items, and allowing users to add dishes to their cart or favorites.
- **Shopping Cart**: Enabling users to review and modify their selected items before proceeding to checkout.
- **Order Management**: Allowing users to place orders, view their order history, and cancel orders if necessary.
- **Interaction with Restaurants**: Showing restaurant locations on a map and providing contact options for easy communication.
- **Notifications**: Keeping users informed about their order status and any promotions or updates.
- **Secure Payment Gateways**: Integrating online payment options to ensure smooth and secure transactions.

The technology stack for this app includes Flutter for the front-end development, Firebase for authentication and data storage, and Google Maps API for location services. This combination of features and technologies creates a solid foundation for a basic food ordering app in Flutter.

2. API INTEGRATION PROCESS.

To handle user authentication, Firebase Authentication can be utilized. Start by setting up a Firebase project and adding the Firebase SDK to your Flutter project. Update your `pubspec.yaml` file to include the `firebase_core` and `firebase_auth` packages. Initialize Firebase in your `main.dart` file and implement authentication functions like sign-up, login, and logout using Firebase Authentication methods. This setup allows users to securely create accounts, log in, and manage their profiles.

For managing product data, Firebase Firestore is a dependable solution. Include the `cloud_firestore` package in your `pubspec.yaml` file. Implement CRUD operations for products by creating functions to add, retrieve, update, and delete product data in Firestore. This enables efficient management of the product inventory, making it straightforward to display products to users and handle updates.

To process payments, Stripe can be integrated. Create a Stripe account and get 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 functions to handle payment processing, allowing users to enter their payment details and complete transactions securely. This integration ensures smooth and secure payment handling in your app.

For displaying shop locations, the Google Maps API can be employed. Get an API key from the Google Cloud Platform Console by enabling the Maps SDK for both Android and iOS. Add the `google_maps_flutter` package to your project and configure the platform-specific settings as per the official documentation. Implement Google Maps functionality to show shop locations on a map by setting up an initial camera position and adding markers. This feature makes it easy for users to find physical shop locations.

3. API DOCUMENTATION REFERENCES.

Here are the essential documentation references for the APIs used in this food ordering app:

- **Firebase Authentication**
  - Firebase Authentication. (n.d.). Retrieved from https://firebase.google.com/docs/auth/
  - Introducing Firebase Authentication. (n.d.). Retrieved from https://firebase.blog/posts/2016/06/introducing-firebase-authentication
- **Firestore Database**
  - CloudFirestore. (n.d.). Retrieved from https://firebase.google.com/docs/firestore/
- **Geolocator**
  - Flutter Geolocator Plugin. (n.d.). Retrieved from https://pub.dev/packages/geolocator
- **Google Maps API**
  - Google Maps for Flutter. (n.d.). Retrieved from https://pub.dev/packages/google_maps_flutter
  - Adding Google Maps to a Flutter app. (n.d.). Retrieved from https://codelabs.developers.google.com/codelabs/google-maps-influtter/#0
- **Stripe Payment Gateway**
  - Stripe. (n.d.). Retrieved from https://pub.dev/packages/flutter_stripe

4. SAMPLE REQUESTS & RESPONSES

Google Maps API
When integrating Google Maps API into your food ordering app, you can make requests to retrieve various types of geographical information. Below are examples of how to make these requests and interpret the responses.

Google Maps API Request
To get directions between two locations, you can send a URL with specific parameters. Here is an example request:

https://maps.googleapis.com/maps/api/directions/json?origin=PLACE_ID_OR_LATLNG&destination=PLACE_ID_OR_LATLNG&key=YOUR_API_KEY

Parameters:
'origin': The starting point for calculating directions. It can be a place ID or latitude/longitude.
'destination': The endpoint for calculating directions. It can be a place ID or latitude/longitude.
'key': Your Google Maps API key.

https://maps.googleapis.com/maps/api/directions/json?origin=6.927079,79.861244&destination=7.290572,80.633726&key=YOUR_API_KEY

Google Maps API Response
The response from the Google Maps API is in JSON format and includes detailed information about the requested data. For the directions example, the response will contain route details, step-by-step navigation, and other relevant information.

5. KEY IMPLEMENTATION DETAILS

Here are some essential implementation details for the E-Commerce Food Ordering App:

1. User Authentication:

Implemented using Firebase Authentication. This included user sign-up, login, and logout functionalities. Secured handling of user credentials and tokens was prioritized to ensure data security.

2. Location Services:

Geolocator: Integrated the Geolocator package to fetch and update the user's real-time location. Managed permissions and ensured location accuracy, providing real-time updates on user location for enhanced functionality.

3. Restaurant Listing:

Google Maps API: Utilized the Google Maps API to fetch and display nearby restaurants based on the user's current location. We implemented features for searching and filtering restaurants, displaying them on a map with relevant details like addresses and ratings.

4. Order Management:

Firestore: Managed order data using Firestore, including storing user orders, updating order statuses, and retrieving order history. Implemented real-time data synchronization to update the app with the latest order information.

5. Payment Processing:

Stripe Integration: Handled secure payment processing using the Stripe API. Managed payment intents, confirmed payments, and ensured compliance with PCI standards for secure transactions.

6. State Management:

Provider/Riverpod: Managed the state of the app using state management solutions like Provider or Riverpod. This ensured efficient data flow and state updates across the app, handling complex scenarios such as cart management and order tracking.

7. Error Handling and Testing:

Implemented comprehensive error-handling strategies for API calls and user interactions. This included retry mechanisms, user-friendly error messages, and logging for troubleshooting. Conducted extensive testing using Flutter's testing framework, including unit tests, integration tests, and user acceptance testing to ensure the app's functionality and reliability.

6. ESSAY.

Developing a food ordering app using Flutter and integrating it with the Geolocator and Google APIs provided a comprehensive end-to-end experience in mobile app development. This process involved significant hands-on work with API integrations and focused on enhancing user experiences. This reflective essay will cover the challenges faced, insights gained, and lessons learned throughout the application development journey.

Challenges

I. Understanding API Documentation:

Understanding the API documentation for Google and Geolocator APIs was the first challenge. Despite the comprehensive nature of these documents, it required significant effort to read and comprehend them due to the vast amount of information and the detailed instructions needed for various operations.

II. API Integration:

Integrating various APIs into a single application necessitated careful handling of asynchronous data collection and processing. A major issue was ensuring that API calls did not cause the application to lag or the user interface to become unresponsive.

III. Handling Permissions:

Proper management of user permissions was essential for geolocation services. Ensuring the app requested and managed location permissions effectively, as well as handling scenarios where permissions were denied, required meticulous planning and execution.

IV. State Management:

Maintaining application stability while continuously fetching data was challenging. Choosing the right state management solution (such as Provider, Bloc, or Riverpod) was critical for ensuring the scalability and performance of the application.

V. Error Handling:

Developing robust error handling mechanisms was another significant obstacle. Ensuring a seamless user experience required smooth management of potential issues such as network failures, API rate limits, and incorrect responses.

Insights

I. Importance of Thorough Planning:

A well-planned design and a comprehensive understanding of how various components interact can save time and reduce the likelihood of significant issues arising later. Proper planning helps in anticipating challenges and devising solutions in advance, ensuring a smoother development process.

II. API Performance and Limitations:

Understanding the features and limitations of the APIs used is crucial. For instance, the Google Maps API has rate limits that can cause service interruptions if exceeded. Implementing caching techniques and minimizing API requests helped mitigate such issues and maintain service continuity.

III. User-Centric Design:

Designing with the end user in mind is essential. Integrating Google APIs with geolocation to enable features like real-time delivery location tracking and local restaurant search significantly improved the user experience. Focusing on user needs and preferences ensures a more intuitive and satisfying app experience.

IV. Asynchronous Programming:

Mastering asynchronous programming in Dart is crucial. Using Streams, Futures, and async/await effectively made UI updates and data fetching more efficient. Understanding how to manage asynchronous operations is key to maintaining a responsive and performant application.

Lessons Learned

I. The importance of documentation:

Maintaining extensive documentation throughout the development phase was highly beneficial. It was valuable not only for tracking progress but also for troubleshooting issues and onboarding new team members. Comprehensive documentation ensures that all team members are aligned and can efficiently address challenges.

II. Testing and Debugging:

Ensuring the reliability and quality of the application required comprehensive testing, including unit tests, integration tests, and user acceptance testing. Debugging API issues demanded a thorough approach, often involving the inspection of network traffic and API responses. This rigorous testing process was essential to deliver a robust and error-free application.

III. Community and Resources:

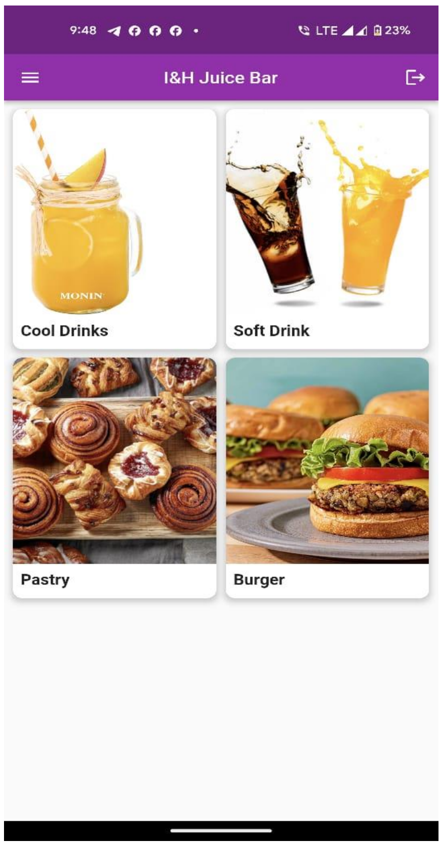
Leveraging libraries, forums, and community resources proved advantageous. The active communities around Flutter and Dart provided valuable support, with many common issues already explored and resolved by others. Utilizing these resources helped expedite problem-solving and improved the development process.

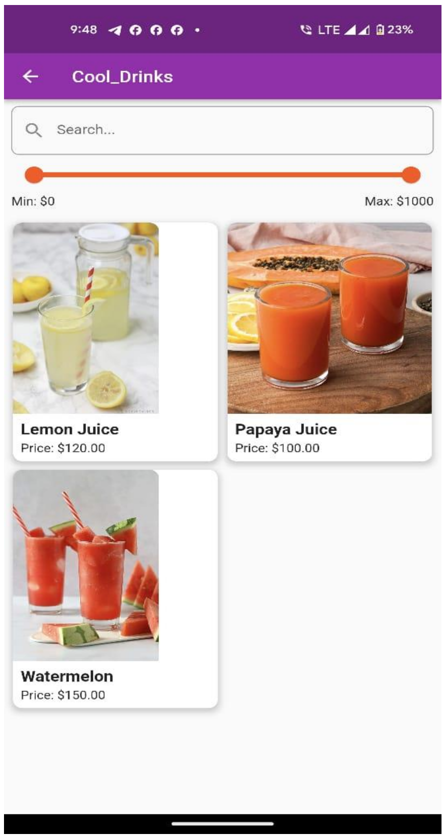
IV. Testing and Debugging:

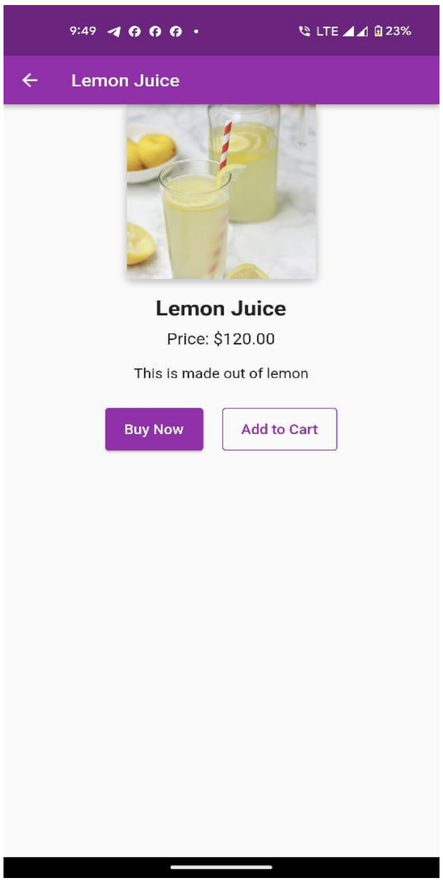
Extensive testing, encompassing unit tests, integration tests, and user acceptance testing, was crucial to ensure the application's reliability and quality. A thorough approach to debugging API issues was necessary, often requiring the analysis of network traffic and API responses. This rigorous testing process was essential for delivering a high-quality, dependable application.

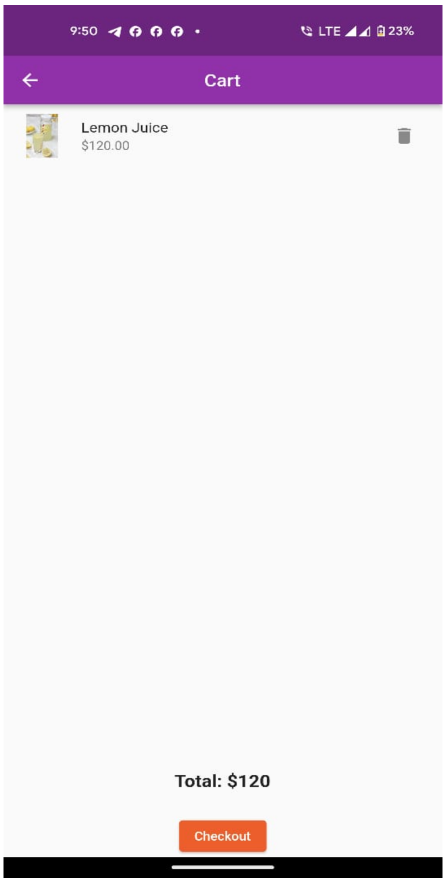
In conclusion, creating a food ordering app integrating Google API and geolocation was both challenging and fulfilling. It provided valuable lessons in state management, user experience design, and API integration. The project underscored the importance of planning, documentation, and continuous learning in software development. The final product, delivering a seamless user experience, reflected the dedication and expertise invested in its development.

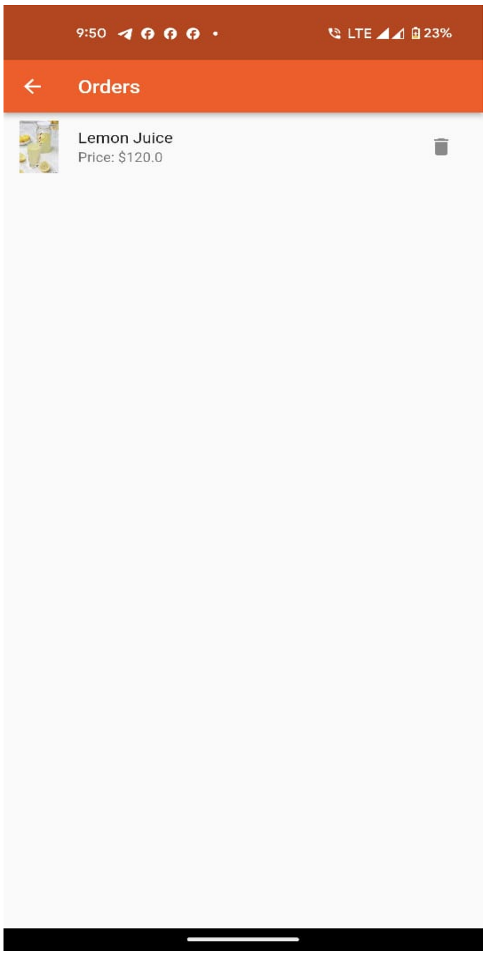
7. INTERFACE DESIGN





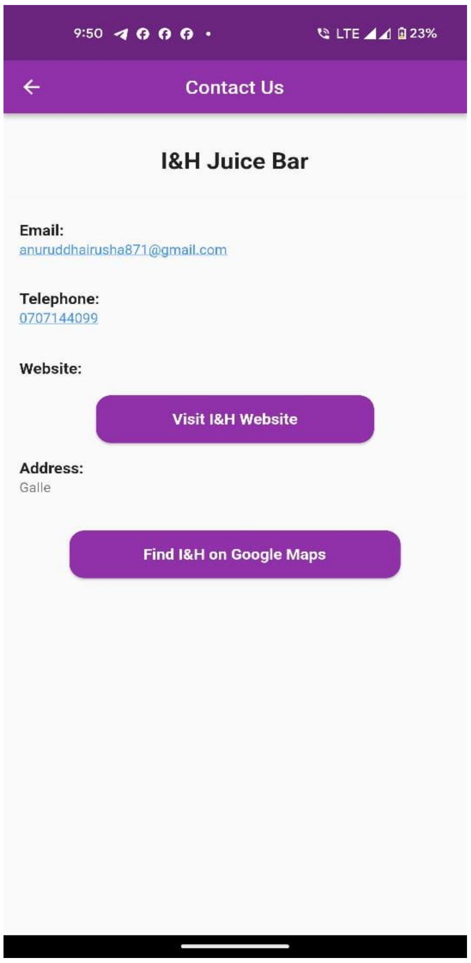





9:50 LTE 23%

Orders

Lemon Juice
Price: $120.0


9:50 LTE 23%

Contact Us

I&H Juice Bar

Email:
anuruddhairusha871@gmail.com

Telephone:
0707144099

Website:

Visit I&H Website

Address:
Galle

Find I&H on Google Maps