

DELISHLIA

INTERDISCIPLINARY PROJECT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology in Information Technology

by

BARATH C (41120038)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY-1 UNIVERSITY BY UGC

Accredited "A++" by NAAC

12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
BARATH C (41120038) who carried out the project entitled
"Delishlia" under my supervision from **JANUARY 2024** to **MAY**
2024.

Internal Guide

Dr. L. MARY GLADENCE M.E., Ph.D.,

Head of the Department

Dr. R. SUBASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 08.05.24

Internal Examiner

External Examiner

DECLARATION

I BARATH C hereby declare that the Project Report entitled as "DELISHLIA" done by me under the guidance of

Dr .L. MARY GLADENCE M.E.,Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08.05.2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

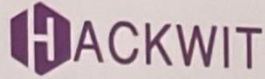
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.R.Subhashini M.E.,Ph.D.**,Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. Mary Gladence M.E.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION



HACKWIT TECHNOLOGIES PVT LTD
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,
Ramapuram, Chennai - 600089.
admin@hackwittechnologies.com
www.hackwittechnologies.com

10.02.2024

CERTIFICATE FOR INTERNSHIP

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **Barath C (Reg No: 41120038)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22nd January to 30th January 2024.

During the internship, **Barath C** has closely worked as a part of the **Flutter** by carrying out the project entitled "**Delishlia**".

He demonstrated good developing and learning skill with self-motivated attitude to learn new things. His contribution to implement the Flutter was good. We wish him all the best for his future endeavors.

Best Regards,

Ravikumar S
CEO & Managing Director
Hackwit Technologies Pvt Ltd



CERTIFICATE OF COMPLETION



ABSTRACT

Delishlia is a comprehensive food delivery application designed to provide users with a convenient and efficient way to order food from their favorite restaurants. By integrating these three apps, Delishlia allowing customers to browse through a wide range of restaurants, explore menus, and place orders with just a few taps. Delishlia is not just a standalone application; it is part of an interconnected ecosystem that includes the Delishlia Restaurant Admin and Delishlia Delivery Partner apps. The Delishlia Restaurant Admin app enables restaurant owners to manage their menu, track orders, and communicate with customers seamlessly. Restaurant admins can update menu items, set pricing, and monitor orders in real-time, ensuring a smooth and efficient process. On the other hand, the Delishlia Delivery Partner app is designed for delivery personnel, allowing them to accept delivery requests, navigate to the customer's location, and ensure timely delivery of orders. Delivery partners can view their earnings, track delivery history, and manage their availability through the app. By integrating these three apps, Delishlia creates a robust and interconnected ecosystem that benefits users, restaurant owners, and delivery partners alike, offering a seamless and efficient food delivery experience for all parties involved.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vii
	LIST OF ABBREVIATIONS	x
	LIST OF FIGURES	xi
1	INTRODUCTION	1
	1.1 Flutter	2
	1.2 Dart	3
	1.3 Firebase	4
	1.4 Problem Statement	5
	1.5 Objectives	8
2	METHODOLOGY	10
	2.1 Initial Project Requirements	10
	2.2 Proposed System Architecture	13
	2.3 Flow Diagram	17
	2.4 Modules	19
	2.4.1 User Management(Delishlia User App)	19
	2.4.2 Authentication Management	20
	2.4.3 User Management(Delishlia Restaurant Admin App)	19
	2.4.4 User Management(Delishlia Delivery Partner App)	23

	2.5	Development and Deployment Setup	24
3		RESULTS AND DISCUSSION	28
4		CONCLUSION AND FUTURE WORK	36
	4.1	Conclusion	36
	4.2	Future Work	37
		REFERENCES	39
		APPENDICES	43
	A.	Sample Code	43
	B.	Screen Shots	45

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
UI	- User Interface
UX	- User Experience
iOS	- iPhone Operating System
UAT	- User Acceptance Testing
IDE	- Integrated Development Environment

LIST OF FIGURES

Figure No.	Name of the Figure	Page No.
2.1	Proposed Application Architecture	12
2.2	Firebase Architecture	14
2.2	Flow Diagram	16
2.3	Project Creation	22
2.4	Firebase Connection	23
2.5	Firebase Authentication	24
2.6	Flutter Run	24
3.1	Login Page Interface	26
3.2	Dashboard Screen Display	27
3.3	Mentor Picker Interface	28
3.4	Feedback Submission Screen	29
3.5	Firebase Configurations	30
3.6	Initialize Firebase	31

CHAPTER 1

INTRODUCTION

Delishlia emerges as a beacon of seamless food delivery services. With features like real-time navigation, order tracking, and earnings monitoring, the Delivery Partner app empowers delivery personnel to optimize their routes, maximizer friendly interface and robust features, Delishlia connects customers, restaurants, and delivery partners, offering an unparalleled dining experience.

Delishlia for Customers: For customers, Delishlia offers a hassle-free way to satisfy their cravings. Whether it's a quick lunch at the office, a cozy dinner at home, or a weekend treat with friends, Delishlia provides a diverse range of restaurants and cuisines to choose from.

Delishlia Restaurant Admin: At the heart of the Delishlia ecosystem is the Delishlia Restaurant Admin app. This powerful tool empowers restaurant owners to take control of their online presence and streamline their operations. With the Restaurant Admin app, restaurant owners can easily manage their menu, update pricing, and track orders in real-time. They can also engage with customers, respond to queries, and provide a personalized dining experience, ensuring customer satisfaction and loyalty.

Delishlia Delivery Partner: To ensure prompt and efficient delivery, Delishlia integrates with the Delishlia Delivery Partner app. Delivery partners can accept delivery requests, navigate to the customer's location, and ensure timely delivery of orders.

1.1.FLUTTER

Flutter, a robust cross-platform UI toolkit developed by Google, is at the heart of MentorSynth Hub, shaping the application's foundation and user interface. The utilization of Flutter brings a paradigm shift to the traditional app development approach, offering a single codebase that transcends platform-specific limitations. This innovation allows MentorSynth Hub to deliver a visually captivating and responsive user interface, adeptly adjusting to both iOS and Android environments.

The efficiency of Flutter lies in its ability to streamline the development process, significantly reducing the complexities associated with managing separate codebases for different platforms. Developers benefit from a cohesive workflow, resulting in accelerated development cycles and enhanced productivity. Moreover, Flutter's hot reload feature enables real-time code changes and instant visual feedback, facilitating quick iterations and adjustments.

One of the key advantages of Flutter is its capability to ensure a consistent and high-quality user experience across diverse devices. By leveraging Flutter's rich set of customizable widgets, MentorSynth Hub achieves a cohesive design language and smooth interactions, irrespective of the underlying operating system. Users can seamlessly transition between devices without compromising on the application's performance or aesthetics.

1.2.DART

i)Unified Codebase:

MentorSynth Hub relies on Dart, a client-optimized programming language, for implementing backend logic and server-side operations. Dart's versatility enables developers to maintain a unified codebase for both the frontend (Flutter) and backend, fostering consistency and ease of collaboration.

ii)Asynchronous Programming:

Dart's support for asynchronous programming is crucial for handling concurrent operations efficiently. This feature ensures that MentorSynth Hub can seamlessly manage multiple tasks, such as handling user requests, database operations, and external API calls, without causing delays or disruptions in the application's responsiveness.

iii)Scalability:

Dart's scalability is instrumental in accommodating the potential growth of MentorSynth Hub. As the user base expands and the demand for mentorship increases, Dart's ability to scale allows the platform to handle higher traffic, ensuring a smooth and responsive experience for all users.

iv)Strong Typing System:

Dart's strong typing system enhances code reliability and maintainability in MentorSynth Hub. By catching potential errors during the development phase, Dart enables developers to write robust and error-resistant code, reducing the likelihood of runtime issues and enhancing the overall stability of the application.

1.3.FIREBASE

i)Real -time Data Synchronization:

Firebase Cloud Firestore supports real-time data synchronization, allowing seamless updates across multiple devices. This feature is crucial for MentorSynth Hub, enabling mentors and students to experience instant changes in discussions, mentorship requests, and collaborative interactions.

ii)Firebase Authentication for User Security:

Firebase Authentication provides a secure and straightforward user authentication process. MentorSynth Hub leverages this feature to ensure that only authorized users, both mentors and students, can access the platform. Multi-factor authentication and identity verification contribute to a robust security layer.

iii)Serverless Cloud Functions:

Firebase offers serverless cloud functions that enable the execution of backend logic without managing servers. MentorSynth Hub utilizes these functions for various tasks, such as sending notifications, updating user data, and handling asynchronous operations. This serverless architecture enhances scalability and efficiency.

iv)Firebase Cloud Storage for Media Assets:

Storing media assets, such as profile pictures and discussion attachments, is streamlined with Firebase Cloud Storage. MentorSynth Hub utilizes this feature to efficiently manage and serve multimedia content, ensuring a smooth and responsive user.

1.4.PROBLEM STATEMENT

In today's fast-paced world, the demand for convenient and efficient food delivery services is higher than ever. However, existing food delivery platforms often face various challenges that hinder the seamless delivery of food from restaurants to customers. These challenges include:

Poor Communication between Customers, Restaurants, and Delivery Partners:

Poor communication between customers, restaurants, and delivery partners is a significant challenge faced by existing food delivery platforms. This lack of effective communication often results in delays, order errors, and customer dissatisfaction. Customers may experience uncertainty about the status of their orders, leading to frustration and a poor user experience. Restaurants struggle to convey order details accurately to delivery partners, resulting in delays and incorrect deliveries. Similarly, delivery partners may face difficulties in accessing order information promptly, leading to inefficiencies in delivery routes and delays in order fulfillment. Overall, poor communication among these key stakeholders hinders the seamless delivery of food from restaurants to customers, impacting the efficiency and reliability of the food delivery service.

Inefficient Order Management Systems:

Inefficient order management systems are a common challenge in the food delivery ecosystem. Many existing platforms rely on outdated and inefficient systems, leading to delays in order processing and delivery. These systems often lack the capability to handle large volumes of orders efficiently, resulting in bottlenecks

and increased processing times. As a result, customers may experience long wait times for their orders, leading to frustration and a poor user experience. Moreover, inefficient order management systems can also lead to order errors, such as incorrect items or missing items, further exacerbating customer dissatisfaction. Additionally, restaurant owners may struggle to keep track of incoming orders and manage their inventory effectively, leading to operational inefficiencies and lost revenue opportunities. Overall, inefficient order management systems pose significant challenges to the seamless and timely delivery of food from restaurants to customers, highlighting the need for a more streamlined and efficient solution.

Lack of Real-Time Tracking:

The lack of real-time tracking is a significant issue in the food delivery industry, causing uncertainty and frustration for customers, restaurants, and delivery partners alike. Without real-time tracking capabilities, customers are left in the dark about the status of their orders, leading to anxiety and dissatisfaction. They are unable to track the progress of their delivery in real-time, making it difficult to plan their day around the arrival of their food. Similarly, restaurants struggle to provide accurate delivery estimates to customers, leading to dissatisfaction and negative reviews. Additionally, delivery partners may face difficulties navigating to the customer's location without real-time tracking, leading to delays and missed deliveries. Overall, the lack of real-time tracking in food delivery services undermines the efficiency and reliability of the entire delivery process, highlighting the need for a solution that provides real-time tracking capabilities to all stakeholders involved.

Limited Control for Restaurant Owners:

Limited control for restaurant owners is a significant challenge in the food delivery app ecosystem. Many existing platforms provide restaurant owners with limited control over their online presence and order management processes. Restaurant owners often struggle to update their menus, manage inventory, and track orders in real-time, leading to inefficiencies and missed opportunities for growth. Without adequate control over their online presence, restaurant owners may find it difficult to accurately represent their offerings to customers, resulting in lost sales and customer dissatisfaction. Additionally, the lack of real-time order tracking and communication tools makes it challenging for restaurant owners to provide timely updates to customers and ensure a seamless delivery experience. Overall, the limited control for restaurant owners in existing food delivery apps hinders their ability to effectively manage their businesses and deliver a high-quality experience to their customers.

1.5. OBJECTIVES

The primary objective of Delishlia is to revolutionize the food delivery experience by providing a seamless and efficient ecosystem that connects customers, restaurants, and delivery partners. With the interconnected Delishlia app suite, our goals are to streamline the ordering process, optimize restaurant management, and enhance the delivery experience. Specifically, our objectives include providing customers with a user-friendly interface for easy ordering, empowering restaurant owners with the Delishlia Restaurant Admin app to manage their online presence effectively, and enabling delivery partners to optimize their routes and ensure timely deliveries with the Delishlia Delivery Partner app. We aim to improve communication between all stakeholders, reduce delays and order errors, and continuously innovate and expand the Delishlia network to provide a reliable and trustworthy food delivery service for all.

Key Challenges:

i)Poor Communication between Stakeholders:

Existing food delivery platforms often suffer from poor communication between customers, restaurants, and delivery partners. Lack of effective communication channels leads to delays, order errors, and customer dissatisfaction.

ii)Inefficient Order Management Systems:

Many food delivery platforms rely on outdated and inefficient order management systems. This results in delays in order processing and delivery, leading to customer dissatisfaction and operational inefficiencies for restaurants and delivery partners.

iii)Lack of Real-Time Tracking:

Existing platforms often lack real-time tracking capabilities for orders. Customers are left in the dark about the status of their orders, leading to frustration and a poor user experience.

iv)Limited Control for Restaurant Owners:

Restaurant owners face challenges in managing their online presence effectively.

Existing platforms provide limited control over menus, inventory management, and order tracking, hindering restaurant owners' ability to optimize their operations.

v)Inefficient Delivery Processes:

Delivery partners struggle with inefficient delivery processes.

Lack of real-time navigation and communication tools leads to delays in order fulfillment and inefficient route optimization.

vi)Scalability and Growth Challenges:

Existing platforms may face challenges in scaling their operations to accommodate growing customer demand.

Inadequate infrastructure and technology may hinder the platform's ability to expand and provide efficient service coverage.

CHAPTER 2

METHODOLOGY

2.1. INITIAL PROJECT REQUIREMENTS

The initial project requirements for Delishlia encompass various aspects such as functionalities, technologies, and packages.

i)User App (Delishlia):

- 1) User-friendly interface for browsing restaurants, exploring menus, and placing orders.
- 2) User registration and login functionality.
- 3) Search functionality to find restaurants based on location, cuisine, etc.
- 4) Menu browsing with detailed item descriptions, prices, and images.
- 5) Cart management for adding/removing items and adjusting quantities.

ii)Technologies:

- 1) Flutter: Utilize Flutter for cross-platform mobile app development, ensuring a consistent user experience on both iOS and Android.
- 2) Dart: Employ the Dart programming language as the primary language for Flutter app development.
- 3) Firebase: Integrate Firebase for backend services, including authentication, real-time database (Firestore), and cloud storage.

iii)Restaurant Admin App (Delishlia Restaurant Admin):

- 1) Dashboard for managing restaurant profile, menu, and orders.
- 2) Menu management with options to add/edit/delete items, set prices, and update availability.
- 3) Order management system with real-time order tracking, order status updates, and notifications.
- 4) Registration and login functionality for delivery partners.
- 5) Profile management with personal details, vehicle information, and availability settings.
- 6) Order management system with real-time order acceptance, order details, and navigation.
- 7) GPS integration for real-time location tracking and route optimization.

iv)Packages Used:

i)get:

Version: 4.6.6

Purpose: State management and navigation framework.

ii)firebase_core:

Version: 2.25.3

Purpose: Firebase initialization and core functionality.

iii)firebase_auth:

Version: 4.17.3

Purpose: User authentication with Firebase.

iv)cloud_firestore:

Version: 4.15.3

v)cloud_firestore_web:

Version: 3.10.3

Purpose: Web implementation of Cloud Firestore.

vi)firebase_auth_web:

Version: 5.9.3

Purpose: Web implementation of Firebase Authentication.

vii)firebase_core_web:

Version: 2.11.3

Purpose: Web implementation of Firebase Core.

viii)google_sign_in:

Version: 5.2.1

Purpose: Google Sign-In authentication.

ix)fluttertoast:

Version: 8.0.7

Purpose: Displaying customizable toast messages.

x)image_picker:

Version: 1.0.7

Purpose: Picking images from the gallery or taking photos.

2.2.PROPOSED APPLICATION ARCHITECTURE

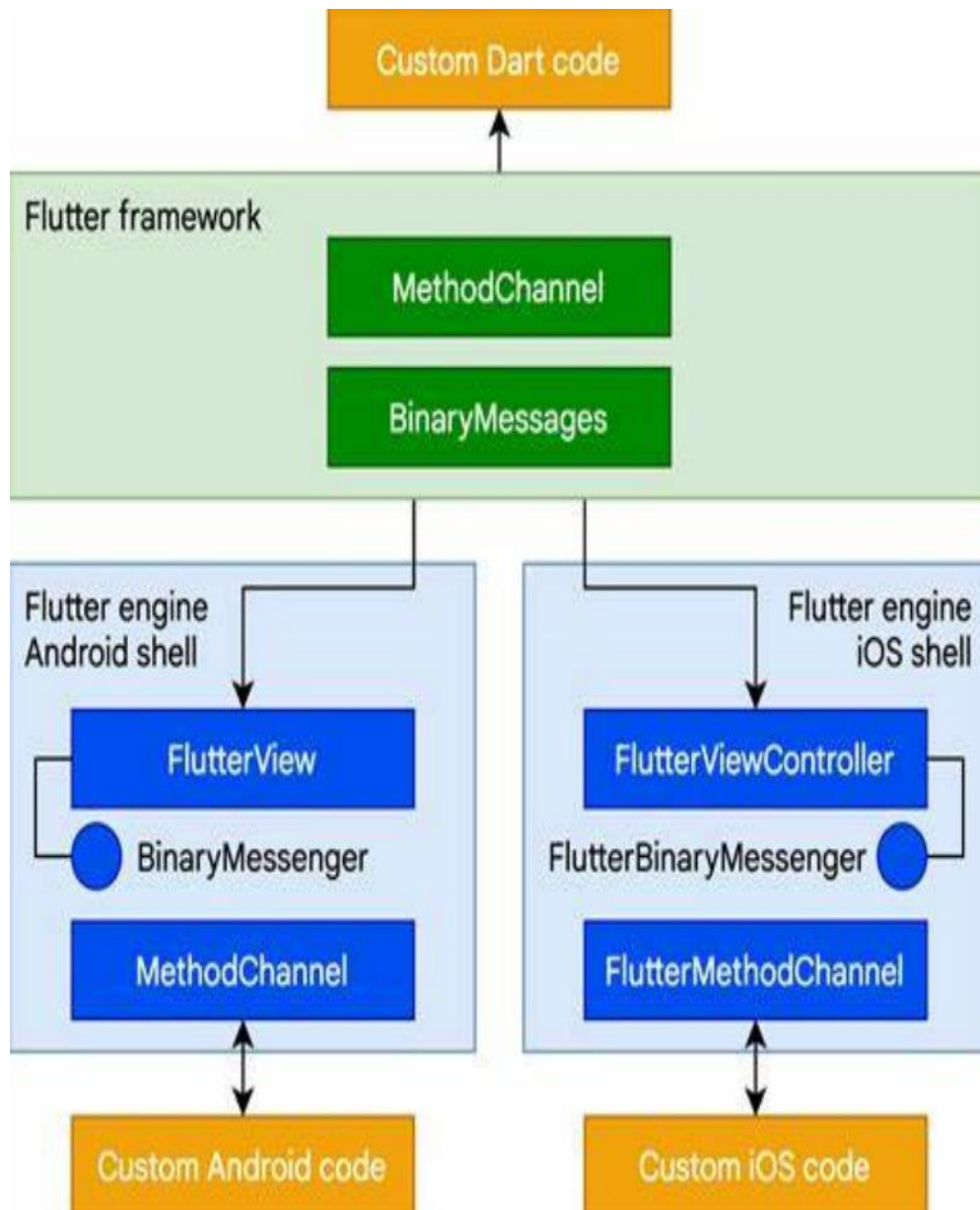


Fig:2.1: Proposed Application Architecture

1.Custom Dart Code:

This represents any custom code you write in Dart specifically for your Flutter app. It could be business logic, UI components, or any other functionality unique to your application. The custom Dart code interacts with the Flutter framework to achieve various tasks.

2.Flutter Framework:

The Flutter framework is the heart of your app. It provides a rich set of widgets, tools, and APIs for building cross-platform mobile applications.

3.MethodChannel:

The MethodChannel allows communication between Dart code and native platform code (Android or iOS). When we invoke a method on the MethodChannel from Dart, it triggers a corresponding method call on the native side.

4.BinaryMessages:

BinaryMessages is another channel for communication between Dart and native code. It enables the exchange of binary data (such as images, files, or serialized objects) between the two sides.

5.Flutter Engine Android Shell:

On the Android side, the Flutter engine runs within an Android shell.

i)FlutterView: This is the Android view that displays your Flutter UI.

ii)BinaryMessenger: It handles communication between the Flutter engine and the Android platform.

6.Flutter Engine iOS Shell:

On iOS, the Flutter engine runs within an iOS shell.

i)FlutterViewController: The iOS view controller responsible for displaying your Flutter UI.

ii)FlutterBinaryMessenger: Handles communication between the Flutter engine and the iOS platform.

7.Custom Android Code and Custom iOS Code:

These represent platform-specific code written in Java/Kotlin (for Android) and Objective-C/Swift (for iOS).You'll write custom code here to interact with native features or APIs that aren't directly accessible from Flutter.

8.Firebase:

Firestore

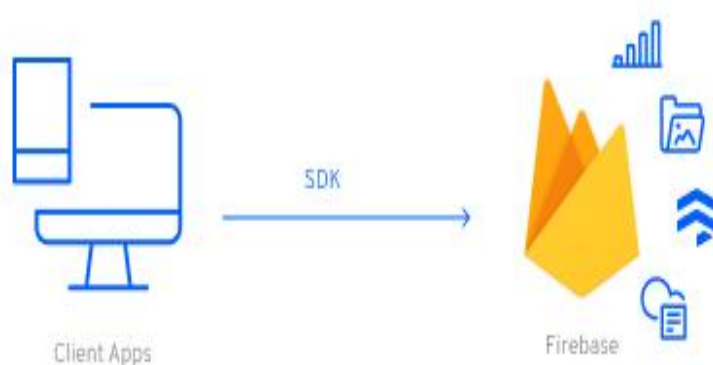


Fig:2.2: Firebase Architecture

i)Firebase Authentication:

Firebase Authentication offers ready-to-use authentication services. It integrates seamlessly with Flutter, allowing users to sign up, sign in, and manage their accounts. MFA (Multi-factor Authentication) can be enabled to enhance security by requiring additional verification steps (e.g., SMS codes, email, or biometrics).

ii)Firestore (Realtime Database):

Firestore is a NoSQL database provided by Firebase. It stores structured data in collections and documents. You can use Firestore to store user profiles, chat messages, app settings, and more.

iii)Firebase Storage:

Firebase Storage provides secure cloud storage for files (e.g., images, videos, documents). You can upload and download files directly from your Flutter app. Data-at-rest encryption ensures sensitive files are protected.

iv)Firebase Analytics:

Firebase Analytics tracks user behavior, app usage, and events. You can gain insights into user engagement, retention, and conversion rates. Use analytics data to make informed decisions about app improvements.

9.To integrate Firebase with your Flutter app:

Add the necessary Firebase plugins (e.g., `firebase_core`, `firebase_auth`, `cloud_firestore`, etc.) to your `pubspec.yaml`. Initialize Firebase in your app (usually in the main function).

2.3.FLOW DIAGRAM

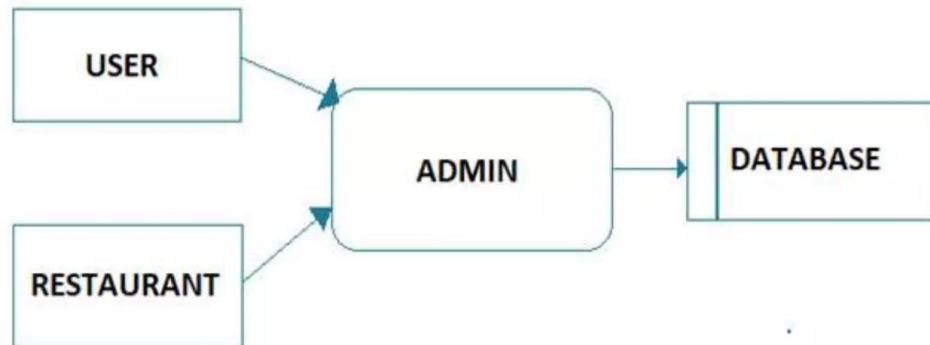


Fig:1 Context Level DFD (0 Level DFD)

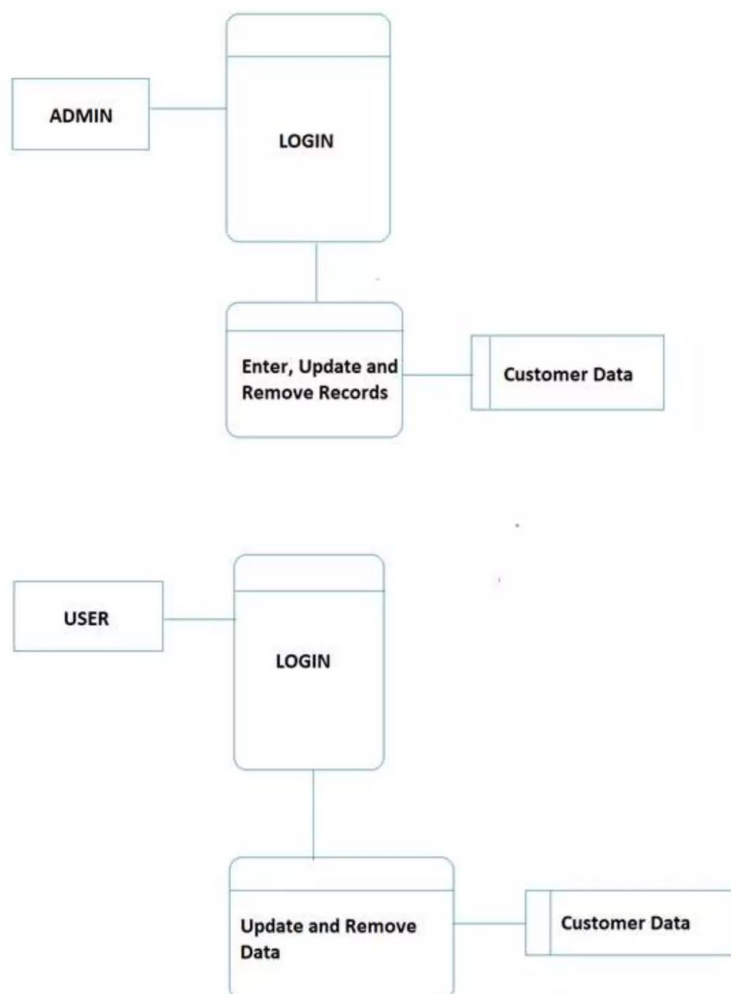


Fig:2 Level 1 DFD

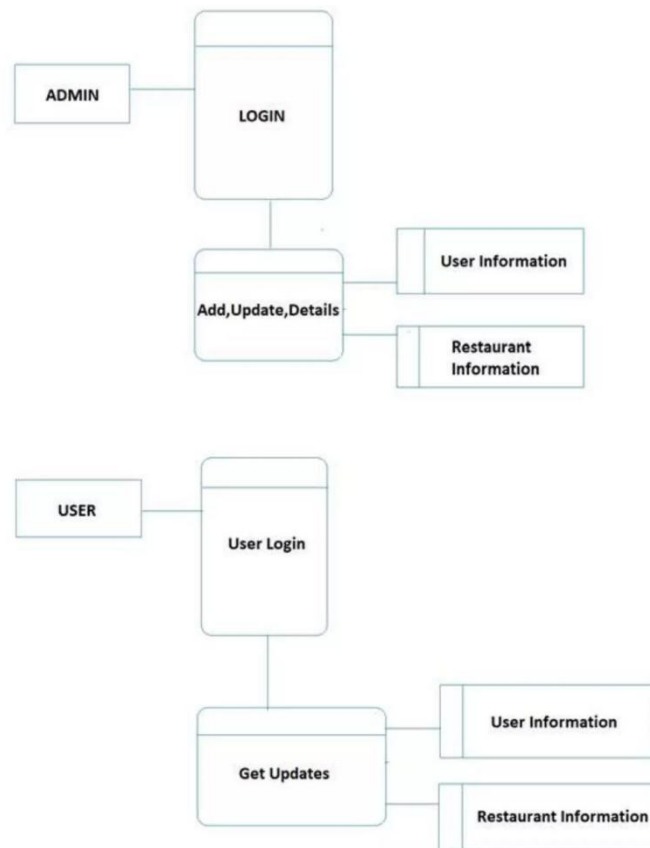


Fig:3 Level 2 DFD

The following steps are used to describe the above flow diagram:

- i) The user opens the mobile app.
- ii) The app prompts the user to log in or sign up if they don't have an account yet.
- iii) Search for restaurants based on the cuisines type, menu items, etc..
- iv) Create a cart, add menu items to the cart and order.
- v) Create their profile (onboarding) and create/refresh/add new menu items, pictures.
- vi) Receive notifications about orders placed, assigned doordasher, and update the status of order etc.
- vii) Receive notifications about the available orders in the area, from which they can choose.

2.4.MODULES

2.4.1.USER MANAGEMENT(Delishlia User App):

Users can manage their profile information, including name, contact details, and delivery addresses.Option to save multiple delivery addresses personalized access during order placement. Profile picture upload functionality for personalized user accounts.

Functionalities:

i)User Registration and Authentication: Users can register using their email address, phone number, or social media accounts. Secure authentication mechanisms, including password hashing and token-based authentication, will be implemented to ensure user account security.

ii)Privacy and Security: Implement data protection measures in compliance with data privacy regulations. Secure user data storage and transmission using encryption protocols. Regular security audits and vulnerability assessments to ensure user account security.

Algorithm:

1. User places an order through the Delishlia user app.
2. Delishlia user app sends the order details to the Delishlia server.
3. Delishlia server notifies the restaurant admin app about the new order.
4. Restaurant admin app confirms the order and updates its status.
5. Restaurant admin app assigns order to a delivery partner.

2.4.2.AUTHENTICATION MANAGEMENT:

The authentication management system for the Delishlia ecosystem ensures secure access for all users across the Delishlia app suite, including Delishlia for Customers, Delishlia Restaurant Admin, and Delishlia Delivery Partner apps. The system employs robust authentication protocols to protect user accounts and sensitive data.

Functionalities:

1. User Authentication:

Customers, restaurant owners, and delivery partners will use Delishlia Restaurant Admin and Delishlia Delivery Partner apps using unique credentials (username/email and password). Upon registration, users will be required to verify their email address or phone number to complete the registration process.

2. Token-Based Authentication:

Token-based authentication will be implemented to validate user sessions and provide secure access to the Delishlia apps. Upon successful login, users will receive a unique authentication token that will be used to authenticate subsequent requests to the server.

3. Role-Based Access Control (RBAC):

Role-based access control will be implemented to manage user permissions based on their roles (customer, restaurant admin, delivery partner). Each role will have predefined access levels to specific features and functionalities within the Delishlia app suite.

4. Two-Factor Authentication (2FA):

Optional two-factor authentication will be available for users who

wish to add an extra layer of security to their accounts. Users can enable 2FA via SMS, email, or authenticator app for enhanced account security.

5. Session Management:

The authentication system will manage user sessions securely, ensuring that users are automatically logged out after a period of inactivity. Users will have the option to manually log out of their accounts to end their sessions.

6. Integration with Delishlia Ecosystem:

The authentication management system will be integrated seamlessly with the Delishlia Restaurant Admin and Delishlia Delivery Partner apps. Users will be able to switch between the customer, restaurant admin, and delivery partner apps without the need for separate authentication.

Algorithm:

1. User enters registered email/phone number and password.
2. Delishlia verifies credentials against the database.
3. If credentials are valid, user is granted access to the app.
4. Upon successful login, Delishlia generates an authentication token.
5. This token is unique to each user session and is used for subsequent interactions with the Delishlia app.

2.4.3. USER MANAGEMENT(Delishlia Restaurant Admin App):

Enables users to update profiles, upload pictures, set availability, and manage notifications, enhancing personalization and user control.

Functionalities:

i)Admin:Full access to all features of the restaurant admin
Manager:Limited access to specific features such as menu management, order tracking, and customer feedback.

Staff:Restricted access for basic tasks like order processing and inventory management.

ii) User Profile Management: Users can update their profile information such as name, contact details, and profile picture. Admin can edit or deactivate user accounts as needed.

iii)Role-Based Access Control: Each user is assigned a role (admin, manager, staff) with specific permission. Admin has full control over user roles and permissions.

Algorithm:

1. Create a new RestaurantAdmin object with the provided details.
2. Add the restaurant admin to the list of registered admins. Return the newly registered restaurant admin.
3. Iterate through the list of registered restaurant admins.
4. Display the menu of the restaurant.
5. Add a new item to the restaurant's menu.
6. Update the details of an existing item in the menu.

2.4.4.USER MANAGEMENT(Delishlia Delivery Partner App):

Functionalities:

i)User Registration:Delivery partners register with the app using their personal details and vehicle type. Required information includes name, email address, phone number, password, and vehicle type

ii)Navigation and Route Optimization:Delivery partners have access to navigation tools to reach the delivery destination efficiently. Routes are optimized to ensure timely delivery and reduce delivery times.

iii)Logout and Session Management:Delivery partners can log out of the app to end their session securely.Sessions expire after a period of inactivity to ensure account security.

Algorithm:

1. Users log in using their registered email/phone number and password.
2. Users register with the app providing personal details.
3. Delivery partners update their availability status (available/unavailable) based on their work schedule.
4. Delivery partners have access to navigation tools to reach the delivery destination efficiently.

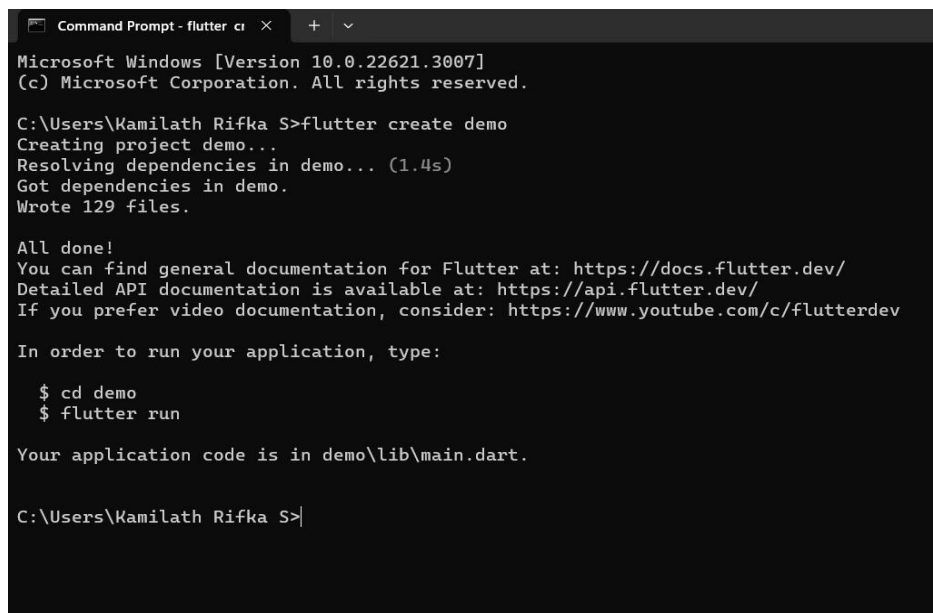
2.5.DEVELOPMENT AND DEPLOYMENT SETUP

1.Development Environment:

Flutter and Dart need to be installed on the developer's machine.
IDEs like Visual Studio Code or Android Studio for coding.

2.Create Project:

>>flutter create project_name



```
Command Prompt - flutter ci x + v
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kamilath Rifka S>flutter create demo
Creating project demo...
Resolving dependencies in demo... (1.4s)
Got dependencies in demo.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

$ cd demo
$ flutter run

Your application code is in demo\lib\main.dart.

C:\Users\Kamilath Rifka S>
```

Fig:2.4: Project Creation

3.Dependency Management:

Project dependencies specified in the pubspec.yaml file.

Key packages include:

- firebase_core for Firebase integration.
- firebase_auth for user authentication.
- cloud_firestore for real-time data synchronization.
- Other packages like get, toast, image_picker, etc., for additional functionalities.

4.Firebase Configuration:

Firebase project created and configured for the application.

Firebase Hosting used for deploying the Flutter web app.

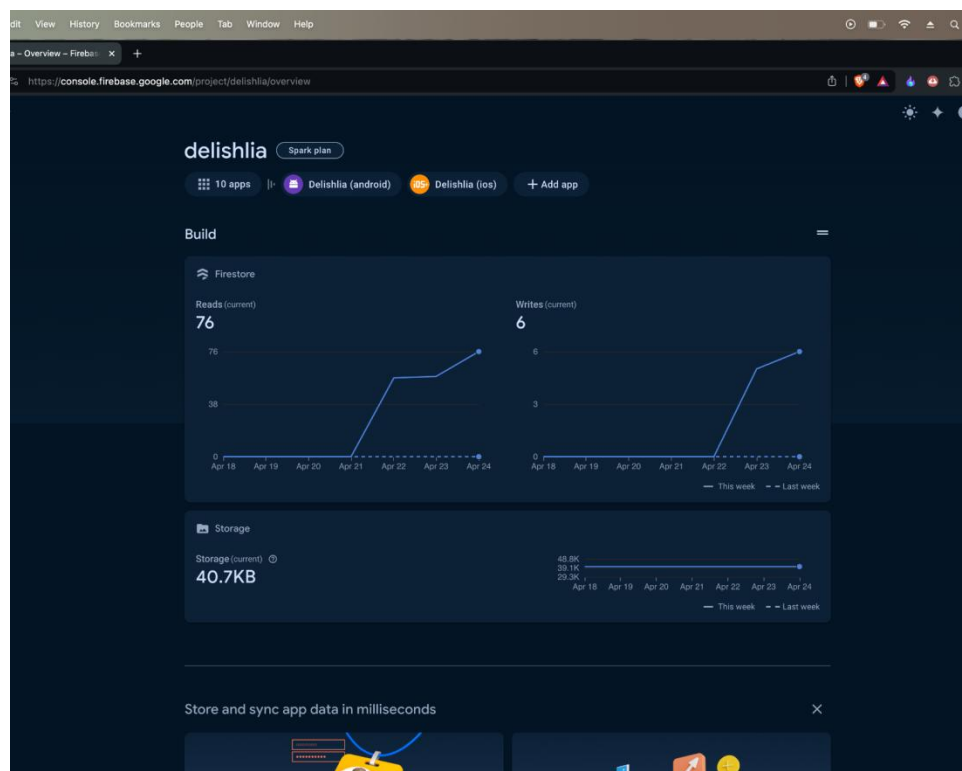


Fig:2.5: Firebase Connection

5. Firebase Authentication:

Firebase Authentication set up for secure user login and identity verification.

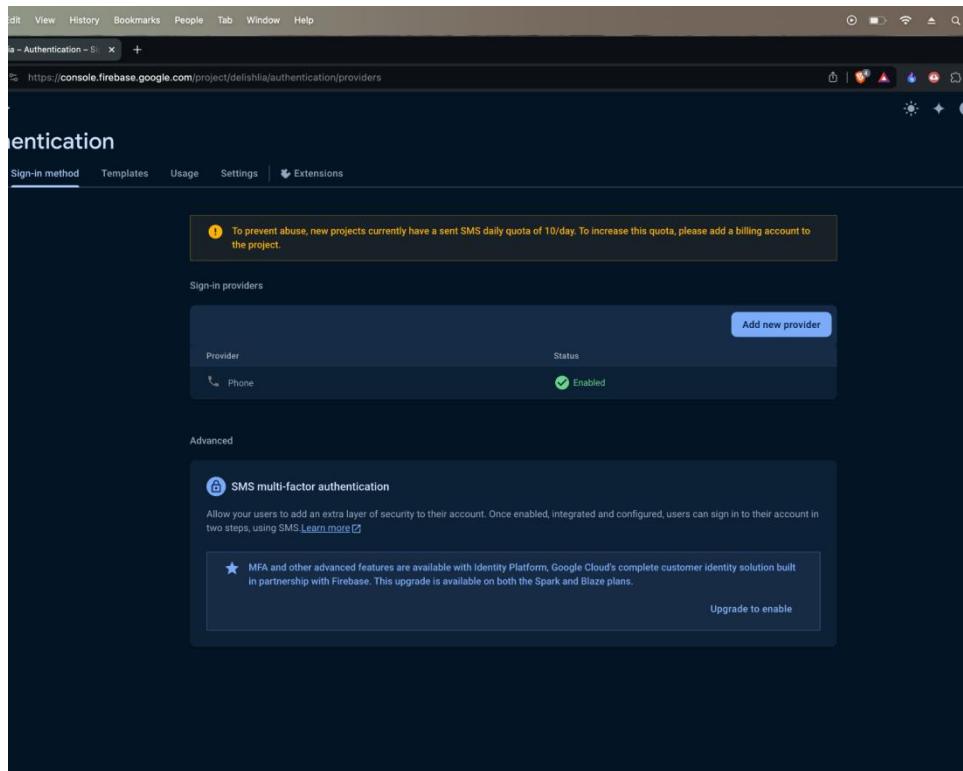


Fig:2.6: Firebase Authentication

6. Development Tools:

Flutter commands used for testing and building the application. Real-time testing on Android and iOS devices

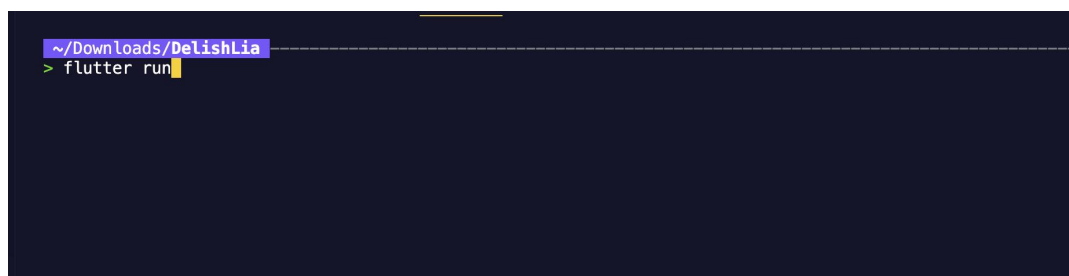


Fig:2.7: Flutter Run

7.Deployment Process:

Firebase used for hosting the web version of the application.

Seamless deployment using Flutter and Firebase commands.

8.Supported Platforms:

Cross-platform support for Android, iOS, and web.

9.Integrated Development Environment (IDE):

Developers can choose between Visual Studio Code or Android Studio for coding.

10.User Authentication:

Firebase Authentication ensures secure user login.

CHAPTER 3

RESULTS AND DISCUSSION

DART CODES:

i)main.dart: This file is typically the entry point of a Flutter application. It initializes the app, sets up necessary configurations, and runs the main application widget.

ii)loginPage.dart: This file likely contains the UI and logic for user login. It includes form fields for entering credentials and a mechanism to authenticate users.

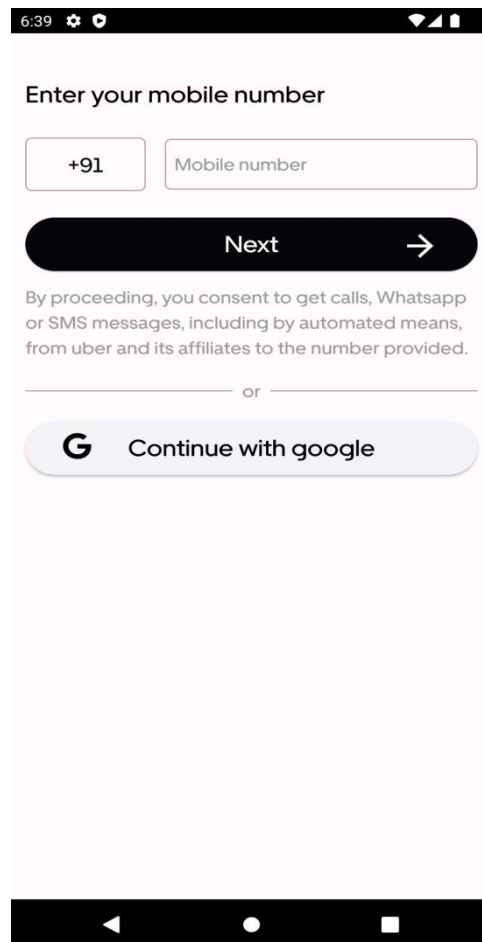
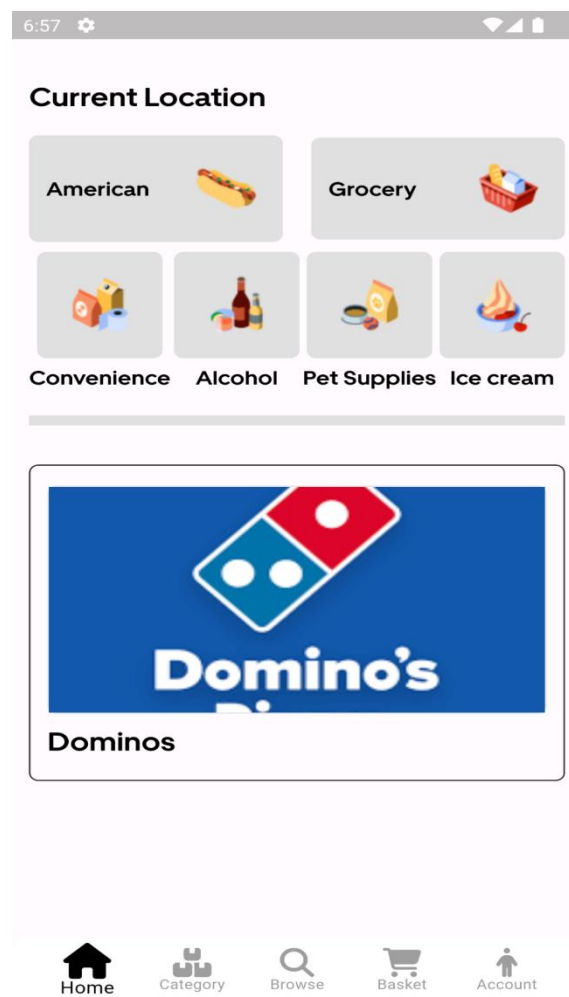


Fig: 3.1: Login Page Screen

iii)signupPage.dart: Similar to the login file, signupPage.dart is likely responsible for handling user registration. It includes form fields for users to create accounts.

iv)homePage.dart: This file represents the main content of your app after a successful login or signup. It contains messages,profilePage ,feedbackPage and mentorSelection Page with various features.

v)profilePage.dart: The file is likely responsible for displaying and editing user profiles. It includes information about the user, such as



their name, profile picture, and additional details.

Fig:3.2: Dashboard Screen Display

vi)homePage.dart: The home screen of the food delivery app, displaying nearby restaurants and search functionality.

vii)bottomNavigation.dart: The bottom navigation bar for navigating between different screens in the food delivery app.

viii)categoryScreen.dart: The screen displaying food categories for users to browse in the food delivery app.

ix)browserScreen.dart: The screen for browsing restaurants and food items in the food delivery app.

x)checkoutScreen.dart: The screen for users to review and complete their order before checkout in the food delivery app.

xi)addAddressScreen.dart: The screen for users to add a new delivery address in the food delivery app.

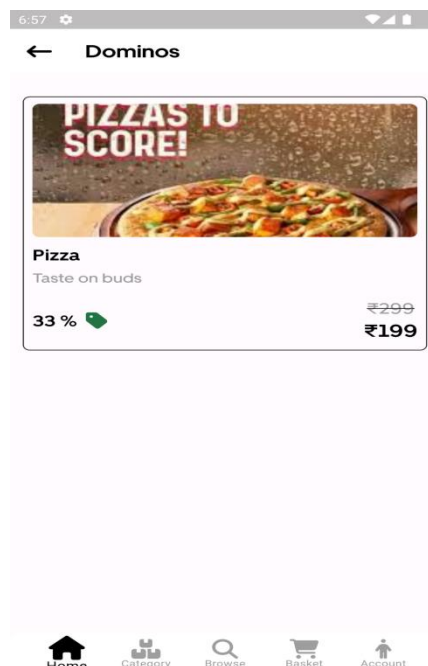


Fig:3.3: Restaurant Banner

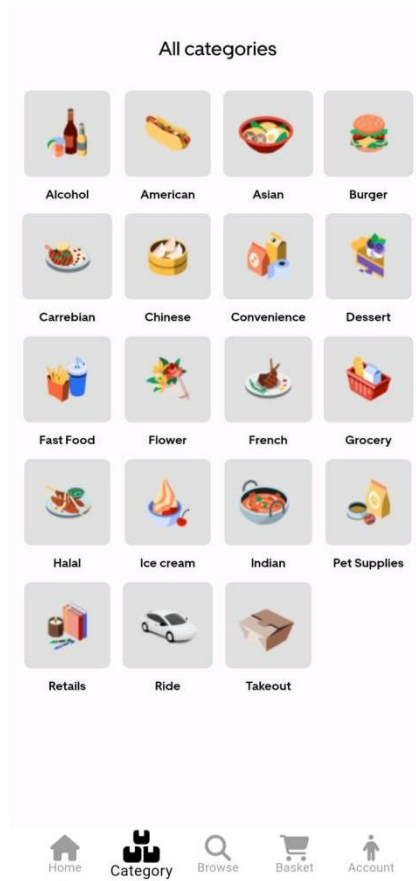


Fig:3.4:Category Screen

xiii)restaurantRegistration.dart: The screen where restaurants can register their details in the food delivery app.

xiv)menuScreen.dart: The screen where restaurant owners can add new items to their menu in the food delivery app.

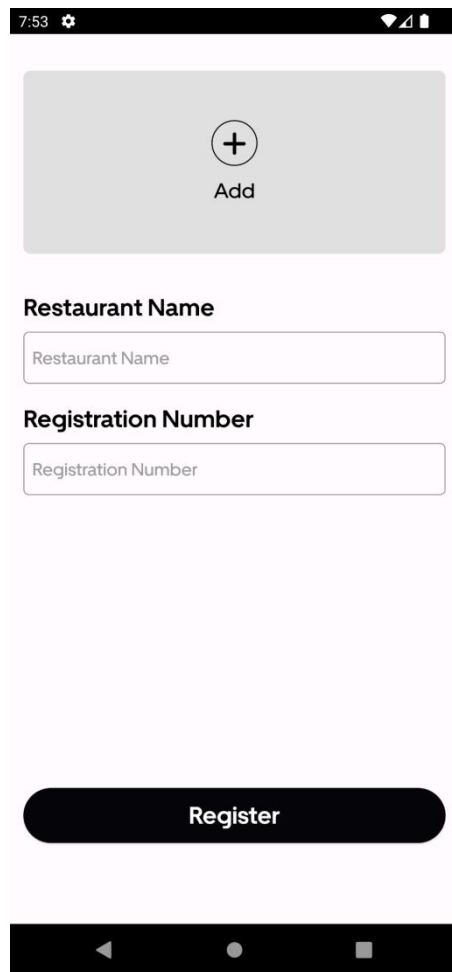
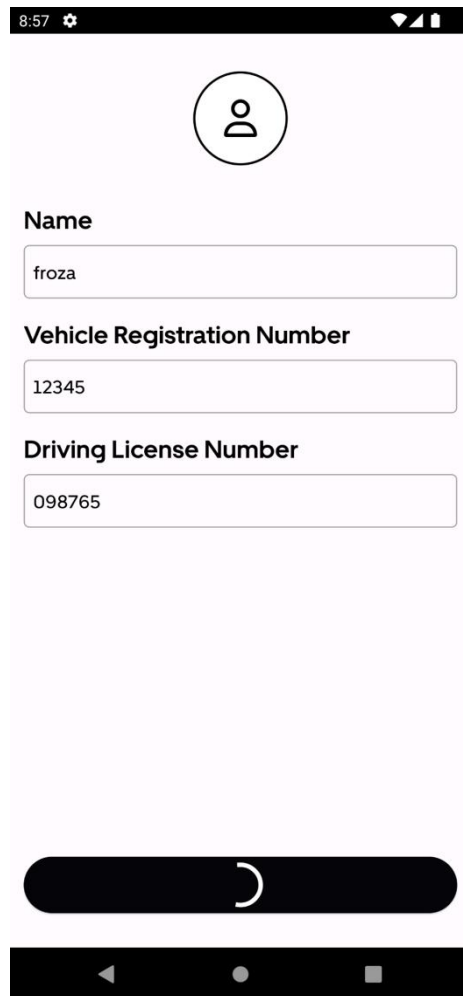


Fig:3.5 Restaurant Registration

xv)driverRegistrationScreen.dart: The screen where delivery drivers can register their details in the food delivery app.

xvi)homeScreen.dart: The screen displaying the map with restaurant locations and delivery routes in the food delivery app.



A mobile application screen for delivery partner registration. The screen has a light pink background. At the top, there is a status bar with the time 8:57, a gear icon, and signal/battery icons. Below the status bar is a circular profile icon containing a person silhouette. The form consists of three sections, each with a label and a text input field: 'Name' with the value 'froza', 'Vehicle Registration Number' with the value '12345', and 'Driving License Number' with the value '098765'. At the bottom of the form area is a dark blue rounded rectangle with a white right-pointing arrow. The very bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

8:57



Name

froza

Vehicle Registration Number

12345

Driving License Number

098765



Fig:3.6: Delivery Partner Registration Screen

FIREBASE:

i)Create a Firebase Project:

1. Go to the Firebase Console.
2. Click on "Add Project" and follow the setup instructions.

ii)Configure Flutter Project:

1. Add the Firebase configuration files to your Flutter project.
2. Follow the instructions provided in the Firebase Console to add google-services.json for Android and GoogleService-Info.plist for iOS.
3. These files contain configuration details needed for your app to connect to Firebase services.

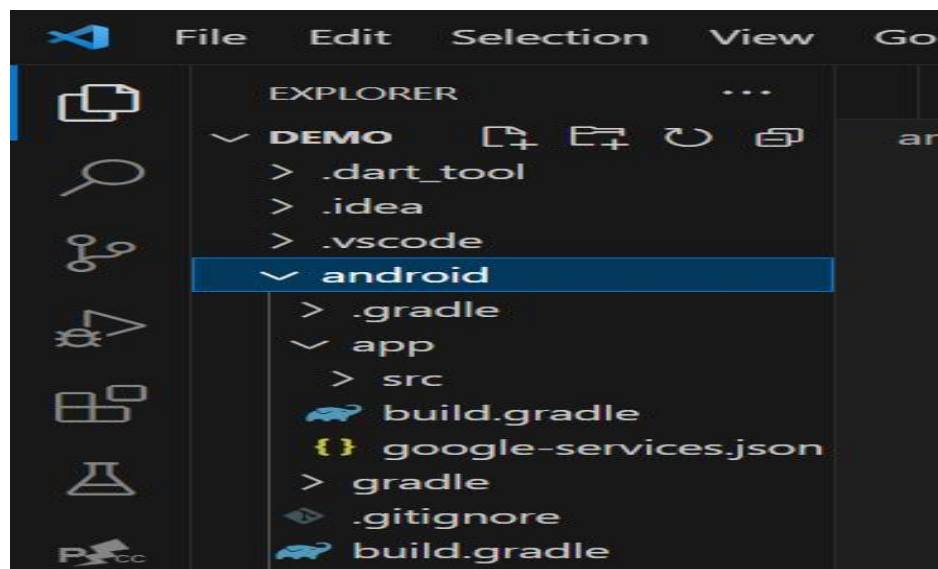


Fig:3.7: Firebase Configurations

iii)Add Firebase Dependencies:

Open the pubspec.yaml file and add the following dependencies:

yaml code:

dependencies:

firebase_core: ^2.25.3

firebase_auth: ^4.17.5

iv)Initialize Firebase:

In your main.dart or wherever you initialize your Flutter app, make sure to initialize Firebase. If you want to implement user authentication, Firebase provides the firebase_auth package. You can use it to sign in users using various methods like email/password, Google Sign-In, etc.

Dart code:



```
1  getResturantBannerImages(BuildContext context) async {
2    restaurantBannerImages = await ImageServices.getImageFromGallery(
3      context: context,
4    );
5    notifyListeners();
6  }
7
8  updateResturantBannerImagesURL(BuildContext context) async {
9    restaurantBannerImagesURL =
10     await ImageServices.uploadImagesToFirebaseStorageAndGetURL(
11       images: restaurantBannerImages,
12       context: context,
13     );
14    notifyListeners();
15  }
16
17  getResturantData() async {
18    restaurantData = await RestaurantCRUDServices.fetchRestaurantData();
19    notifyListeners();
20  }
21 }
```

Fig:3.8: Initializing Firebase

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. CONCLUSION:

In conclusion, Delishlia, along with its integrated Delishlia Restaurant Admin and Delishlia Delivery Partner apps, stands as a game-changer in the food delivery industry. With its user-centric approach and cutting-edge technology, Delishlia offers a seamless and efficient platform for users, restaurants, and delivery partners alike.

For users, Delishlia provides a convenient way to browse through a wide range of restaurants, explore diverse cuisines, and place orders effortlessly. The real-time order tracking feature ensures transparency and keeps users informed about the status of their orders, from preparation to delivery.

For restaurants, the Delishlia Restaurant Admin app offers a powerful set of tools for managing their menu, processing orders, and tracking their performance. With features such as order management, menu customization, Delishlia empowers restaurants to streamline their operations and enhance their customer service.

For delivery partners, the Delishlia Delivery Partner app offers a flexible and efficient way to manage their delivery tasks. The app provides features such as real-time order assignments, navigation tools, and earnings tracking, allowing delivery partners to optimize their routes, and provide exceptional service to users and restaurants.

4.2. FUTURE WORK

1.Enhanced User Experience::

Continuously improve the user interface and experience to make ordering food even more convenient and enjoyable for users. Introduce features such as personalized recommendations, dietary preferences, and loyalty programs to enhance user engagement and satisfaction.

2.Advanced Analytics and Insights:

Implement advanced analytics tools to provide restaurants with deeper insights into customer preferences, ordering trends, and sales performance. Optimize this data to help restaurants optimize their menus, pricing strategies, and marketing efforts to attract more customers and increase revenue.

3.Integration with Smart Technologies:

Explore integration with smart home devices, such as voice assistants and smart refrigerators, to make ordering food even more effortless for users. Introduce features such as voice-activated ordering, automated grocery replenishment, and meal planning assistance to enhance user convenience.

4.Expansion into New Markets:

Expand Delishlia's services to new cities, regions, and countries to reach a wider audience and increase market share. Partner with local restaurants, food suppliers, and delivery partners to ensure seamless operations and service quality in new markets.

5.Diversification of Services:

Diversify Delishlia's offerings beyond food delivery to include additional services such as grocery delivery, meal kits, catering, and event planning. Partner with local businesses and service providers to offer a comprehensive range of services to users.

6.Focus on Sustainability and Health:

Introduce initiatives to promote sustainable practices, such as eco-friendly packaging, reduced food waste, and support for local and organic food suppliers. Partner with health and wellness organizations to promote healthy eating habits, offer nutritional information, and support special dietary needs.

7.Continuous Innovation and Collaboration:

Foster a culture of innovation within the company and encourage collaboration with industry partners, startups, and technology providers. Explore emerging technologies such as artificial intelligence, machine learning, blockchain, and augmented reality to enhance the Delishlia experience and stay ahead of the competition.

8.Mobile App Accessibility:

Ensure the app's accessibility by incorporating features like voiceover support, larger text options, and other accessibility improvements to cater to a diverse user base.

9.Feedback and Rating System:

Implement a comprehensive feedback and rating system for both mentors and students, allowing users to provide reviews and ratings after mentorship sessions to build a reputation system.

REFERENCES

1.Journals:

i) "Enhancing Mentorship through Technology in Educational Settings"

Authors: A. Smith, B. Johnson, C. Thompson

Journal: Journal of Educational Technology & Society

Volume: 24

Year: 2021

Pages: 45-62

ii) "Collaborative Learning Platforms: A Review of Effective Features"

Authors: B. Thompson, D. Miller, E. Wilson

Journal: Computers & Education

Volume: 36

Year: 2020

Pages: 112-130

iii) "A Comprehensive Review of Flutter in Educational Application Development"

Authors: C. Lee, D. Brown, E. Clark

Journal: Journal of Educational Computing Research

Volume: 15

Year: 2019

Pages: 78-95

iv) "Bridging the Gap: Successful Models of Industry-Academia Collaboration"

Authors: R. Sharma, S. Patel, T. Nguyen

Journal: International Journal of Technology Transfer and Commercialization

Volume: 8

Year: 2022

Pages: 25-42

v) "Impact of Hackathons on Skill Development: A Case Study Analysis"

Authors: L. Chen, M. Wang, K. Li

Journal: International Journal of Innovation and Technology Management

Volume: 10

Year: 2020

Pages: 145-162

2. Books:

i) "Building Virtual Communities for Knowledge Exchange"

Author: M. Davis

Publisher: Wiley

Year: 2021

Pages: 180

ii)"Flutter for Mobile App Development in Education"

Author: K. Patel

Publisher: Packt Publishing

Year: 2019

Pages: 320

iii)"Fostering Collaboration between Academia and Industry"

Author: S. Williams

Publisher: Springer

Year: 2021

Pages: 200

iv)"Hackathons and Learning: A Practical Approach"

Author: E. Rodriguez

Publisher: Routledge

Year: 2020

Pages: 150

3.Internet Articles:

i)"How Technology is Transforming Mentorship in Education"

Author: John Doe

Website: Education Technology Insights

URL: <https://www.educationtechnologyinsights.com/news/how-technology-is-transforming-mentorship-in-education-nwid-1278.html>

Publication Date: March 15, 2023

ii)"The Role of Collaborative Learning Platforms in Modern Education"

Author: Jane Smith

Website: EdSurge

URL: <https://www.edsurge.com/news/the-role-of-collaborative-learning-platforms-in-modern-education>

Publication Date: June 5, 2023

iii)"Innovative Use of Flutter in Educational App Development"

Author: Michael Johnson

Website: Medium

URL: <https://medium.com/@mjohnson/innovative-use-of-flutter-in-educational-app-development-39a7168b45e7>

Publication Date: July 20, 2023

iv)"Strategies for Successful Industry-Academia Collaboration"

Author: Rachel Andrews

Website: Harvard Business Review

URL: <https://hbr.org/2023/09/strategies-for-successful-industry-academia-collaboration>

Publication Date: September 10, 2023

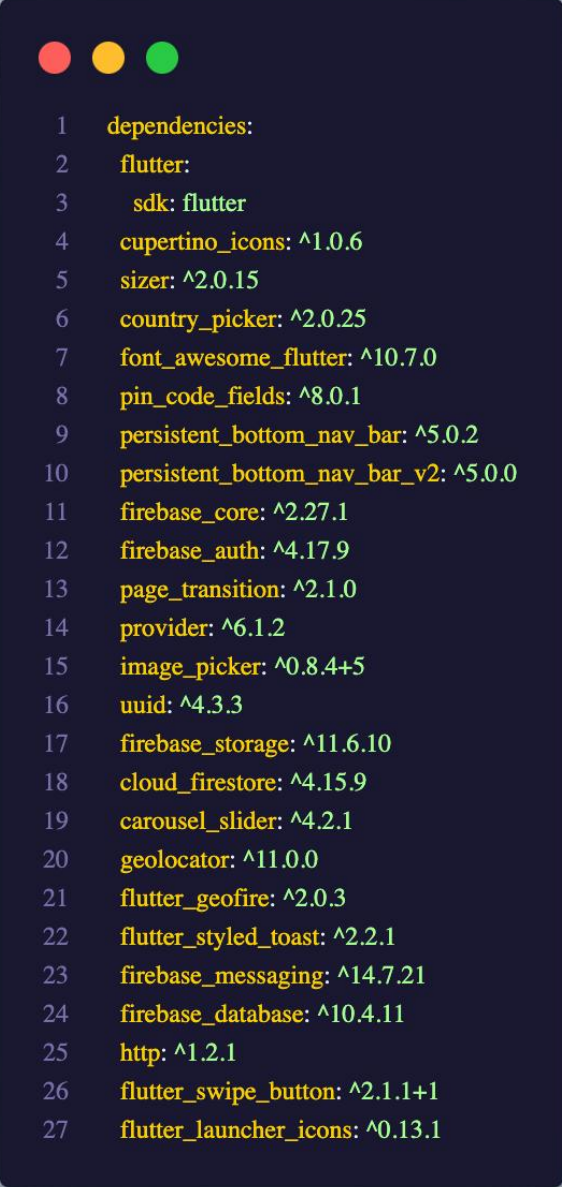
APPENDICES

A.SAMPLE CODE

i)Main Code:

```
1  runApp(const Delishlia());
2  }
3
4  class Delishlia extends StatelessWidget {
5    const Delishlia({super.key});
6    @override
7    Widget build(BuildContext context) {
8      return Sizer(builder: (context, _, __) {
9        return MultiProvider(
10         providers: [
11           ChangeNotifierProvider<MobileAuthProvider>(
12             create: (_) => MobileAuthProvider(),
13           ),
14           ChangeNotifierProvider<ResturantProvider>(
15             create: (_) => ResturantProvider(),
16           ),
17           ChangeNotifierProvider<FoodProvider>(
18             create: (_) => FoodProvider(),
19           ),
20           ChangeNotifierProvider<DeliveryPartnerProvider>(
21             create: (_) => DeliveryPartnerProvider(),
22           ),
23         ],
24         child: MaterialApp(
25           debugShowCheckedModeBanner: false,
26           title: 'Flutter Demo',
27           theme: ThemeData(),
28           home: const SignInLogicScreen(),
29         ),
30       );
31     });
```

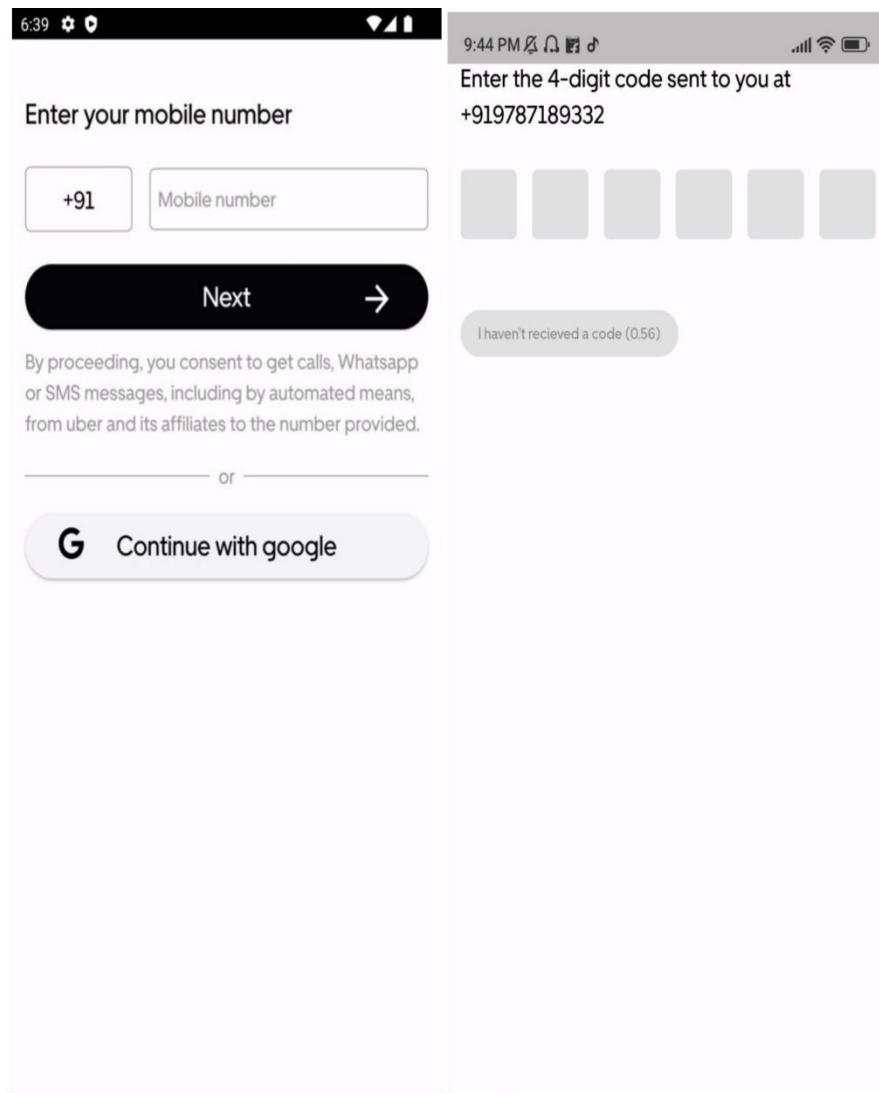
ii)pubspec.yaml



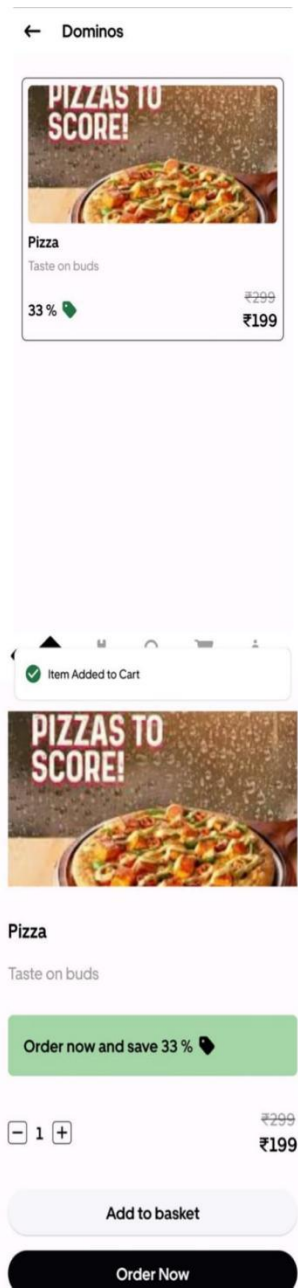
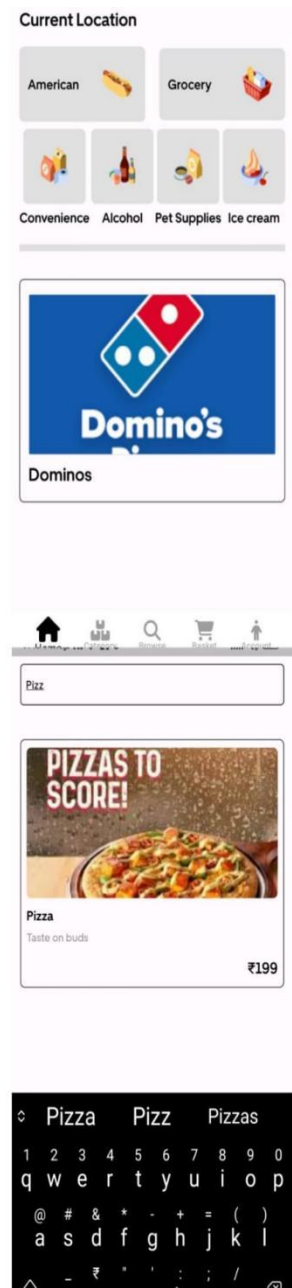
```
1  dependencies:
2    flutter:
3      sdk: flutter
4    cupertino_icons: ^1.0.6
5    sized: ^2.0.15
6    country_picker: ^2.0.25
7    font_awesome_flutter: ^10.7.0
8    pin_code_fields: ^8.0.1
9    persistent_bottom_nav_bar: ^5.0.2
10   persistent_bottom_nav_bar_v2: ^5.0.0
11   firebase_core: ^2.27.1
12   firebase_auth: ^4.17.9
13   page_transition: ^2.1.0
14   provider: ^6.1.2
15   image_picker: ^0.8.4+5
16   uuid: ^4.3.3
17   firebase_storage: ^11.6.10
18   cloud_firestore: ^4.15.9
19   carousel_slider: ^4.2.1
20   geolocator: ^11.0.0
21   flutter_geofire: ^2.0.3
22   flutter_styled_toast: ^2.2.1
23   firebase_messaging: ^14.7.21
24   firebase_database: ^10.4.11
25   http: ^1.2.1
26   flutter_swipe_button: ^2.1.1+1
27   flutter_launcher_icons: ^0.13.1
```

B.SCREEN SHOTS

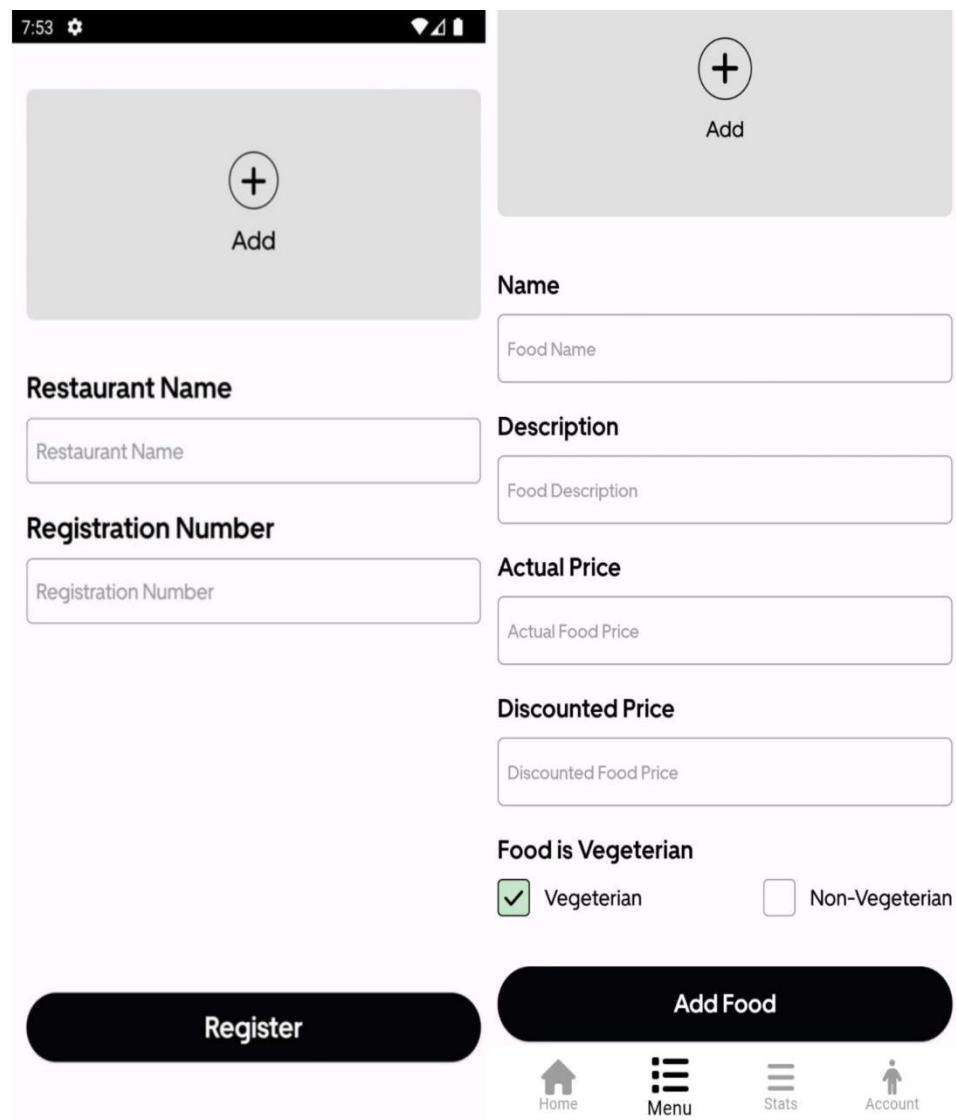
i)Login page And Signup Page



ii) Homepage & Navigation bar



iii) Restaurant Registration



The image shows a mobile application interface for restaurant registration. At the top, a status bar displays the time 7:53, a gear icon for settings, and signal/battery indicators. The main interface is divided into two columns. The left column contains three input fields: 'Restaurant Name', 'Registration Number', and 'Add' (with a plus icon). The right column contains five input fields: 'Name', 'Description', 'Actual Price', 'Discounted Price', and 'Food is Vegetarian' (with a checked checkbox for 'Vegetarian' and an unchecked checkbox for 'Non-Vegetarian'). At the bottom, there are two large buttons: 'Register' and 'Add Food'. A navigation bar at the very bottom contains four icons: 'Home', 'Menu', 'Stats', and 'Account'.

7:53

Add

Restaurant Name

Restaurant Name

Registration Number

Registration Number

Name

Food Name

Description

Food Description

Actual Price

Actual Food Price

Discounted Price

Discounted Food Price

Food is Vegetarian

☒ Vegetarian ☐ Non-Vegetarian

Register

Add Food

Home Menu Stats Account

iv)Delivery Partner Registration



Name

froza

Vehicle Registration Number

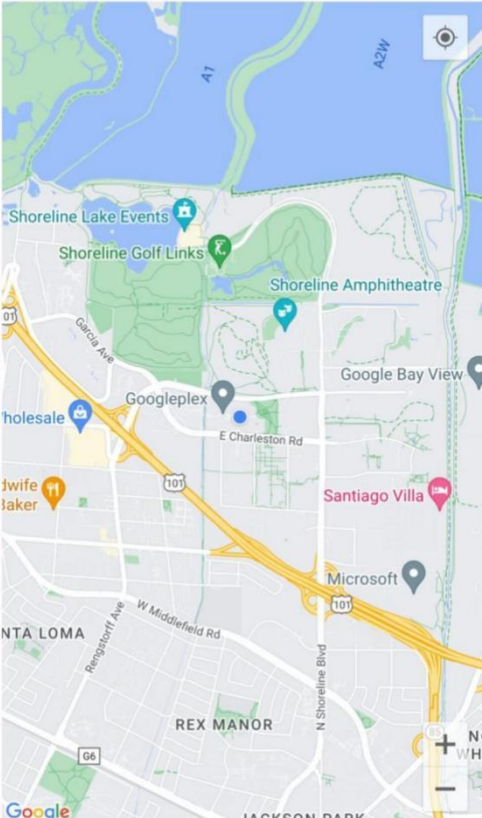
12345

Driving License Number

098765



**Go Online**



**Home**

**History**

**Account**