INTEGRATED CULINARY SERVICES PLATFORM

PROJECT REPORT

Submitted by

MOHAMED IBRAHIM AFRITH M

Register No: 23MCA035

in the partial fulfillment for the award of the

degree of

MASTER OF COMPUTER APPLICATIONS

in

DEPARTMENT OF COMPUTER APPLICATIONS

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Chennai)

COIMBATORE 641049

APRIL 2025

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Chennai)

Department of Computer Applications

PROJECT WORK

April 2025

This is to certify that the project entitled

Integrated Culinary Services Platform

is a Bonafide record of project work done by

MOHAMED IBRAHIM AFRITH M

Register No: 23MCA035

of MCA (Computer Applications) during the year 2024 – 2025

Project Guide

Head of the Department

Submitted for the Project Viva – Voce examination held on 23-04-2025

Internal Examiner

External Examiner

DECLARATION

I affirm that the project work **titled Integrated Culinary Services Platform** being submitted in partial fulfilment for the award of **Master of Computer Applications** is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

(Signature of the candidate)

MOHAMED IBRAHIM AFRITH M

23MCA035

I certify that the declaration made above by the candidate is true.

(Signature of the Guide)
Dr. V. VIJILESH
ASSOCIATE PROFESSOR AND HEAD
DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

ROCKG MICRO

ROCKG MICRO TECHNOLOGY (I) PVT. LTD.

CIN: U74210TN2004PTC052989 GST: 33AACCK3693C1ZS

Internship Offer Letter

06/Dec/2024

To
Mohamed Ibrahim Afrith M, (23MCA035)
M.C.A
KUMARAGURU COLLEGE OF TECHNOLOGY,
Coimbatore

Dear Mohamed Ibrahim Afrith M,

We are pleased to offer you an internship at our company in the Dev-Intern department at our Trichy office. Your internship shall commence on 11/Dec/2024 and shall end in Apr 2025 (Five months). The terms and conditions of your internship with the Company are set forth below:

1. Subject to your acceptance of the terms and conditions contained herein, your project and responsibilities during the Term will be determined by the supervisor assigned to you for the duration of the internship.
2. Your timings will be from 10 AM to 6 PM, Monday to Friday. Please be sure to bring the required documents with you on your first day to complete your profile.
3. Optionally you need to sign a confidentiality agreement with the company before you commence your internship.
4. The internship cannot be construed as an employment or an offer of employment with ROCKG MICRO TECHNOLOGY.

Kindly confirm your acceptance of the terms of this offer by 11/Dec/2024 failing which, we have the right to cancel the internship. We look forward to having you on our team! If you have any questions, please feel free to reach out to us.

Sincerely,

Madhavan JV
(For ROCKG Micro Technology)

ROCKG MICRO TECHNOLOGY (I) PVT. LTD. TRICHY

Accepted by,

Signature
Date: 07-12-2024
Doc Ref: RGM241202

East Side, First Floor, 33, Bharathidasan Salai (Promenade Road)
Cantonment, Tiruchirappalli, Tamil Nadu 620001
+91 72000 76254
+91 431 2761158
www.rockgmicrotech.com
admin@rockgmicrotech.com

ACKNOWLEDGEMENT

I thank almighty who has been with me through every walk of life, guarding me and showering the blessings throughout the endeavor to put forth my dissertation.

I sincerely thank **SHRI. SHANKAR VANAVARAYAR**, Joint Correspondent, Kumaraguru College of Technology, for providing the ample facilities to carry out this project.

I sincerely thank **Dr. M. EZHILARASI**, Principal, Kumaraguru College of Technology, for giving me an opportunity to undertake this full project work.

I am grateful to **Dr. V. VIJILESH**, Associate Professor and Head, Department of Computer Applications, for his continuous encouragement and support during the course of this project.

I am obliged to **Dr. S. HAMEED IBRAHIM**, Assistant Professor - III, Department of Computer Applications, Project Coordinator, for his guidance, support, valuable suggestions and great encouragement during the course of this project.

I sincerely thank my guide **Dr. V. VIJILESH,** Associate Professor and Head, Department of Computer Applications, Kumaraguru College of Technology, for his sincere advice, thought provoking discussions and immense help throughout the project and encouragement given by him.

I wish to thank my parents, friends and well-wishers who have been a catalyst throughout the development of the project work. Finally, I would like to mention here that I am greatly indebted to each and everybody who has been associated with my project at any stage but whose name does not find a place in this acknowledgment.

TABLE OF CONTENT

ABSTRACT

Objective: To develop a digital solution for efficient food court management that enhances user experience through secure authentication, intuitive navigation, and seamless menu and order management.

Introduction: The **Integrated Culinary Services Platform** addresses the inefficiencies of traditional token-based systems. By digitizing the food court operations, this project aims to provide a modern, user-friendly platform for customers and vendors. The application will streamline the process of browsing menus, placing orders, and tracking order statuses, offering a robust foundation for future integrations like online payments and feedback systems.

Key Features:

1. **User Authentication:** Secure sign-up and login using Firebase Authentication.
2. **Menu Navigation:** Interactive modules for browsing menu items with detailed descriptions.
3. **Order Management:** Simplified process for placing and tracking orders.
4. **Main Tab Navigation:** Bottom navigation bar for seamless access to key features.
5. **Scalability:** Designed to incorporate future enhancements like payment systems and user feedback.

Technical Specifications:

- **Frontend:** Flutter framework with Dart for cross-platform compatibility.
- **Backend:** Firebase Firestore for real-time database and Firebase Authentication.
- **Platform:** Supports Android, iOS, and Web.
- **Database Design:** Optimized collections for users, food items, and orders.

Expected Outcomes:

1. Digitized and efficient food court operations.
2. Enhanced customer experience through intuitive UI/UX design.
3. A centralized system for menu browsing, order placement, and tracking.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATIONAL PROFILE

ROCKG Micro Technology, based in Trichy, Tamil Nadu, is a prominent technology solutions provider with a rich history of delivering high-quality services. Established over two decades ago, the company specializes in mobile product development, custom software solutions, and business process transformation services. Their goal is to assist businesses across various sectors by leveraging technology to optimize operations and enhance productivity. With a diverse portfolio of services, ROCKG Micro Technology has catered to industries such as banking, financial services, healthcare, hospitality, travel, and more.

The company offers a wide range of innovative products and services tailored to meet the unique needs of their clients. One of their flagship products, Smartfuel, is designed specifically for petrol bunks, handling tasks such as sales, purchase, inventory, administration, and accounting. Another noteworthy product, SmartSMSOne, is an intuitive software application that simplifies the process of handling short messages, allowing for easy integration with various data sources. For the automobile industry, ROCKG Micro Technology developed Smartshop, a client-server application that helps manage product sales, service, and parts movements. Additionally, their SmartHP software caters to the hire purchase industry, streamlining operations and automating processes to boost efficiency.

In addition to their product development capabilities, ROCKG Micro Technology excels in offering comprehensive services such as web development, mobile application development, and customized software solutions. Their web development services are focused on creating engaging and user-friendly websites that provide a seamless online experience for businesses. With their mobile development expertise, they build high-performance mobile apps that integrate advanced technologies to streamline business operations. The company also provides tailored software solutions, which are designed to address specific client requirements, ensuring that businesses operate smoothly and effectively. This includes offering services related to XBRL compliance and other industry-specific software needs.

Operating from their headquarters in Trichy, the company maintains a commitment to delivering exceptional service to clients. Their office is located at First Floor, KRT Building, 33 Bharathidasan Salai (Promenade Road), Cantonment, Trichy – 620 001, Tamil Nadu, India. ROCKG Micro Technology operates Monday to Friday, from 9 AM to 7 PM IST. Clients can

reach out to them through phone numbers +91 72000 76254 or +91-431-2761158, or via email at hr@rockgmicrotech.com for inquiries and support.

1.2 OBJECTIVES

The **Integrated Culinary Services Platform** is developed with the primary objective of digitizing and enhancing the overall food ordering and dining experience within a closed food court environment. It provides a centralized, user-friendly mobile application that bridges the gap between customers, restaurant vendors, and food court management.

The application allows **users (customers)** to conveniently browse multiple restaurant menus, view categorized food items, place orders, and make payments seamlessly through the app. It enhances customer satisfaction by offering real-time order tracking, digital cart management, and a responsive, intuitive interface.

For **restaurant vendors**, the app offers an efficient platform to showcase food categories, manage available products, update pricing, track orders, and monitor daily sales. Each vendor is provided with a dedicated interface to streamline operations and engage directly with customers.

Developed using **Flutter**, the app ensures cross-platform compatibility with consistent performance across Android and iOS devices. **Firebase Firestore** serves as the cloud database for real-time data management, while **Firebase Storage** is integrated to handle restaurant images, product thumbnails, and user-uploaded media assets.

Firebase Authentication is used to securely manage login and role-based access for customers, vendors, and administrators. The system also supports **payment gateway integration (such as Google Pay)** to facilitate secure and convenient online payments.

This mobile-first solution is designed to provide a **scalable, modular, and real-time digital infrastructure** for managing food court operations, improving vendor efficiency, and delivering a smooth and modern dining experience for users.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Traditional food ordering in food courts typically relies on physical queues, manual order taking, and token-based pickup systems. While some restaurants use digital billing software or POS systems, most food courts lack a unified digital ecosystem that connects all vendors under a centralized platform. This leads to inefficiencies such as long waiting times, communication gaps between customers and vendors, difficulty in tracking orders, and limited visibility into available menu items.

Moreover, customers must physically visit each food stall to view menus, compare prices, and place orders. This not only hampers convenience but also discourages users from exploring all food options. For vendors, managing orders manually often results in errors, slower service, and difficulty tracking sales or analysing customer behaviour. There is no digital mechanism to view peak order hours, monitor out-of-stock items, or receive real-time order updates.

From a management standpoint, it becomes challenging to maintain records, track performance across vendors, and ensure a smooth operation. There is also no centralized way to promote offers, collect feedback, or implement loyalty programs. Additionally, the lack of integrated payment systems creates friction for customers who prefer cashless transactions.

With the rise of smartphones and increasing digital adoption, there is a growing need for a comprehensive system that can modernize food court operations, improving the experience for both vendors and customers.

2.2 PROPOSED SYSTEM

The **Integrated Food Court Management Application** is developed to digitize and centralize all aspects of food court operations. Designed using **Flutter** for a cross-platform mobile experience and **Firebase** as the backend infrastructure, the app offers a streamlined and interactive interface for **customers, vendors, and administrators**.

Customers can browse menus from multiple restaurants, view food items categorized by type (e.g., burgers, fries, beverages), and place orders directly through the mobile app. The app supports **real-time order tracking**, **Wishlist**, and **cart functionality**, allowing customers to personalize and manage their food preferences efficiently. They can also make secure payments using integrated gateways like **UPI** or **PayPal**, eliminating the need for cash transactions.

Vendors are given a dedicated dashboard where they can upload images, categorize their items, update availability, and manage incoming orders. They receive instant notifications on new orders, allowing them to prepare food efficiently and reduce wait times.

Administrators can manage user accounts, review performance metrics, moderate content, and monitor vendor activity through a centralized panel. They can push promotional banners, handle customer feedback, and manage overall platform maintenance.

Additional features include:

- **Firebase Firestore** for real-time data syncing.
- **Firebase Storage** for storing food images, banners, and restaurant branding assets.
- **Role-based authentication** using **Firebase Auth** to securely separate users, vendors, and admins.
- **Offline fallback UI**, shimmer loaders, and intuitive navigation to ensure a smooth UX.

The proposed solution aims to deliver a **scalable, user-friendly, and real-time** food ordering system that improves the food court experience, enhances vendor performance, and simplifies management operations.

CHAPTER 3

SYSTEM SPECIFICATION

A System Requirements Specification outlines the necessary hardware and software components required for the successful development, deployment, and operation of the proposed project. For the **Integrated Culinary Services Platform**, which is built using Flutter and Firebase, the following system specifications are recommended to ensure optimal performance and seamless functionality across devices.

3.1 HARDWARE REQUIREMENTS

Component	Specification
RAM	4 GB or higher
Hard Disk	256 GB or higher (SSD recommended)
Processor	Intel i3 or above / Apple M1 or higher
Internet	Stable broadband connection (minimum 5 Mbps)

3.2 SOFTWARE REQUIREMENTS

Component	Specification
Operating System	Windows / macOS / Linux
IDE	Android Studio / Visual Studio Code
Mobile SDK	Flutter SDK (latest stable version)
Programming Language	Dart
Backend	Firebase (Firestore, Firebase Auth, Storage)
Emulator/Devices	Android Emulator / Physical Android or iOS Devices
Browser (for Web debug)	Google Chrome / Edge
Package Manager	Dart Pub

CHAPTER 4

SOFTWARE DESCRIPTION

The Integrated Food Court Management Application is a mobile-first system that digitizes and simplifies the process of food ordering, vendor management, and real-time service tracking across a multi-vendor food court. The application has been architected to offer high usability, real-time responsiveness, and modularity, making it suitable for deployment in campus environments, malls, or commercial food courts.

The application is divided into three main layers:

- **Frontend (User Interface)**
- **Backend (Logic & Services)**
- **Database and Storage (Persistent Data and Media)**

4.1 FRONTEND

The frontend is developed using **Flutter**, a cross-platform UI toolkit by Google. It ensures consistent performance and native-like user experiences on both Android and iOS platforms from a single codebase.

4.1.1 Flutter

Flutter offers a reactive, widget-based structure which simplifies UI construction. Key screens like restaurant listings, product menus, cart, order history, and chatbot assistant (Délix) are built using reusable components, allowing scalability and efficient state management via **GetX**.

Flutter’s hot reload and built-in Material and Cupertino design libraries were utilized to build an intuitive interface. UI components were styled with consistency in padding, layout, and shadows to enhance user accessibility.

4.1.2 GetX

The application employs **GetX** for state management, dependency injection, and routing. It helps manage complex flows like cart updates, Wishlist toggles, and chatbot interaction without the need for bulky boilerplate.

4.1.3 UI Libraries & Tools

- **Carousel Slider**: Used in promotional banner sliders.
- **Shimmer Effects**: Added for skeleton loaders during data fetch.
- **Iconsax**: For consistent and modern icons across the app.
- **ReadMoreText**: To elegantly expand/collapse long descriptions.

4.2 BACKEND

The backend is built on **Firebase**, leveraging its suite of services to handle authentication, real-time data operations, file storage, and notifications.

4.2.1 Firebase Authentication

User authentication (registration, login) is handled securely via **Firebase Auth**, supporting email/password login and validation. The authentication logic also ensures that specific user data is only accessible based on roles (e.g., customer or admin).

4.2.2 Cloud Firestore

Firestore, Firebase's NoSQL cloud database, is used for storing:

- Restaurant details and menus
- User profiles and cart data
- Orders and delivery status
- Wishlist and chatbot messages

Firestore provides real-time synchronization, ensuring that any changes (like cart updates or order progress) are reflected instantly in the app.

4.2.3 Firebase Storage

Firebase Storage handles image uploads for:

- Product thumbnails
- Restaurant banners
- User profile images

Uploaded images are stored as URLs, which are later accessed and rendered in the UI via `NetworkImage()` or `CachedNetworkImage`.

4.2.4 Firebase Cloud Functions (Planned)

Future iterations will integrate **Cloud Functions** for server-side tasks like:

- Sending real-time order status notifications
- Payment webhook listeners
- Analytics logging

4.3 DATABASE

The project uses **Firebase Firestore** for scalable and structured document-based data storage.

4.3.1 Data Collections

- **Users**: Stores user profiles, roles, phone, and favourites.
- **Restaurants**: Contains food court vendor data and categories.
- **Products**: Items offered under each restaurant with fields like name, image URL, price, and material.
- **Orders**: Cart and order tracking including delivery status.
- **Chatbot Logs**: Stores user queries and bot replies to assist in feedback loops.

4.3.2 Real-time Updates

Firestore enables live UI updates whenever data changes. For example, once a user adds a product to the cart or places an order, all associated screens reflect the changes instantly.

4.3.3 Firebase Rules & Security

Firebase rules are applied to enforce access control, ensuring users only access their own data, and restricting write access for non-admin roles to critical paths like product or restaurant updates.

This software stack ensures a robust, scalable, and real-time experience for users across customer, admin, and service roles in a food court ecosystem.

CHAPTER 5

PROJECT DESCRIPTION

5.1 PROJECT DEFINITION

The **Integrated Food Court Management Application** is a comprehensive digital solution aimed at streamlining the ordering and management experience within a food court environment. It allows users to browse restaurant menus, place orders, and interact with the system in real time. The application is built to enhance customer convenience, reduce wait times, and optimize restaurant operations using modern mobile technology.

The system supports multiple stakeholders, including customers, restaurant staff, and administrators. Customers can explore menus, add items to their cart, and complete orders digitally, while restaurant staff receive and manage these orders efficiently. An admin panel is also integrated to monitor and manage restaurants, menus, users, and performance metrics.

Built using **Flutter** for cross-platform compatibility and **Firebase** for real-time data synchronization, the app offers a seamless experience across both Android and iOS devices. Firebase Firestore is used for dynamic database management, and Firebase Storage handles media files such as restaurant images and menu photos.

5.2 PROJECT OVERVIEW

The application provides a mobile-first interface where users can browse various restaurants within the food court. Each restaurant has its own virtual space showcasing its menu, categorized by cuisines (e.g., South Indian, Chinese, Beverages). Users can add items to their cart, view a summary of their orders, and place them directly through the app.

The chatbot module enhances user engagement by helping with common queries, navigating menus, or giving recommendations. A persistent cart widget ensures users can access their current order from any screen. Role-based navigation enables customers, restaurant owners, and admins to access different functionalities securely.

All user interactions, including authentication, ordering, and restaurant management, are synchronized using Firebase’s real-time database and authentication services.

5.3 MODULES DESCRIPTION

5.3.1 User Authentication and Role Management

Users sign up and log in via **Firebase Authentication**. During registration, they choose a role such as "Customer," "Restaurant Owner," or "Admin." Role-based routing ensures each user only accesses their respective dashboard and features.

5.3.2 Restaurant Listings and Menu Management

Each restaurant is listed on the home screen with dynamic cards showing images, names, and short descriptions. Tapping on a restaurant opens its full-screen view with menu categories such as "All," "South Indian," "Drinks," and "Snacks." Menu data is stored and fetched from Firestore.

5.3.3 Product Cards and Cart Functionality

Items are displayed in horizontally scrollable product cards with options to add them to the cart or Wishlist. The cart widget overlays all screens and updates in real-time, ensuring users can manage their order from anywhere.

5.3.4 Admin Panel

Admins can view a list of restaurants, add or remove menu items, approve restaurant submissions, and monitor app performance. Firebase Firestore allows real-time updates to the system without requiring downtime.

5.3.5 Firebase Integration

- **Firestore** handles all structured data such as restaurant info, user orders, and menu items.
- **Firebase Storage** is used to upload and manage media assets including restaurant images and food item thumbnails.
- **Firebase Authentication** provides secure user sign-in with email and password.

5.3.6 Chatbot Module (Delix)

A conversational chatbot feature helps users navigate the app, suggest items based on time of day or mood, and offer support for general inquiries.

5.3.7 Real-time Order Management

Orders placed by customers are immediately pushed to restaurant dashboards. Restaurant staff can mark orders as "Preparing," "Ready for Pickup," or "Delivered."

5.3.8 Search and Filtering

The app includes a responsive search bar and category filters to help users quickly locate their favourite dishes or explore new ones.

5.3.9 Responsive UI and UX

The interface adapts to different screen sizes, offering a fluid experience on both smartphones and tablets. SliverAppBars, scrollable tabs, and elegant product cards enhance usability.

5.3.10 Future Enhancements

Future modules include:

- Payment Gateway Integration (e.g., Razorpay or UPI)
- QR-based Table Ordering
- Real-Time Notification System
- Feedback & Review System
- Analytics Dashboard for Admins

5.4 PROCESS FLOW DIAGRAM

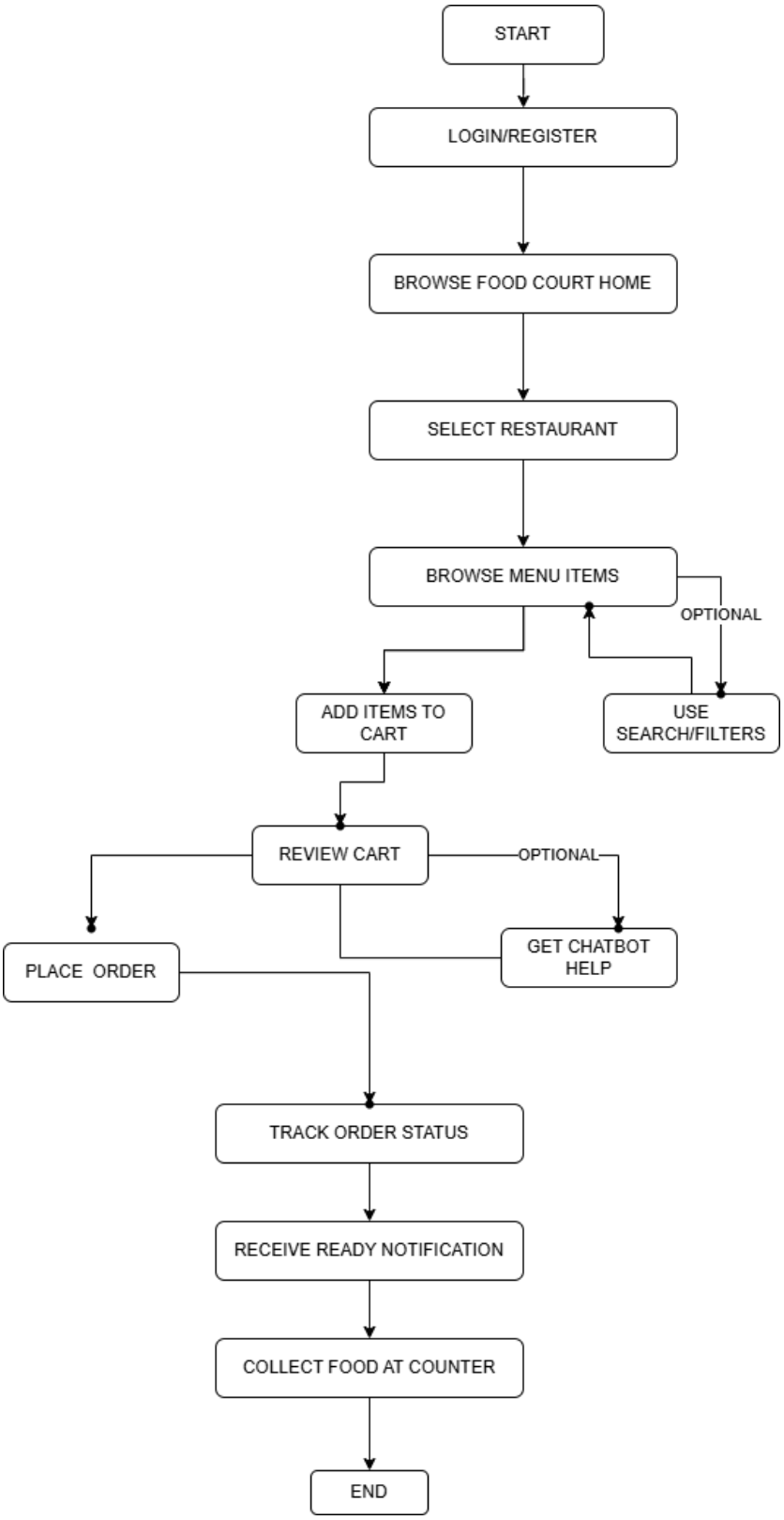

Figure 5.4 Delice – Overall Workflow

5.5 TABLE DESIGN

USER TABLE

Field Name	**Data Type**	**Constraints**
_id	ObjectId	Primary Key
userName	String	Not Null, Unique
name	String	Not Null
email	String	Not Null, Unique
password	String	Not Null (hashed)
phoneNumber	String	Not Null

Table 5.5.1 Delice - User Table

VENDOR TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
vendorId	ObjectId	Foreign Key → references users._id
name	String	Not Null
description	String	Not Null
image	String	URL format, Not Null
bannerImage	String	URL format, Optional
category	String	Not Null

Table 5.5.2 Delice - Vendor Table

PRODUCT TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
restaurantId	ObjectId	Foreign Key → references restaurants.id
name	String	Not Null
description	String	Not Null
image	String	URL format, Not Null
price	Number	Not Null, must be > 0
category	String	Not Null
foodType	String	Enum: ['veg', 'non-veg']

Table 5.5.3Delice - Product Table

CART TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
userId	ObjectId	Foreign Key → references users._id
items	Array	Contains objects with productId, quantity, price
totalAmount	Number	Calculated field
createdAt	Date	Default: Current timestamp
updatedAt	Date	Default: Current timestamp

Table 5.5.4 Delice - Cart Table

ORDER TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
userId	ObjectId	Foreign Key → references users.id
restaurantId	ObjectId	Foreign Key → references restaurants.id
items	Array	Contains objects with productId, name, quantity, price
totalAmount	Number	Not Null
orderStatus	String	Enum: ['placed', 'preparing', 'ready', 'delivered', 'cancelled']
paymentMethod	String	Enum: ['cash', 'upi', 'card']
paymentStatus	String	Enum: ['pending', 'completed', 'failed']

Table 5.5.5 Delice - Order Table

CATEGORY TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
name	String	Not Null, Unique
description	String	Optional
image	String	URL format, Optional
createdAt	Date	Default: Current timestamp

Table 5.5.6 Delice - Category Table

NOTIFICATION TABLE

Field Name	Data Type	Constraints
_id	ObjectId	Primary Key
userId	ObjectId	Foreign Key → references users.id
title	String	Not Null
message	String	Not Null
type	String	Enum: ['order', 'promo', 'system']
isRead	Boolean	Default: false
createdAt	Date	Default: Current timestamp
relatedOrderId	ObjectId	Foreign Key → references orders.id, Optional

Table 5.5.7 Delice - Notification Table

CHAPTER 6

SYSTEM TESTING

As part of the system testing process for the **Integrated Food Court Management Application**, a structured testing approach was adopted to evaluate the application's functionality, reliability, and performance. The application underwent various testing phases including **unit testing**, **integration testing**, and **manual test case validation** to ensure it meets the intended requirements for users such as customers, restaurant owners, and administrators.

6.1 UNIT TESTING

Unit testing plays a critical role in verifying individual components and functions of the application. In the Integrated Food Court Management System, unit testing was performed on:

- **Authentication Logic**: Ensuring Firebase Authentication correctly validates users and assigns the right role (Customer, Admin, or Restaurant Owner).
- **Product Display & Cart Operations**: Testing the logic that handles the addition and removal of food items from the cart.
- **Order Submission**: Verifying the correct format and structure of orders sent to Firestore.
- **Form Validations**: Validating input fields such as username, phone number, and food details to prevent invalid data entry.
- **Navigation and State Management**: Ensuring GetX state updates reflect instantly on UI components like cart count or order status.

Each module was independently tested to verify output, catch logic errors early, and maintain code quality throughout the development lifecycle.

6.2 INTEGRATION TESTING

Integration testing ensured the smooth interaction between various modules and Firebase services such as:

- **Firestore Integration**: Ensuring real-time reading and writing of user data, menu items, and orders.
- **Firebase Storage**: Validating the image upload mechanism for restaurant thumbnails and food images, and linking those images to Firestore documents.
- **Firebase Authentication & Role Routing**: Confirming that user roles are accurately identified and redirected to the correct dashboards.
- **Chatbot Modal Navigation**: Ensuring modal interactions do not affect the navigation flow of other features in the app.

The tests simulated real-world usage including ordering food, updating restaurant menus, and accessing the admin dashboard, validating seamless coordination between frontend and backend services.

6.3 TEST CASES

TEST CASE ID	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	RESULT
T1	If user enters incorrect login credentials	Displays: “Please enter valid email/password”	Displays: “Please enter valid email/password”	Success
T2	If user adds a food item to the cart	Displays: “Item added to cart” and updates cart overlay	Displays: “Item added to cart” and updates cart overlay	Success
T3	If restaurant receives a confirmed order from a customer	Restaurant dashboard shows new order with real-time status update	Restaurant dashboard shows new order with real-time status update	Success
T4	If restaurant owner marks an order as delivered	Order status updates to “Delivered” in customer and restaurant dashboards	Order status updates to “Delivered” in customer and restaurant dashboards	Success
T5	If admin views restaurant details	Displays all restaurant records with edit/remove options	Displays all restaurant records with edit/remove options	Success
T6	If a user opens the chatbot (Delix) from the navbar	Chatbot modal opens without disrupting current screen state	Chatbot modal opens without disrupting current screen state	Success
T7	If an image is uploaded for a food item	Image is uploaded to Firebase Storage and URL is saved to Firestore	Image is uploaded to Firebase Storage and URL is saved to Firestore	Success

Table 6.3.1 – Delice - Testcases

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 FOR CUSTOMERS (USERS):

- **Download the App:**
  Customers begin by downloading the **Integrated Food Court Management** application from the Google Play Store (Android). The app is designed using Flutter to support both Android and iOS platforms (future scope).
- **Installation:**
  After download, customers follow standard installation prompts to set up the app on their smartphones.
- **User Registration:**
  Customers register by entering essential information such as their name, email address, phone number, and password. The app supports secure sign-up via Firebase Authentication.
- **Login and Authentication:**
  Users authenticate using their registered credentials. Firebase ensures secure session handling and real-time login validation.
- **Explore Restaurants:**
  Upon successful login, customers are presented with a list of registered restaurants. They can browse based on cuisine, popularity, categories (e.g., South Indian, Burgers, Beverages), or restaurant name.
- **View Restaurant Menu:**
  Users can access detailed menus for each restaurant, including product names, descriptions, images, and prices.
- **Place Orders:**
  Customers can add items to their cart, view order summaries, and proceed to place orders. Order data is stored in **Firebase Firestore** for real-time tracking.
- **Track Order Status:**
  Customers can view the current status of their order (e.g., Ordered, Preparing, Ready, Delivered) updated in real time by the restaurant/vendor.

- **Favorites and Wishlist:**
  Users can like or favorite items for future reference, making it easy to reorder preferred dishes.
- **Feedback and Ratings:**
  After order completion, customers can rate their experience and submit feedback to help restaurants improve service quality.

7.2 FOR RESTAURANT OWNERS:

- **Download the App:**
  Restaurant vendors download the **Integrated Food Court Management** application from the Play Store.
- **Installation and Setup:**
  Vendors install the app and proceed with onboarding by creating an account as a restaurant owner.
- **Restaurant Registration:**
  Shop owners provide details like restaurant name, location, food categories offered, and contact info. Images, logos, and banners can be uploaded using **Firebase Storage**.
- **Menu Upload and Management:**
  Owners add menu items under specific categories, including food images, pricing, and descriptions. Each item can be edited or updated from the vendor dashboard.
- **Order Management:**
  Real-time order tracking is available, enabling restaurants to accept, update, and mark orders as delivered via their interface.
- **Restaurant Profile Maintenance:**
  Owners can maintain their store profile with up-to-date information, promotional offers, or updated menus. Regular updates ensure continued engagement and relevance.
- **User Interaction (Future Scope):**
  A future enhancement includes live chat or customer query resolution modules to enable direct communication with customers regarding orders or issues.

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

8.1 CONCLUSION

The **Integrated Food Court Management Application** successfully bridges the gap between food vendors, customers, and administrators through a centralized, digital ordering platform. Built using **Flutter** for cross-platform compatibility and **Firebase** for backend services, the application ensures a seamless user experience across devices while maintaining real-time synchronization of data.

The system simplifies food ordering, vendor menu management, and order tracking by incorporating essential features like restaurant browsing, categorized product listings, cart management, and dynamic user dashboards. Through role-based access, it provides tailored interfaces for administrators, restaurant owners, and end-users, ensuring streamlined workflows and ease of access.

The integration of Firebase Firestore, Storage, and Authentication ensures data security, scalability, and efficient media management. Additionally, the use of modern UI/UX design principles enhances the app's usability and appeal, making the overall system a highly functional and user-friendly solution for managing multi-restaurant food courts.

8.2 FUTURE ENHANCEMENTS

- **iOS Compatibility**: Expanding the current Android-first application to support iOS will enable wider adoption and provide seamless cross-platform functionality for all users.
- **Online Payment Integration**: Integrating payment gateways like **Razorpay**, **UPI**, or **Stripe** will allow customers to pay for orders online, making the ordering process smoother and contactless.
- **Order Tracking System**: Implementing real-time order tracking with live status updates (e.g., "Preparing," "Ready for Pickup," "Delivered") will enhance user engagement and transparency.

- **Restaurant Analytics Dashboard**: Introducing analytics for restaurant owners (e.g., order volume, sales reports, and popular items) can help them make informed business decisions.
- **Push Notifications**: Adding real-time push notifications for order status updates, promotional offers, and feedback requests will keep users actively engaged.
- **Admin Panel (Web Interface)**: Developing a dedicated web-based admin panel for overseeing restaurants, users, and orders will improve system maintainability and administrative control.
- **AI-Powered Recommendations**: Future versions may include ML-driven food suggestions based on user behavior and purchase history to increase personalization.
- **Multilingual Support**: Supporting multiple regional languages will make the application accessible to a broader demographic, catering to diverse users across regions.

CHAPTER 9

APPENDICES

9.1 SOURCE CODE

Main.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter_native_splash/flutter_native_splash.dart';
import 'package:get_storage/get_storage.dart';
import 'data/repositories/authentication_repository/authentication_repository.dart';
import 'features/shop/controllers/cart_controller.dart';
import 'firebase_options.dart';
import 'app.dart';
import 'package:get/get.dart';

Future<void> main() async {
  ///widgets binding
  final WidgetsBinding widgetsBinding =
      WidgetsFlutterBinding.ensureInitialized();

  ///getx Local Storage
  await GetStorage.init();

  ///Await splash until other items load
  FlutterNativeSplash.preserve(widgetsBinding: widgetsBinding);

  ///initialize firebase
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform)
      .then(
    (FirebaseApp value) => Get.put(AuthenticationRepository()),
  );
  Get.put(CartController());
```

```
  runApp(const App());
}
```

Firebase Services Options

```
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
```

```
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.windows:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for windows - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.linux:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for linux - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyA7YXq52HaDhUQsN3FSvZ7YYoh4cyTvgj4',
  appId: '1:548543607005:web:37c4e776a36de3d199ab34',
  messagingSenderId: '548543607005',
  projectId: 'diner-cbac3',
  authDomain: 'diner-cbac3.firebaseapp.com',
  databaseURL: 'https://diner-cbac3-default-rtdb.firebaseio.com',
  storageBucket: 'diner-cbac3.firebasestorage.app',
  measurementId: 'G-Y1E26ZFSFV',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyAPV-w7w1pnZcjfaAFMCstMjjCP05SkEso',
  appId: '1:548543607005:android:2a70d396eb7666ca99ab34',
  messagingSenderId: '548543607005',
```

```
    projectId: 'diner-cbac3',
    databaseURL: 'https://diner-cbac3-default-rtdb.firebaseio.com',
    storageBucket: 'diner-cbac3.firebasestorage.app',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyDMBsizq0x5O2NXlzrsCYmZ4dYJqSmOfBQ',
    appId: '1:548543607005:ios:4d90dcd81b37e8d499ab34',
    messagingSenderId: '548543607005',
    projectId: 'diner-cbac3',
    databaseURL: 'https://diner-cbac3-default-rtdb.firebaseio.com',
    storageBucket: 'diner-cbac3.firebasestorage.app',
    androidClientId:
        '548543607005-o7g69369g0m4t5er51leet1si7elchmc.apps.googleusercontent.com',
    iosClientId:
        '548543607005-mbrb3ba49cgsuldl8kq698fg7c9vu16k.apps.googleusercontent.com',
    iosBundleId: 'com.example.projectImprovised',
  );
}
```

PUBSPEC.yaml

```
name: project_improvised
description: "A Flutter project of my major but improvised with some new components"
publish_to: 'none'
version: 1.0.0+1
environment:
  sdk: ^3.6.0
#- - - - - - - - - - - Packages ----------#
dependencies:
  flutter:
    sdk: flutter
  #Utility Packages
  http: ^1.1.0
```

intl: ^0.20.2
logger: ^2.0.1
image_picker: ^1.0.4
url_launcher: ^6.1.12
flutter_native_splash: ^2.4.3
video_player: ^2.9.2
connectivity_plus: ^6.1.3
smooth_page_indicator: ^1.2.1
carousel_slider: ^5.0.0
shimmer: ^3.0.0

#Icons
iconsax: ^0.0.8
cupertino_icons: ^1.0.8

#State management
get: ^4.6.5
get_storage: ^2.1.1
#Product Specific
readmore: ^3.0.0

#Firebase
firebase_core: ^2.23.1
firebase_auth: ^4.14.4
cloud_firestore: ^4.13.1
firebase_storage: ^11.5.4
lottie: ^3.3.1
google_sign_in: ^6.1.6
cached_network_image: ^3.4.0

#- - - - - - - - ./ Packages END----------#

```
dev_dependencies:
  flutter_test:
    sdk: flutter
flutter_lints: ^5.0.0
flutter:
  uses-material-design: true

  #- - - -Local Assets------#
  assets:
    - assets/images/
    - assets/videos/
    - assets/icons/
    - assets/animations/


  #- - - -Local Fonts------#
  fonts:
    - family: Fira Sans
      fonts:
        - asset: assets/fonts/FiraSans-Regular.ttf
        - asset: assets/fonts/FiraSans-Light.ttf
        - asset: assets/fonts/FiraSans-Medium.ttf
          weight: 500
        - asset: assets/fonts/FiraSans-SemiBold.ttf
          weight: 600
        - asset: assets/fonts/FiraSans-Bold.ttf
          weight: 700
        - asset: assets/fonts/FiraSans-Bold.ttf
          weight: 800
```

LOGIN VIEW

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:project_improvised/common/styles/spacing_styles.dart';
import
'package:project_improvised/features/authentication/screens/login/widgets/login_form.dart';
import
'package:project_improvised/features/authentication/screens/login/widgets/login_header.dart';
import 'package:project_improvised/utils/constants/sizes.dart';
import 'package:project_improvised/utils/constants/text_strings.dart';

import '../../../../common/widgets/form_divider.dart';
import '../../../../common/widgets/social_buttons.dart';

class LoginView extends StatelessWidget {
  const LoginView({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Padding(
          padding: DSpacingStyle.paddingWithAppBarHeight,
          child: Column(
            children: [
              ///Logo, Title & Subtitle
              LoginHeader(),

              ///Form
              LoginForm(),

              ///Divider
              FormDivider(
```

```
              dividerText: DTexts.orSignIn.capitalize!,
            ),
            const SizedBox(
              height: DSizes.spaceBtwSections,
            ),

            /// Footer
            SocialButtons(),
          ],
        ),
      ),
    ),
  );
 }
}
```

SIGN UP VIEW

```
// ignore_for_file: file_names

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:project_improvised/common/widgets/form_divider.dart';
import 'package:project_improvised/common/widgets/social_buttons.dart';
import
'package:project_improvised/features/authentication/controllers/signup_controller.dart';
import
'package:project_improvised/features/authentication/screens/signUp/widgets/singUp_form.da
rt';
import 'package:project_improvised/utils/constants/sizes.dart';

import '../../../../utils/constants/text_strings.dart';

class SignupView extends StatelessWidget {
```

```
const SignupView({super.key});

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(),
    body: SingleChildScrollView(
      child: Padding(
        padding: EdgeInsets.all(DSizes.defaultSpace),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            ///Title
            Text(
              "Create your Account",
              style: Theme.of(context).textTheme.headlineMedium,
            ),
            const SizedBox(
              height: DSizes.spaceBtwSections,
            ),

            ///Form
            SignUp_form(),
            const SizedBox(
              height: DSizes.spaceBtwSections,
            ),

            ///Divider
            FormDivider(dividerText: DTexts.orSignUp.capitalize!),
            const SizedBox(
              height: DSizes.spaceBtwSections,
            ),

            ///Social Buttons
```

```
            SocialButtons(),
          ],
        ),
      ),
    ),
  );
 }
}
```

PROFILE VIEW

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:get/get_core/src/get_main.dart';
import 'package:project_improvised/common/widgets/appbar/appBar.dart';
import 'package:project_improvised/common/widgets/images/circular_image_container.dart';
import 'package:project_improvised/common/widgets/texts/sectionHeading.dart';
import 'package:project_improvised/features/personalization/controllers/user_controller.dart';
import
'package:project_improvised/features/personalization/screens/profile/widgets/profile_menu.d
art';
import 'package:project_improvised/utils/constants/colors.dart';
import 'package:project_improvised/utils/constants/image_strings.dart';
import 'package:project_improvised/utils/constants/sizes.dart';
import 'package:project_improvised/utils/devices/devices_utilities.dart';

import 'change_name.dart';

class ProfileView extends StatelessWidget {
  const ProfileView({super.key});

  @override
  Widget build(BuildContext context) {
    final controller = UserController.instance;
```

```
return Scaffold(
  appBar: DAppBar(
    height: DDeviceUtils.getAppBarHeight(),
    showBackArrow: true,
    title: Text(
      "Profile",
      style: Theme.of(context).textTheme.headlineMedium,
    ),
  ),

  ///Body
  body: SingleChildScrollView(
    child: Padding(
      padding: EdgeInsets.all(DSizes.defaultSpace),
      child: Column(
        children: [
          ///Profile Pic
          SizedBox(
            width: double.infinity,
            child: Column(
            children: [
                Image(
                  image: AssetImage(DImages.appLogo),
                  width: 140,
                  height: 140,
                  color: DColors.primary,
                ),
              ],
            ),
          ),

          ///Details
          SizedBox(
            height: DSizes.spaceBtwItems,
```

```
),
SizedBox(
  height: DSizes.spaceBtwItems,
),
DSectionHeading(
  title: "Profile Information",
  showActionButton: false,
),
SizedBox(
  height: DSizes.spaceBtwItems,
),
DProfileMenu(
  title: "Name",
  value: controller.user.value.fullName,
  onPressed: () => Get.to(
    () => ChangeName(),
  ),
),
DProfileMenu(
    onPressed: () {},
    title: "Username",
    value: controller.user.value.username),

SizedBox(
  height: DSizes.spaceBtwItems,
),
Divider(),
SizedBox(
  height: DSizes.spaceBtwItems,
),

///Heading Personal info
DSectionHeading(
  title: "Personal Information",
```

```
  showActionButton: false,
),
SizedBox(
  height: DSizes.spaceBtwItems,
),

DProfileMenu(
  onPressed: () {},
  title: "User ID",
  value: controller.user.value.id,
  icon: Icons.copy,
),
DProfileMenu(
  onPressed: () {},
  title: "E-mail",
  value: controller.user.value.email,
  icon: Icons.email,
),
DProfileMenu(
  onPressed: () {},
  title: "Phone Number",
  value: controller.user.value.phoneNumber,
  icon: Icons.phone,
),
DProfileMenu(
    onPressed: () {},
    title: "Gender",
    value: "user's name",
    icon: Icons.person_2_rounded),
DProfileMenu(
  onPressed: () {},
  title: "Date of Birth",
  value: "user's name",
  icon: Icons.calendar_month,
```

```
            ),
            Divider(),
            SizedBox(
              height: DSizes.spaceBtwItems * 2,
            ),

            Center(
              child: SizedBox(
                width: double.infinity,
                child: OutlinedButton(
                  onPressed: () => controller.deleteAccountWarningPopup(),
                  style: OutlinedButton.styleFrom(
                    side: BorderSide(
                        color: Colors.red), // Set border color to red
                  ),
                  child: Text(
                    "Close Account",
                    style: Theme.of(context)
                        .textTheme
                        .labelLarge!
                        .apply(color: Colors.red),
                  ),
                ),
              ),
            )
          ],
        ),
      ),
    ),
  );
 }
}
```

HOME VIEW

```
// ignore_for_file: file_names, avoid_print, unnecessary_import

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:get/get_core/src/get_main.dart';
import 'package:project_improvised/common/widgets/icons/icon_categories.dart';
import
'package:project_improvised/common/widgets/products/product_card_horizontal.dart';
import 'package:project_improvised/common/widgets/products/product_card_vertical.dart';
import 'package:project_improvised/common/widgets/shimmer/vertical_shimmer.dart';
import 'package:project_improvised/common/widgets/texts/sectionHeading.dart';
import 'package:project_improvised/features/shop/screens/all_products/all_products.dart';
import 'package:project_improvised/features/shop/screens/home/widgets/banner_slider.dart';
import 'package:project_improvised/features/shop/screens/home/widgets/home_appBar.dart';
import 'package:project_improvised/features/shop/screens/restaurants/all_restaurants.dart';
import
'package:project_improvised/features/shop/screens/restaurants/widgets/restaurant_screen.dart'
;
import 'package:project_improvised/features/shop/screens/store/storeView.dart';
import 'package:project_improvised/utils/constants/image_strings.dart';
import 'package:project_improvised/utils/constants/sizes.dart';
import '../../../../common/widgets/custom_shapes/containers/header_container.dart';
import '../../../../common/widgets/custom_shapes/containers/searchbar.dart';
import '../../../../common/widgets/images/imageVerticalText.dart';
import '../../../../common/widgets/layouts/grid_layout.dart';
import '../../../../utils/constants/colors.dart';
import '../cart/widgets/cart_overlay.dart';
import '../sub-categories/sub_categories.dart';

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          child: Column(
          children: [
             ///header
             HeaderContainer(
               color: DColors.primary,
               height: 365,
               child: Column(
                 crossAxisAlignment: CrossAxisAlignment.start,
                 children: [
                   SizedBox(
                     height: 10,
                   ),

                   /// ---App bar---
                   HomeAppBar(),
                   SizedBox(
                     height: DSizes.spaceBtwInputFields,
                   ),

                   /// --- Searchbar ---
                   DSearchBar(
                     text: "Search",
                     onTap: () => Get.to(() => StoreView()),
                     padding: EdgeInsets.symmetric(horizontal: 10),
                   ),

                   /// --- Categories changed into an image according to the festival ---
                   InkWell(
```

```
        onTap: () {
          print("must redirect to another page");
        },
        child: Padding(
          padding:
              const EdgeInsets.only(top: 0, left: 4, right: 4),
          child: Image(
            ///width 400 height 190
            image: AssetImage(
                "assets/images/festivalBanner_image.png"),
            fit: BoxFit.contain,
            alignment: Alignment
                .center, // Centers image inside container
          ),
        ),
      ),
    ],
  ),
),

///Body
Padding(
  padding: EdgeInsets.only(left: 10, right: 10, top: 0),
  child: Column(
    children: [
      ///promo slider
      SizedBox(
      height: 15,
      ),
      DBannerSlider(),
      SizedBox(
        height: DSizes.spaceBtwSections,
      ),
```

```
      ///heading
      DSectionHeading(
        title: 'Popular Restaurants',
        onPressed: () => Get.to(
          () => AllRestaurants(),
        ),
      ),
      SizedBox(
        height: DSizes.spaceBtwItems,
      ),

      /// popular Products
      DCategories(),
    ],
  ),
),
SizedBox(
  height: DSizes.spaceBtwSections,
),
Container(
  color: Color.fromARGB(94, 230, 229, 229),
  width: double.infinity,
  child: Padding(
    padding: const EdgeInsets.only(
      left: 15,
    ),
    child: Column(
      mainAxisAlignment:  MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        /// "Live it up!" Text
        Text(
          "Live\nit up!",
          textAlign: TextAlign.start,
```

```
    style: TextStyle(
      fontSize: 78, // Adjust as needed
      fontWeight: FontWeight.bold,
      color: DColors.darkGrey, // Dark grey color
      height: 1.1, // Adjust line spacing
    ),
  ),

  const SizedBox(
      height:
          DSizes.defaultSpace), // Spacing between texts

  /// "Crafted with ❤ in Bengaluru, India"
  RichText(
    textAlign: TextAlign.center,
    text: TextSpan(
      style: TextStyle(
        fontSize: 16,
        fontWeight: FontWeight.w500,
        color: Colors.grey[600], // Medium grey color
      ),
      children: [
        const TextSpan(text: "Crafted with "),
        WidgetSpan(
          child: Icon(Icons.favorite,
              color: Colors.red, size: 18), // Heart Icon
        ),
        const TextSpan(text: " in Tirchy, Tamil Nadu"),
      ],
    ),
  ),
  const SizedBox(height: 100),
],
```

```
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
 }
}
```

STORE VIEW

```
// ignore_for_file: file_names

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:project_improvised/common/widgets/category_tab.dart';
import
'package:project_improvised/common/widgets/custom_shapes/containers/searchbar.dart';
import 'package:project_improvised/common/widgets/texts/sectionHeading.dart';
import 'package:project_improvised/features/shop/models/brand_model.dart';
import 'package:project_improvised/features/shop/screens/all_brands/all_brands.dart';
import 'package:project_improvised/utils/constants/sizes.dart';
import '../../../../common/widgets/appbar/appBar.dart';
import '../../../../common/widgets/appbar/tabbar.dart';
import '../../../../common/widgets/brand/brand_card.dart';
import '../../../../utils/constants/colors.dart';
import '../../../../utils/devices/devices_utilities.dart';

class StoreView extends StatelessWidget {
  StoreView({super.key});
```

```
final List<Brand> brands = [
  Brand(
      name: "McDonald's",
      imagePath: "assets/images/McD_logo.png",
      productCount: 48),
  Brand(
      name: "Hotel Kannappa",
      imagePath: "assets/images/kannnappalogo.png",
      productCount: 37),
  Brand(
      name: "B.G.Naidu",
      imagePath: "assets/images/bgnaidu.png",
      productCount: 30),
  Brand(
      name: "KMS Hakkim",
      imagePath: "assets/images/kmslogo.png",
      productCount: 23)
];
@override
Widget build(BuildContext context) {
  return DefaultTabController(
    length: 8,
    child: Scaffold(
      appBar: DAppBar(
        title: Text("Store"),
        height: DDeviceUtils.getAppBarHeight(),
      ),
      body: NestedScrollView(
        headerSliverBuilder: (_, innerBoxIsScrolled) {
          return [
            SliverAppBar(
              automaticallyImplyLeading: false,
              pinned: true,
              floating: true,
```

```
backgroundColor: DColors.white,
expandedHeight: 440,
flexibleSpace: Padding(
  padding: EdgeInsets.all(DSizes.md),
  child: ListView(
    shrinkWrap: true,
    physics: NeverScrollableScrollPhysics(),
    children: [
      ///Search Bar
      DSearchBar(
      text: "Search",
        showBackground: false,
        showBorder: true,
        padding: EdgeInsets.zero,
      ),
      SizedBox(
        height: DSizes.spaceBtwSections / 2,
      ),

      /// Featured Brands
      DSectionHeading(
        title: "Top Picks For You",
        onPressed: (() => Get.to(AllBrandsView())),
      ),
      SizedBox(
        height: DSizes.spaceBtwItems / 1.75,
      ),

      ///brand Grids
      Container(
        width: double.infinity,
        child: ListView.separated(
        itemCount: brands.length,
        shrinkWrap: true,
```

```
        padding: const EdgeInsets.all(DSizes.defaultSpace),
        separatorBuilder: (_, __) =>
            const SizedBox(height: DSizes.spaceBtwItems),
        itemBuilder: (context, index) {
          final brand = brands[index];
          return DBrandCard(
            brand: brand,
            showBorder: true,
          );
        },
      ),
    ),
    SizedBox(
      height: DSizes.spaceBtwItems,
    ),
  ],
 ),
),

/// Tabs
bottom: DTabBar(tabs: [
 Tab(
  child: Text("All"),
 ),
 Tab(
  child: Text("South Indian"),
 ),
 Tab(
  child: Text("North Indian"),
 ),
 Tab(
  child: Text("Chinese"),
 ),
 Tab(
```

```
                child: Text("Italian"),
              ),
              Tab(
                child: Text("American"),
              ),
              Tab(
                child: Text("Arabian"),
              ),
              Tab(
                child: Text("Desserts and Beverages"),
              )
            ]),
          ),
        ];
      },
      body: TabBarView(children: [DCategoryTab()]),
    ),
  ),
 );
 }
}
```

9.2 SCREENSHOTS

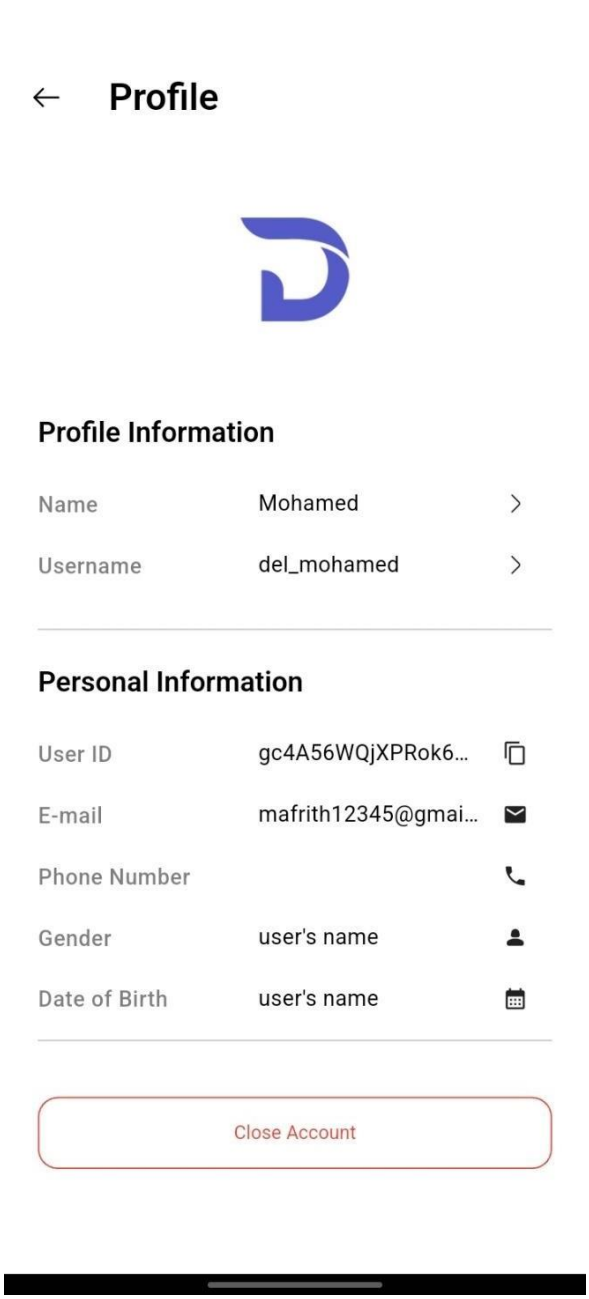

Figure 9.2.1 Edit Profile View

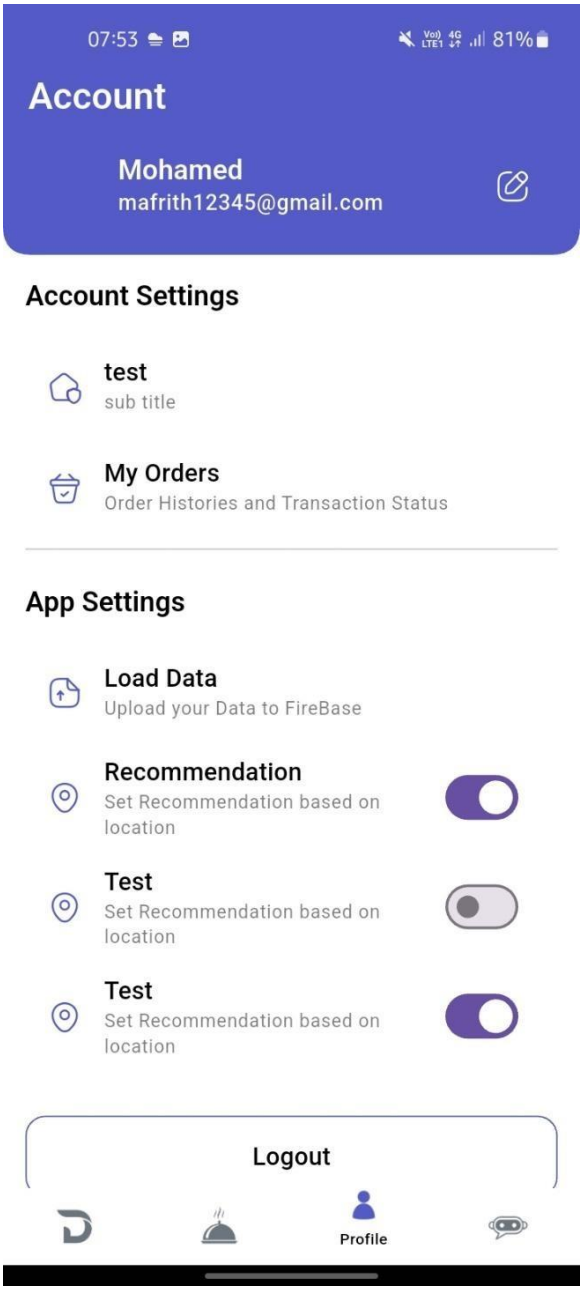

Figure 9.2.2 Profile View

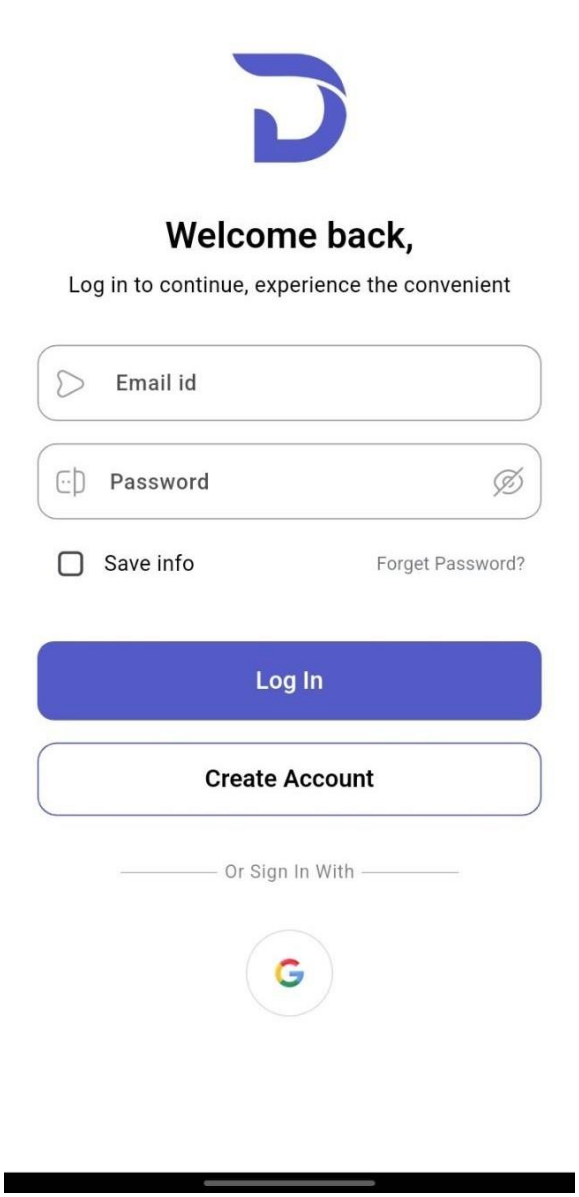

Figure 9.2.3 Login View

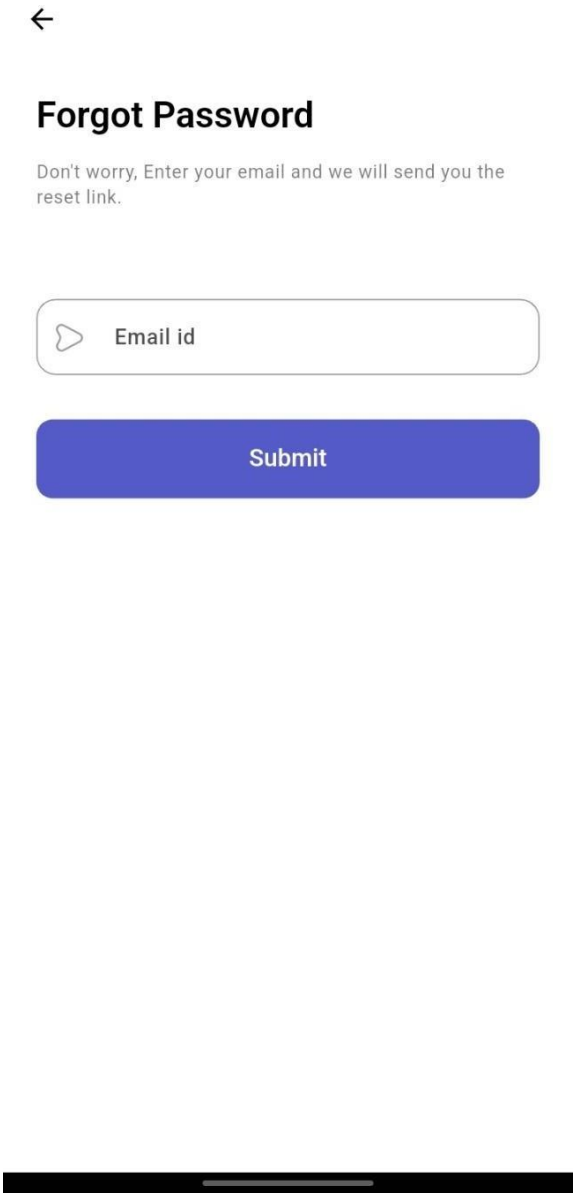

Figure 9.2.4 Reset Password View

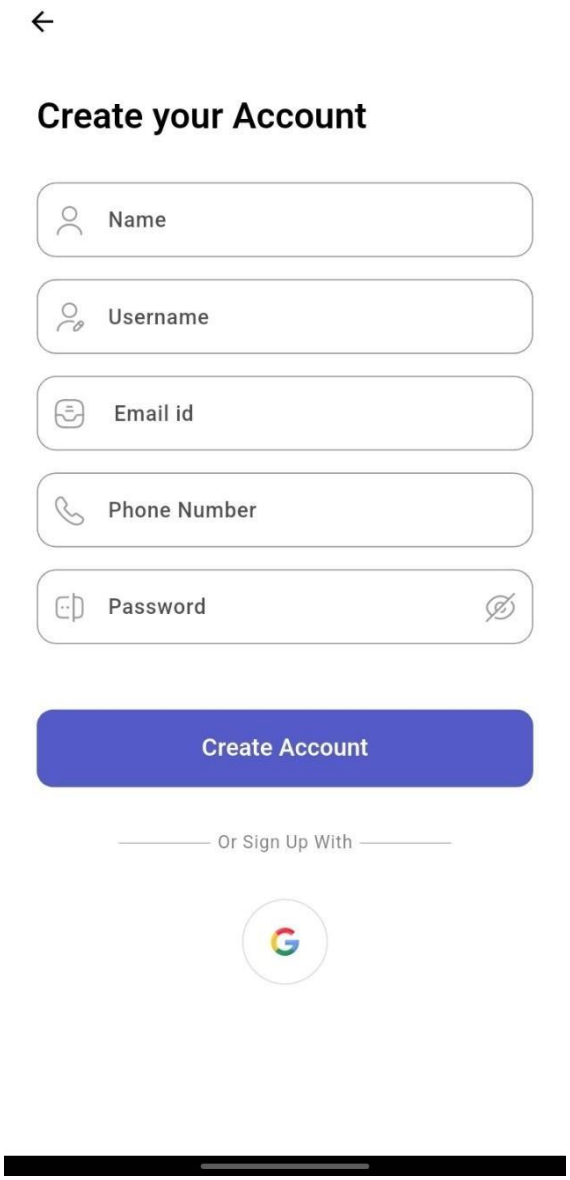

Figure 9.2.5 Signup View

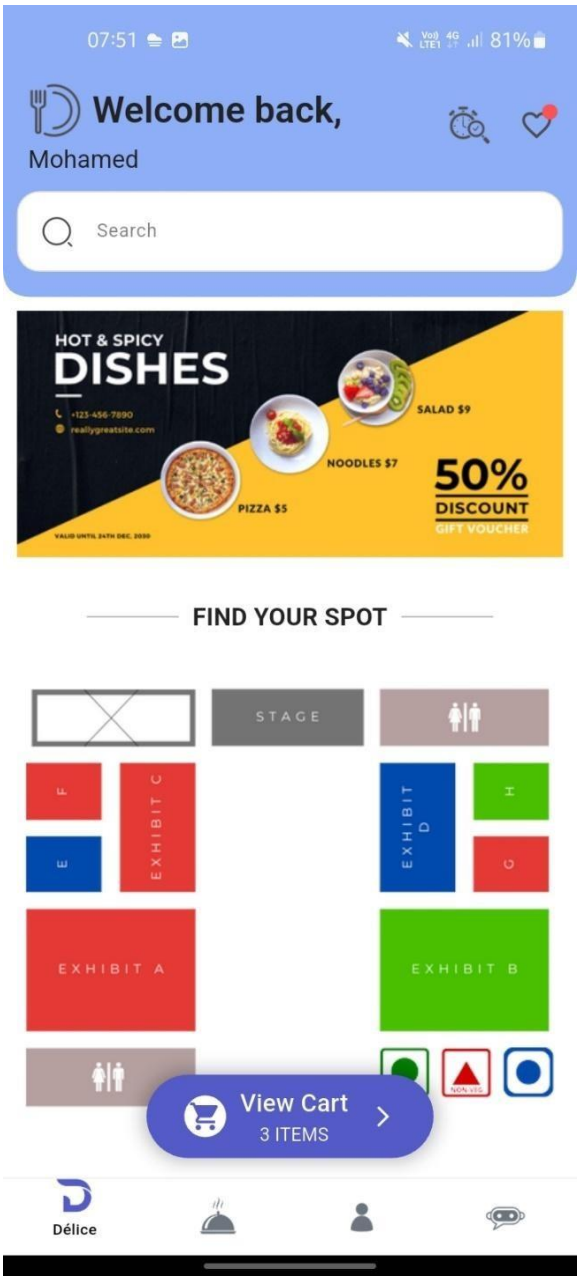

Figure 9.2.6 Home View

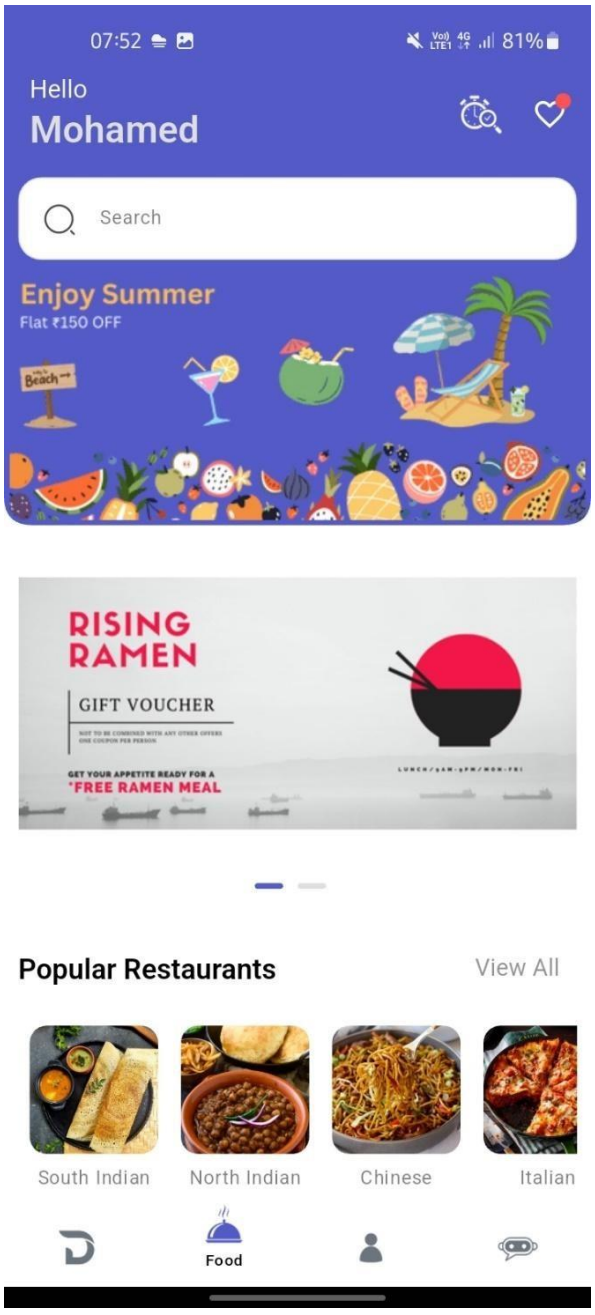

Figure 9.2.7 Menu View

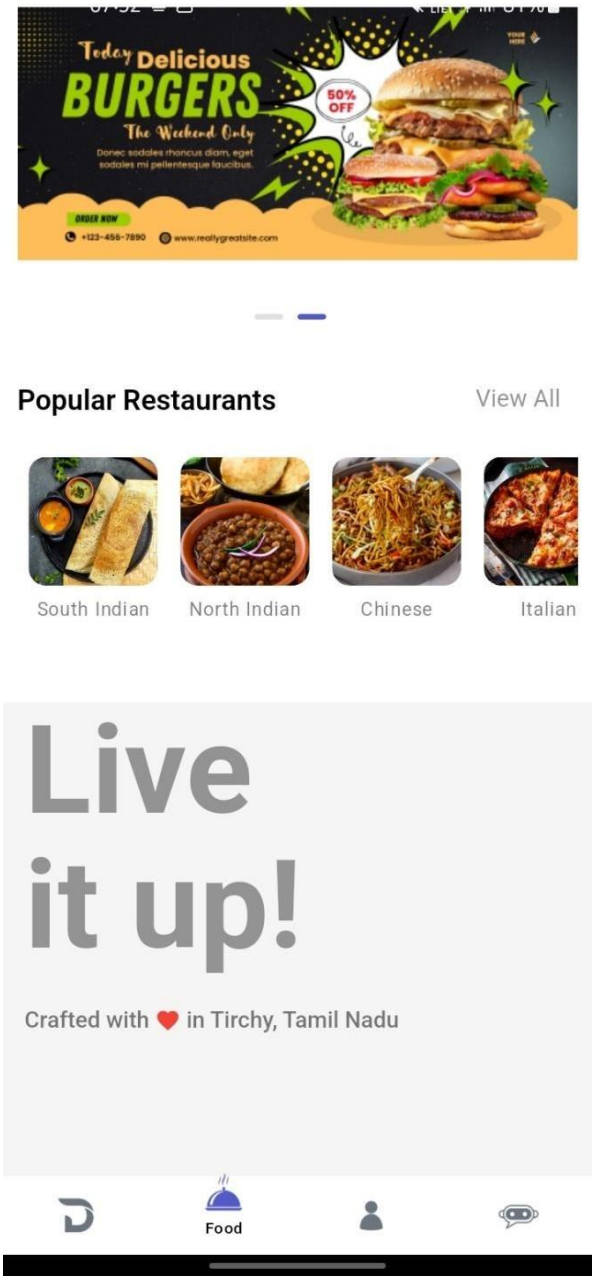

Figure 9.2.8 Sub-Category View

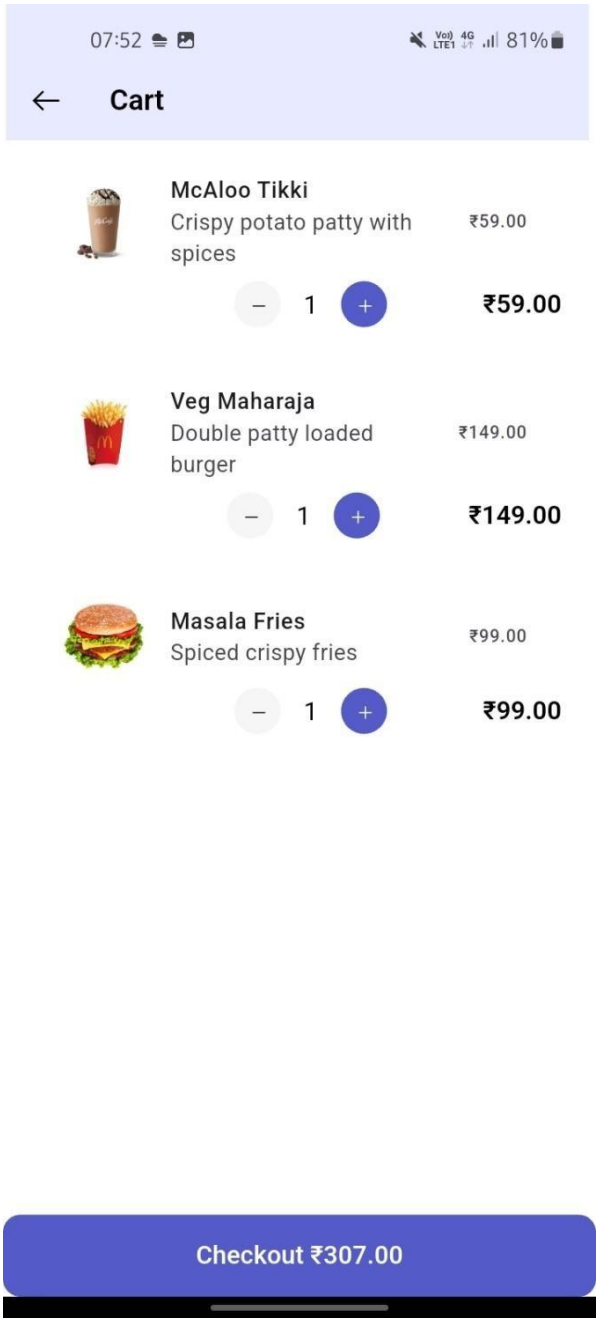

Figure 9.2.9 Cart View

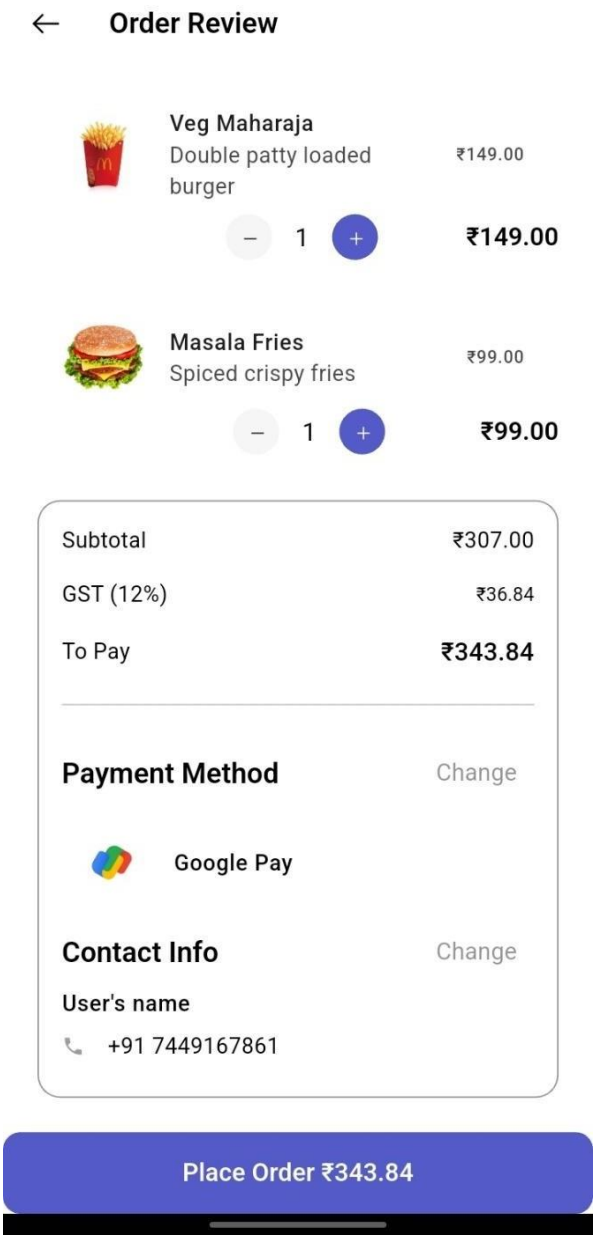

Figure 9.2.10 Checkout View

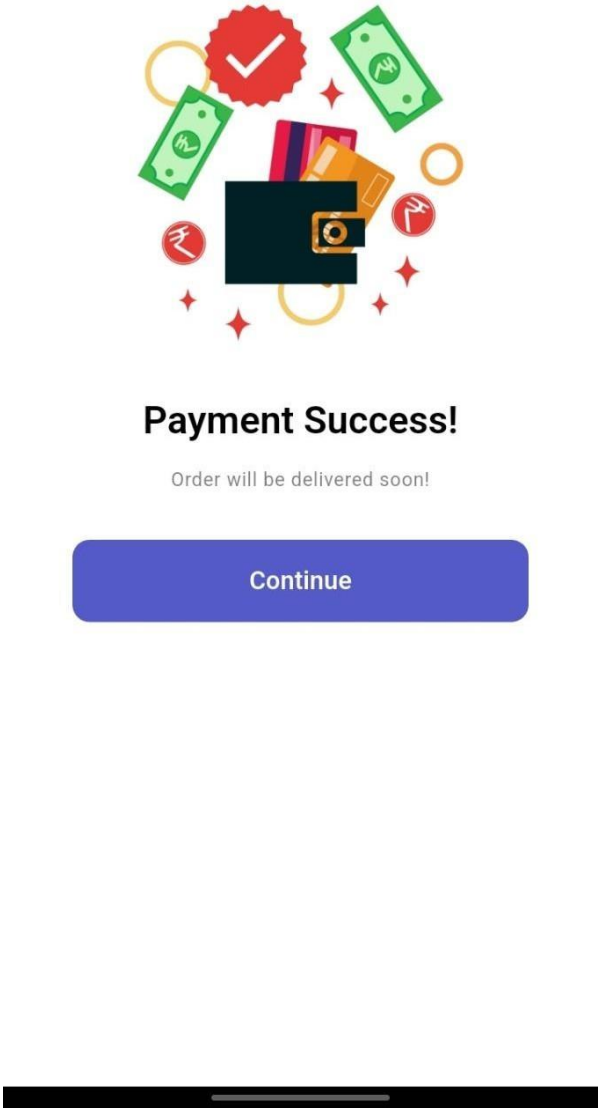

Figure 9.2.11 Payment Success View

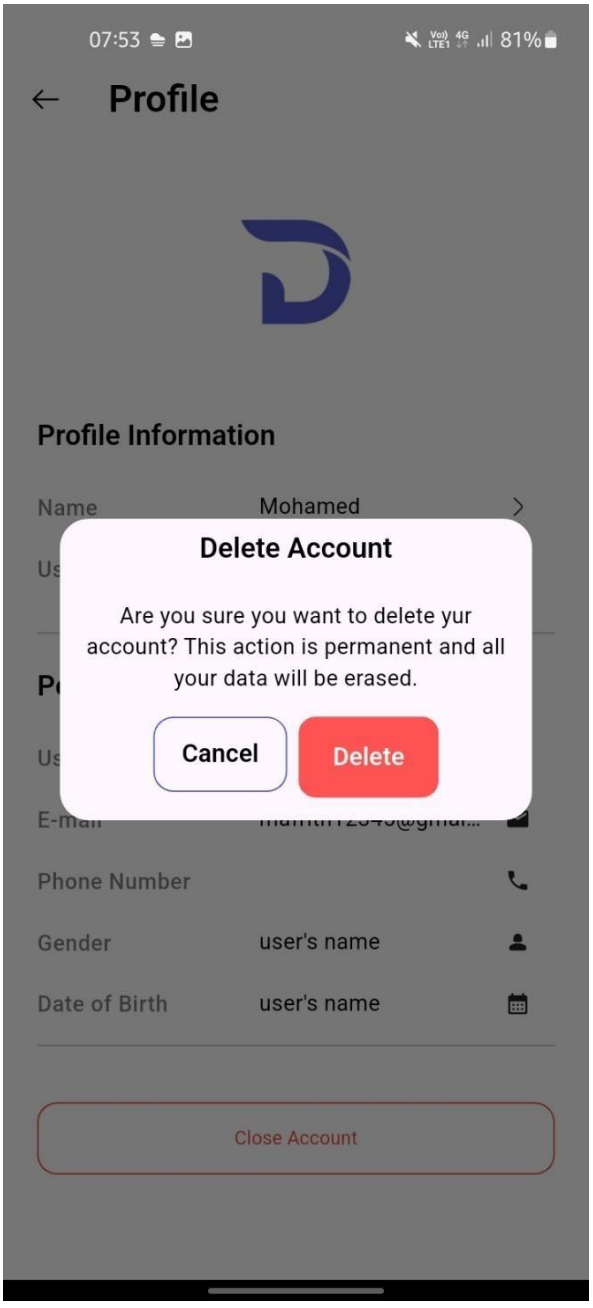

Figure 9.2.12 Delete Account View

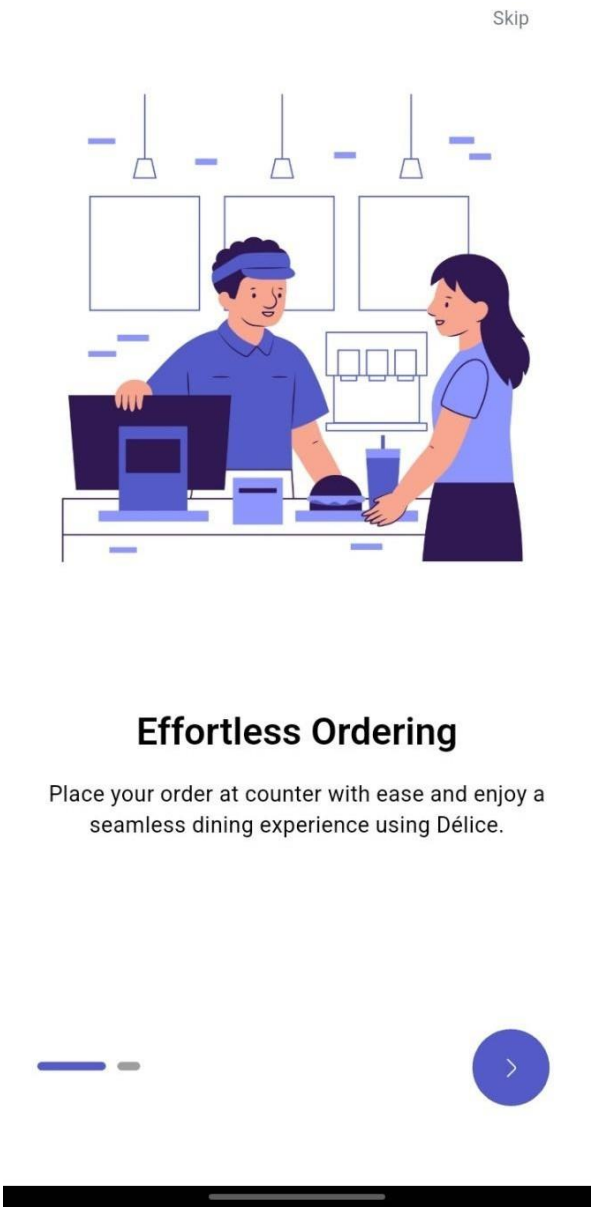

Figure 9.2.13 Onboard View-1

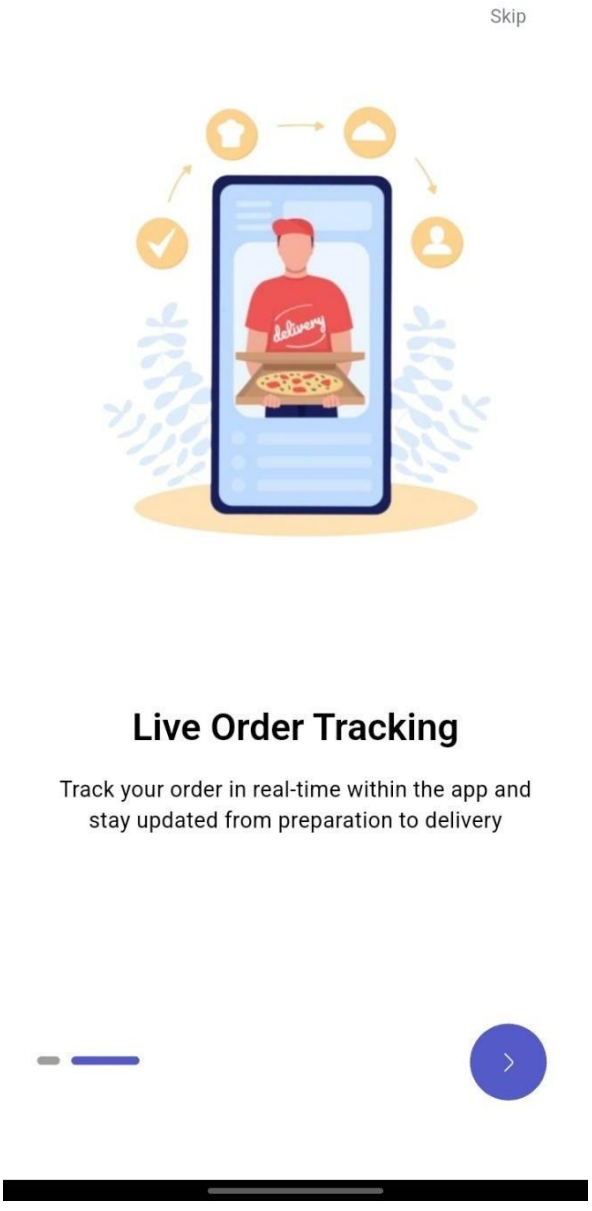

Figure 9.2.14 Onboard View-2

CHAPTER 10

REFERENCES

1. **Flutter Documentation** – Official documentation used for building cross-platform mobile UIs for both Android and iOS.
https://docs.flutter.dev

2. **Firebase Firestore Documentation** – Referenced for real-time cloud database integration to store user, restaurant, and order data.
https://firebase.google.com/docs/firestore

3. **Firebase Storage Documentation** – Used for uploading and retrieving food images, restaurant banners, and user profile pictures.
https://firebase.google.com/docs/storage

4. **Firebase Authentication** – Used for secure login, registration, and OTP verification across different user roles.
https://firebase.google.com/docs/auth

5. **GetX Documentation** – For state management, routing, and dependency injection to ensure smooth navigation and UI updates.
https://pub.dev/packages/get

6. **Dart Programming Language** – For developing business logic, model classes, and data validation in Flutter.
https://dart.dev/guides

7. **Google Fonts and Material Icons** – Integrated for enhancing visual appeal and iconography across the user interface.
https://fonts.google.com
https://fonts.google.com/icons

8. **Cloud Firestore Security Rules** – To manage read/write permissions for customers, restaurants, and admins.
https://firebase.google.com/docs/firestore/security/get-started