Creating a Food Ordering and Delivering App with a Pre-Ordering Feature

by

Sopnil Roy Niloy
20101232

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
July 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sopnil Roy Niloy

20101232

Approval

The thesis/project titled "Creating a food ordering app like with scheduled ordering feature" submitted by

1. Sopnil Roy Niloy (20101232)

of Fall 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelors on 20-06-2025.

Examining Committee:

Supervisor:
(Member)

Arif Shakil

Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Over the past ten years, the market for food delivery and ordering has expanded rapidly due to both rising customer demand and technology improvements. Notwithstanding the market revolution brought about by contemporary platforms, there are still issues with offering genuinely seamless, customized, and end-to-end ordering experiences, particularly with regard to order customization and flexible scheduling. This project seeks to fill these gaps through the creation of a state-of-the-art mobile application for food ordering and delivery that emphasizes user convenience, streamlines operational efficiency, and supports local restaurants. With a modern technology architecture that uses React Native to enable effortless cross-platform mobile functionality, in conjunction with a Python-based backend to allow for heavy processing needs, and a Large Language Model (Llama 3) to enable advanced conversational AI, the app effectively optimizes the ordering process. Key functionalities like conversational ordering, personalized recommendations, live order tracking, and a critical scheduled ordering feature are incorporated to significantly minimize processing times and enhance customer satisfaction. Apart from the benefits provided to customers, the platform aims to enable restaurants at the local level by providing a simple and effective digital platform to reach a wider customer base. This research examines the development and influence of food ordering and delivery apps, focusing specifically on market dynamics, social implications, and the critical technology and user interface features necessary for the development of highly adopted mobile apps. By thoroughly examining these factors and their integration according to a scalable, secure, and user-focused design, this research improves understanding and aims to contribute meaningfully to the changing environment of food delivery systems.

Keywords: Food ordering and delivery industry, Mobile ordering system, Convenience and Efficiency

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our co-advisor Teacher Name sir for his kind support and advice in our work. He helped us whenever we needed help.
Thirdly, Name and the whole judging panel of Conference Name. Though our paper not accepted there, all the reviews they gave helped us a lot in our later works.
And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

Chapter 1

Introduction

In the last decade, the food ordering and delivery industry has undergone significant transformations, adapting to advanced technologies, and shifting consumer tastes and behaviors. Smartphones have become integral to everyday life as the world becomes more devitalized. The human mind is so curved to taste different types of cuisine. The pace of the distinctive variety of restaurants and their food menu has increased to the next level. Now all restaurants have incorporated food ordering apps into their gadgets. These mobile applications provide an unrivaled level of comfort since they allow users to place orders from thousands of restaurants directly to their doorstep with just a few taps on their mobile devices. Urbanization, faster lifestyle, and higher demand for various types of food are generally the few factors that have influenced the growth of food delivery services. The conventional techniques of placing food orders such as making phone calls or physically going down to the food joints are being replaced by modern technological platforms as they offer efficient, easy, and convenient interfaces. Uber Eats, Door Dash, Grub Hub, Deliveroo, and Food Panda have changed the industry standard with real-time order tracking, multiple payment options, personalized recommendations, and many more. This research paper will discuss the evolution and influence of food ordering and delivery apps. It will also discuss the technologies, their market, and their social impact. This document will also discuss the essential elements for creating a popular mobile food delivery application. Any mobile application has three fundamental sections: the user interface, server setup, and third-party integration.

Chapter 2

Problem Statement

The spectacular growth of mobile apps for online ordering and delivery has brought a structural change in how the food business is being done. Never before had customers had it so easy, and never before have restaurants been able to generate more revenue than they do now. But like all good things, these come with a share of ills and problems associated with them, which, if handled poorly, could decelerate the healthy growth of this industry. These problems are associated with technology, operations, economics, and social aspects.

2.1 Technological Challenges

Food ordering apps require a scalable, stable, and robust technology backbone to ensure hassle-free operations. App crashes, data security, and inefficient user experience can drive away your customers and affect their trust level with your brand. Also, an ordering app involves multiple third-party integrations (to facilitate payment integrations, geolocation integrations, etc.) to ensure flawless app operations. A major technical challenge is keeping track of all these integrations, maintaining data consistency, and ensuring data safety.

2.2 Operational Challenges

Managing the logistics, and getting the food delivered on time and in pristine condition is an ongoing challenge. Orders vary drastically, traffic is unpredictable and the time it takes for the restaurant to prepare the order adds to the delivery time. All these factors combined result in late deliveries disappointing the customer. Also, multiple operators are involved in the chain (restaurants, delivery boys) and customers whose real-time tracking and communication are of utmost importance.

2.3 Economic Challenges

Food ordering apps have opened new avenues for restaurants to generate more revenue. But they also charge hefty commissions leaving the already low profit margins of restaurants even more squeezed. This economic challenge can be very impactful for small independent restaurants. Also, with the emerging on-demand gig economy,

there are concerns about fair remuneration and poor working conditions of the order delivery staff who are often hired as gig employees.

2.4 Social and Regulatory Challenges

Impacts of widespread acceptance of food ordering services include changes in food habits and an incline in the lazy, sedentary lifestyle of the people. Regulatory compliances include adhering to local jurisdiction food safety standards, hiring compliances, data privacy laws, and whatnot.

Chapter 3

Project Objectives

The central goal of this effort is to design an innovative mobile food ordering and delivery application that responds to users' needs for convenience, streamlines processes, and works to support local establishments. A unique feature of this application is the integration of advanced conversational artificial intelligence, which promotes streamlined ordering processes, allowing users to have more natural conversational flows while getting personalized suggestions. Basic features include an intuitive interface, efficiency optimizations for minimizing wait times, being able to order from anywhere at any time, and an important feature that allows ordering in advance for pickup or delivery.

Through leveraging modern mobile and backend technologies, this system hopes to dramatically improve customer satisfaction in terms of an enhanced direct and streamlined ordering process, while at the same time enabling local businesses to expand their footprint and increase their customer base. This effort focuses on minimizing processing time and giving an intuitive user interface that improves general usability, setting an overall new benchmark for accessible and reactive food service apps.

Chapter 4

Literature Review

4.1 Backend Infrastructure and Data Management

Some studies have examined the technical requirements necessary for successful meal delivery applications. Li et al. (2018) pointed out that e-commerce enterprises should build strong backend systems to deal with massive traffic, notify users of the current status of their orders, and communicate with other service providers. Cloud deployment and micro-service structure were also suggested for agility. In the area of security and privacy, Kumar and Singh (2019) noted that attacks like data tampering and fraudulent transactions are possible in meal delivery applications. Accordingly, they recommended the use of suitable encryption techniques and application programming interfaces.

4.2 Creating a Stable, Scalable, and Secure Platform

While developing any food delivery mobile app, you need to ensure that there is a strong technological platform that can handle massive traffic loads during your business's peak hours. Also, it should scale as more customers begin to use your app without any slowdown or crashing of the app. Security, of course, is another important aspect. If there is a data breach on your vendor's app, it could, in theory, compromise customer credit card numbers and home addresses. Such an event could destroy the trust of app users. Thus, you need to ensure that there are strong cybersecurity measures in place including data encryption and regular security audits.

4.3 Mobile Applications and User Experience

Persaud and Azhar (2012) researched the importance of mobile applications for food delivery from the consumer's perspective. According to the report, mobile applications are widely used since they increase customer comfort and contentment. The survey also revealed that interface and a pleasant client experience are critical components to the success of any meal delivery mobile application. Alalwan (2020) evaluated the impact of usability and plan variables on clients' persistent intentions.

The analysis discovered that routes and all-around designed visual components in food delivery mobile applications had a high likelihood of customer retention.

4.4 App Crashes and Inefficiencies

If the application fails or is sluggish, the user will have a terrible experience and may switch to a competitor's app. To minimize the app crashes, the app must be tested through and through and optimized for different devices and platforms. Similarly, the user flow must be good enough that users can easily find meals and place orders within the app without wandering here and there or doing other hassles.

4.5 Third-party integrations

Subsystems for payment, geolocation, and order assignment are crucial for food delivery apps. Usually, they are implemented by third-party vendors and the mobile app developers have to integrate mobile apps with those services. Data exchange with all subsystems is often not formalized and strongly depends on the technical details of the implementation. Moreover, the security of the user data should be maintained at the highest level. Payment gateway should be compliant with the security standards to prevent data leaks and hijacking of financial transactions. The geolocation subsystem should accurately determine the position of the restaurant, customer, and driver to prevent order delivery errors. It's a technical challenge to integrate the mobile app with third-party subsystems and maintain smooth app work.

4.6 Current Competitors

4.6.1 Uber Eats

Uber Eats is an online food ordering and delivery service by Uber, the global ride-sharing company, introduced in 2014. Food is delivered to customers through partners (couriers) who use cars, scooters, bicycles, or by walking. It is available in more than 6,000 cities within 45 countries and territories as of 2021.
Pros:

1. Uber Eats is available in more than 6,000 cities, a huge number. So, the Uber Eats app is very much accessible to people all around the globe. Also, being so popular, you will get whatever you want from wherever you are. The most demanded meals and restaurants in any territory are available there.

2. You can log in to the Uber or Uber Eats apps from the same page. Therefore, it offers a great advantage of using a single platform for your transportation needs whether you need a ride or want to eat out and have a meal delivered. This is a very convenient option, especially in a busy schedule.

3. If you are an existing Uber user, then using the Uber Eats app will feel like a breeze as the pattern and methodology are the same.

4. Uber has already deployed this app on both iOS and Android platforms. So, you won't face any problem downloading and using it on your smartphone.

5. The Uber Eats app is very neatly organized with all necessary details and options available in appropriate tabs. So, you won't get lost and it saves time.

Cons:

1. Uber Eats doesn't deliver out of state though their network is huge, they are quite strict with this rule.

2. Delivery charges are high: Many times delivery charges are high compared to the meal you are buying which becomes a big deterrent in using the service repeatedly.

3. Restaurants are highly competitive: Since multiple restaurants are showing up on the same app, the competition is fierce. Restaurants that partner with Uber Eats spend quite a bit on in-app promotions and marketing to get noticed.

4. Uber Eats Still Isn't In Most Rural Areas: Uber Eats currently has a very limited rural footprint, so customers in smaller towns and cities still can't order food through the app.

5. Restaurants Say Uber Eats Is Too Expensive: Restaurants that choose to partner with Uber Eats claim it's very expensive to use the service. Restaurants are charged a 350 dollars fee to get started, and then a 15 percent to 30 percent marketplace fee on each order. That adds up pretty quickly and can eat into profits.

4.6.2 DoorDash

DoorDash, Inc. is an American privately held company based in San Francisco that develops and operates an online food ordering and delivery platform. The company operates under the name DASH. DoorDash has a 56 percent market share in the U.S. food delivery sector, making it the country's largest food delivery platform. It also commands a 60 percent market share in the convenience delivery sector. The company's platform as of December 31, 2020, was used by 450,000 merchants, 20,000,000 consumers, and one million delivery couriers.
Pros:

- Wide Area of Service: With operations in over 4,000 cities, DoorDash covers a vast area – both for customers and restaurants alike.

- Low Customer Fees: In most cases, DoorDash has lower customer fees than its competitors. However, customer fees vary by restaurant and area, so it ultimately depends on which restaurants and areas are targeted.

- POS System Integration: DoorDash has partnered with many of the popular Point of Sale (POS) systems, including Toast and Square. This helps restaurants with logistics and ensures smooth operation during order processing and delivery tracking.

- No Activation Fees: DoorDash does not charge any activation fees or credit card processing fees, which can help restaurants reduce their initial costs.
- 30-day Commission Free

Cons:

1. Service Fees: On top of the delivery fees, DoorDash also charges its users service fees which makes the order pricier for the customer. These fees tend to accumulate for frequent DoorDash users.
2. App glitches: User reports have noted app crashes and slow loading times, which is annoying when placing orders or tracking food delivery.
3. Inconsistent Delivery Times: During peak times, DoorDash delivery times are often inconsistent and can be delayed by hours, creating an unstable user experience.

4.6.3 Grubhub

Grubhub Inc. is an American prepared food ordering and delivery website and mobile app headquartered in Chicago, Illinois. It was founded in 2004 and has been a subsidiary of Just Eat Takeaway of the Netherlands since 2021. Grubhub has faced antitrust price-fixing allegations, listings of restaurants without their consent, and the alleged misclassification of employees. Grubhub Seamless is publicly traded as of April 2014 and listed on the New York Stock Exchange (NYSE) under the symbol GRUB. It had 19.9 million monthly active users as of 2019 and 115,000 partner restaurants listed in 3,200 cities across 50 US states.

Advantages:

- Market Demand: Operating in 4,000+ cities and working with 265,000 restaurants means that GrubHub has tremendous market penetration and diversity, translating to more customers for your restaurant if located within a city where GrubHub operates.
- Order Links: Restaurants have the option to place order links directly on their website or within email marketing campaigns, eliminating the marketing commission that GrubHub usually charges restaurants, resulting in potential savings on marketing commission fees.
- Easy to Use App: The GrubHub App is intuitive and straightforward.
- No Activation Fee: Unlike some competitors, there is no cost to activate your restaurant with GrubHub, saving money upfront.
- POS Integrations: GrubHub works with many POS options including Focus, Micros, and Toast, making order management smooth and efficient.

Drawbacks:

- Service Fees: Grubhub charges high service fees that add to the overall cost of the order, potentially turning off price-sensitive customers from frequent usage.

- Order Accuracy: Anecdotal reports suggest that occasionally flawed orders are being delivered, leading to customer dissatisfaction and hassle.
- User Interface: The app interface looks crowded and confusing to some users, leading to avoidable places and steps to place an order. A more streamlined user interface may improve the user experience.

4.7 Operational Challenges

Supply Chain Management: From the moment the order is placed, supply chain management dominates the race to deliver food on time with the same freshness and quality. Delivery routes have to be managed, traffic has to be managed, restaurants have to be prepared to accept the food in the minimum possible restaurants, and waiting time at the restaurants has to be reduced. However well-designed the algorithms or developed the real-time tracking technology, they will eventually become obsolete. You need to continuously update your algorithms and regularly test your technologies to ensure they are relevant and reliable. Technologies that optimize food delivery need constant fine-tuning to keep up with dynamic conditions and requirements.

Timely Delivery: Delivery within a specified time is a complex issue with many external factors to consider. If demand is very high during lunch or dinner peak hours, it can lead to an overload of orders that can clog your optimization system and consequently delay some orders. Traffic conditions, especially in metropolitan areas, can also drastically affect delivery time. Another major source of unpredictability in food delivery is the preparation time by the restaurant, which varies greatly depending on the type of restaurant. The solution to all these problems lies in perfect real-time communication between all stakeholders: the restaurant, the delivery boy, and the customer. The restaurant must appropriately inform the food delivery service about the estimated preparation time, the delivery lad needs real-time traffic reports and several route alternatives to choose from, and most essentially, the customer needs to know what's happening with their order to keep expectations in check. Giving tasks to delivery personnel in a flexible way helps balance the workload and cuts down on the time they're not doing anything. This makes deliveries quicker and more precise, and customers also get updates when there's a change in their order status.

Food Quality: Maintaining food quality during transit is the next big challenge. A majority of the food items need to be transported within a certain temperature range and handled hygienically without any spoilage or contamination. While developing niche brands, specialized packaging that can retain heat or cold and using insulated bags for delivery can address this issue but they increase the operational cost.

Chapter 5

Business Models and Market Mechanisms

5.1 Revenue and Pricing Schemes

The revenue and pricing schemes implemented by food delivery applications influence both their financial success and that of the restaurants they partner with. Goh and Yeo (2019) identify three primary schemes: commission-based, subscription-based, and hybrid systems. Commission-based schemes involve the platform charging restaurants a percentage of the order, placing significant financial pressure on the restaurant. Monthly fee plans provide a steady income from restaurants paying a set rate to be listed, regardless of order volume, but may not be suitable for apps with few users. Hybrid models combine a flat fee with a per-order charge, offering stable income and potential growth with more sales (Goh and Yeo, 2019).

5.2 Competitive Landscape

Datta et al. (2020) investigate the competitive environment in the online meal delivery sector, focusing on top firms: Uber Eats, DoorDash, and Grubhub. They find that these companies use various techniques to market their platforms, such as pricing cuts, exclusivity relationships with restaurants, and significant spending on television advertising. These strategies result in market share consolidation and may create monopoly situations, disadvantaging smaller rivals and deterring new entrants. The competitive dynamics have implications for consumer welfare, limiting choice and increasing costs for restaurants (Datta et al., 2020).

Chapter 6

Novel Advancements and Upcoming Trends

6.1 Technology

Advancements in artificial intelligence (AI) and machine learning will revolutionize food delivery applications, improving operational efficiency and strategic decision-making (Zhang et al., 2022). AI algorithms can optimize delivery routes, anticipate customer orders, and provide accurate demand estimates, leading to higher service quality and lower costs.

6.2 Industry Trends

Cloud kitchens are emerging as a significant trend in the food delivery sector (Wang and Wong, 2021). These purpose-built facilities enable restaurants to prepare food exclusively for delivery orders, reducing overhead costs and improving operational efficiency. Cloud kitchens are expected to continue growing in popularity, catering to the increasing demand for food delivery.

6.3 Upcoming Challenges

The food delivery industry will face challenges related to labor laws, environmental impact, and technological optimization (Roberts and Smith, 2020). Compliance with labor laws and fair wages for delivery workers, reduction of packaging waste, and optimization of user experience will be crucial areas for the industry to address in the future.

Chapter 7

Working Plan

This application will be built using Flutter as the backend and Dart as the frontend. The functional requirements and non-functional requirements analysis are as follows:

7.1 Must-Have Features in a Food Ordering and Delivery App

User Registration and Login

- Allow users to sign up with their email IDs, phone numbers, or social media profiles.
- Perform secure authentication of users upon login (e.g., password, OTP).

List of Restaurants

- Display the list of restaurants to order from based on the current location of the user.
- Let users filter and sort the restaurants by cuisine type, ratings, delivery time, and price.

Browse Menu

- Show the menu items of the selected restaurants with their images, descriptions, and prices.
- Let users configure their ongoing orders (e.g., toppings, cook preferences).

Place Order

- Let users add menu items to the cart.
- Present the aggregated order before checkout, including itemized costs and delivery fees.
- Let users complete the order and receive an order confirmation message.
- Let users pre-book the food of their choice for any scheduled time and location.

Payment Integration

- Offer multiple payment options like credit/debit cards, digital wallets, and cash on delivery.
- Securely process the payment, and the user must get a payment receipt.

Track Order

- Let users know the status of the order (e.g., received, preparing, out for delivery).
- Let users locate the delivery boy on the map.

Notification

- Send order confirmation, status updates, and order arrived push notifications.
- Alert users about promotions, discounts, and special offers.

Reviews and Ratings

- Let users rate and review the restaurants and delivery partners.
- Display reviews and ratings to help users place smart orders.
- Let users rate every food item that they order.

Support

- Have a help section with FAQs and contact facility to reach out the customer support.
- May include live chat or in-app messaging features with customer support executives.

Manage Profile

- Let users update their personal details, delivery addresses, and payment methods.

7.2 Non-Functional Requirements in a Food Ordering and Delivery App

Performance

- Survive a minimum peak load of 10,000 concurrent users.

Scalability

- The app architecture must be scalable to add more users and restaurants.
- The backend must scale to process more orders and data without performance degradation.

Security

- Employ a secure connection (e.g., SSL/TLS) to protect the information transmitted over networks.
- Address data protection regulations (e.g., GDPR) and protect the users' privacy.
- Implement adequate authentication methods to prevent unauthorized access.

Maintainability

- The app code must be organized and commented on to facilitate easy maintenance and updates.
- Accommodate continuous integration and deployment (CI/CD) for frequent bug fixes and app updates.

Compatibility

- Must work on the latest versions of major mobile operating systems (iOS and Android).
- Must work on multiple device types and varying screen sizes (smartphones and tablets) and must provide a uniform user experience.

Compliance

- Adhere to local and national laws, regulations, and standards, including food safety, delivery laws, and digital transactions.
- Must adhere to the industry standards for data protection and payment processing (e.g., PCI DSS).

Chapter 8

Explanation of Analysis, Requirements and Design

Use Case Diagram

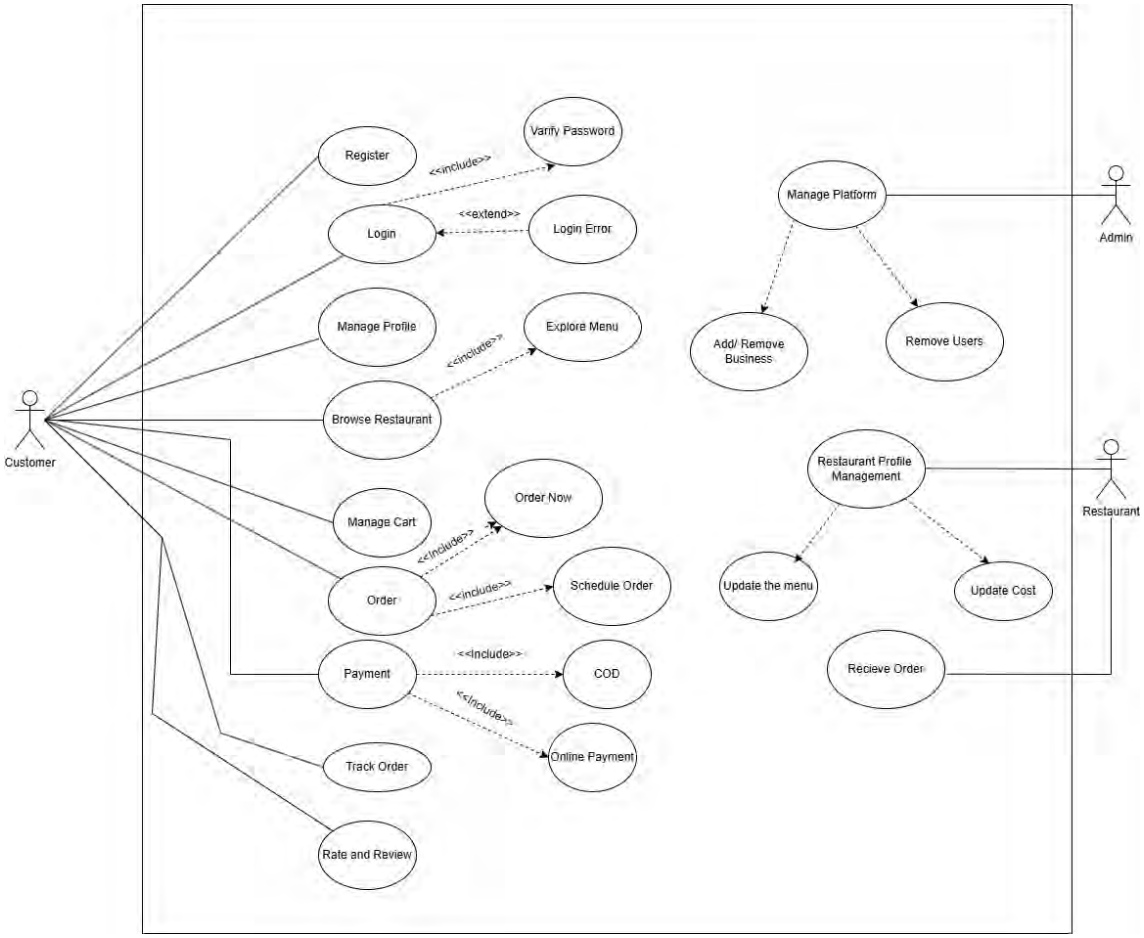

Figure 8.1: Use Case Diagram

Actors:

- **Rider:** A person responsible for quickly picking up restaurant orders and delivering them to the customers.
- **Customer:** A person who orders food from restaurants through the application interface. Most functions, such as shopping cart management, setting up the time frame for delivery services, imposing control on orders, searching through various lists of restaurants, and posting comments on their services, are interactive.
- **Restaurants:** The clients who demand food cooked in restaurants place orders through the app. The clients place the orders and the system confirms them with the restaurant before making arrangements for the rider to deliver the food. The clients give ratings to the restaurants based on their satisfaction, quality, and the amount of food provided.
- **Admin:** The system administrator manages and controls the activities of the given platform. They deal with order issues, manage users (riders and consumers), and ensure that the system works properly. They also add new businesses that want to join and use the system.

Use Cases:

1. User Registration (Actors: Customer, Rider)

The Description:
Both Customers and riders need to go through the user registration phase. By doing this, the application opens for their use, allowing them to place and monitor their orders. They have to input their emails and passwords as a login process.
The Functionality:

- The new user must provide certain information in the registration form such as name, email address, password, and phone number. Users may also create an account with a Google account to access the application.
- After registration, the users of both actors can log in to their respective accounts.

2. User Login (Actors: Customer, Rider, Admin)

The Description:
Registered customers and riders login with credentials and passwords to get services such as managing order requests and delivery status. The admin also has their own set of credentials which allows them to log in to the administration back end for managing users and controlling the entire activities of the platform.
The Functionality:

- Authentication to the application using email/password.
- Admin has higher-level access, managing both customers and riders.

3. Find Restaurants (Actor: Customer)

The Description:
As a customer, one would be able to check the restaurants available according to users' location or based on the type of cuisine that they serve. To make your food ordering experience seamless and to find the restaurant user's needs, we have a filtering option. Moreover, they can share their dish experience and rate the restaurants.
The Functionality:

- Filters, depending on the location of a user according to ratings for each restaurant received with that score, and type of cuisine.

4. Explore Menu (Actor: Customer)

The Description:
Once the customer has selected a restaurant, they can navigate through the app and choose what food items to order. This has a list of available items, descriptions, and prices.
The Functionality:

- Display food types (starters, mains, desserts, beverages).
- Increases customer cart items based on their selection.

5. Manage Cart (Actor: Customer)

The Description:
Customers can add or remove items they are willing to order and manage them in the cart. The cart displays all the selected and added items, including the subtotal amount including VAT and tax before proceeding to checkout. The cart will be empty after delivery and orders will stay in records.
The Functionality:

- Setting up a user to add, remove, or change the quantity of items. Apart from confirming before ordering, also canceled items.
- Shows the full price including all taxes and delivery costs.
- Customers can apply coupons before submitting an order.

6. Place Order (Actor: Customer)

The Description:
After the cart is complete, customers can make an order. All it does is ask for delivery information including address and payment, which it then confirms before sending the order to the restaurant.
The Functionality:

- Verifies delivery address and timing.
- Special instructions (e.g., do not use peanuts while cooking or do not use shrimp).

7. Track Orders (Entities: Customer, Rider, Restaurant, Admin)

The Description:
Whether the customers, riders, or restaurants all can see in real-time at which stage their order is. This way, customers are informed about the status of their food preparation and delivery; meanwhile, riders know when and where to pick up or drop off orders.
The Functionality:

- Live status updates on the order (preparing, out for delivery, delivered).
- The apps come with maps and directions for riders to use when making deliveries.

8. Payment Processing (Actors: Rider, Restaurant, Customer)

The Description:
Customers can pay in a few different ways: credit cards, cash on delivery, and online. Payment receipts and service tracking are handled by restaurants and riders.
The Functionality:

- Accepts a range of payment options.
- Resolves disagreements regarding credit or refunds.
- Gives the client a receipt and keeps track of the money they have paid the eatery.

9. See Order History (Customer, Restaurant)

The Description:
By looking at their prior orders, customers may see how much they've spent and place new orders. By examining past orders, restaurants can observe sales data and client preferences.
The Functionality:

- The items, date, and price of the order are shown.
- Facilitates quick reordering.

10. Rate and Review (Actor: Customer)

The Description:
After receiving the food, customers can rate and review the restaurant and delivery. It will improve the service quality and keep other customers well informed.
Functionality:

- Rating restaurant and delivery: e.g., star rating - 1 to 5.
- Written reviews describing the customer's experience.
- Colors restaurant ratings and app performance metrics.

11. Manage Profile (Actors: Customer, Rider, Admin)

The Description:
Users can maintain their personal information regarding delivery addresses and contact data. Admins will be able to review and edit user profiles to debug any problems that might arise.
Functionality:

- Allow the user to edit their personal information: name, e-mail, and cellphone number.
- Allow for different delivery addresses and different ways of payment.
- Let profiles be visible for admins for monitoring/support purposes.

12. Access Customer Support (Customer, Admin)

The Description:
Help can be made available via the application for questions regarding issues with orders, payments, and account-related problems. There is an area for frequently asked questions where the user can go through if his problem is common and how he can fix it without admin interference. Admins are allowed to respond to support tickets and resolve issues for customer satisfaction.
The Functionality:

- FAQs, Chatbot Support, or direct human support.
- Admins manage the queries and ensure timely responses.

Entities:

- **Restaurant:** Represents the participating establishment of the system.
- **Client:** Represents clients who place orders.
- **Rider:** This represents delivery personnel who perform the order.
- **Admin:** In this case, an admin means a system administrator who manages the system.

Data:

- **Restaurant Information:** Restaurants' name, address, menu items.
- **Customer Information:** Includes all customer names, delivery addresses, and phone numbers.
- **Rider Information:** Description of the riders, including the name, telephone number, and type of vehicle.
- **Order Information:** Includes the client, restaurant, products ordered, delivery address, and the status of the order.

- **Menu Information:** Information about the dishes offered by the different food outlet establishments.

Relationships:

- **Restaurant Management:** Responsible for the addition, modification, and removal of restaurants, as well as maintenance of their information.
- **Customer Management:** The department responsible for maintaining customer information and verification.
- **Order Management:** Involves all the activities of order management, including allotment of riders and updating information on orders.
- **Order Database:** This database maintains all the information about orders.
- **Restaurant Database:** This database maintains all the restaurant information.
- **Client Database:** This database maintains all the information about clients.

Data Movement:

- The customers utilize the system to place an order.
- The Order Database is used for storing data on orders.
- Order Management assigns a rider to the order.
- The rider delivers the order to the client after picking up the order from the restaurant.
- The new status of the order is updated in the Order Database.

Dataflow Diagram

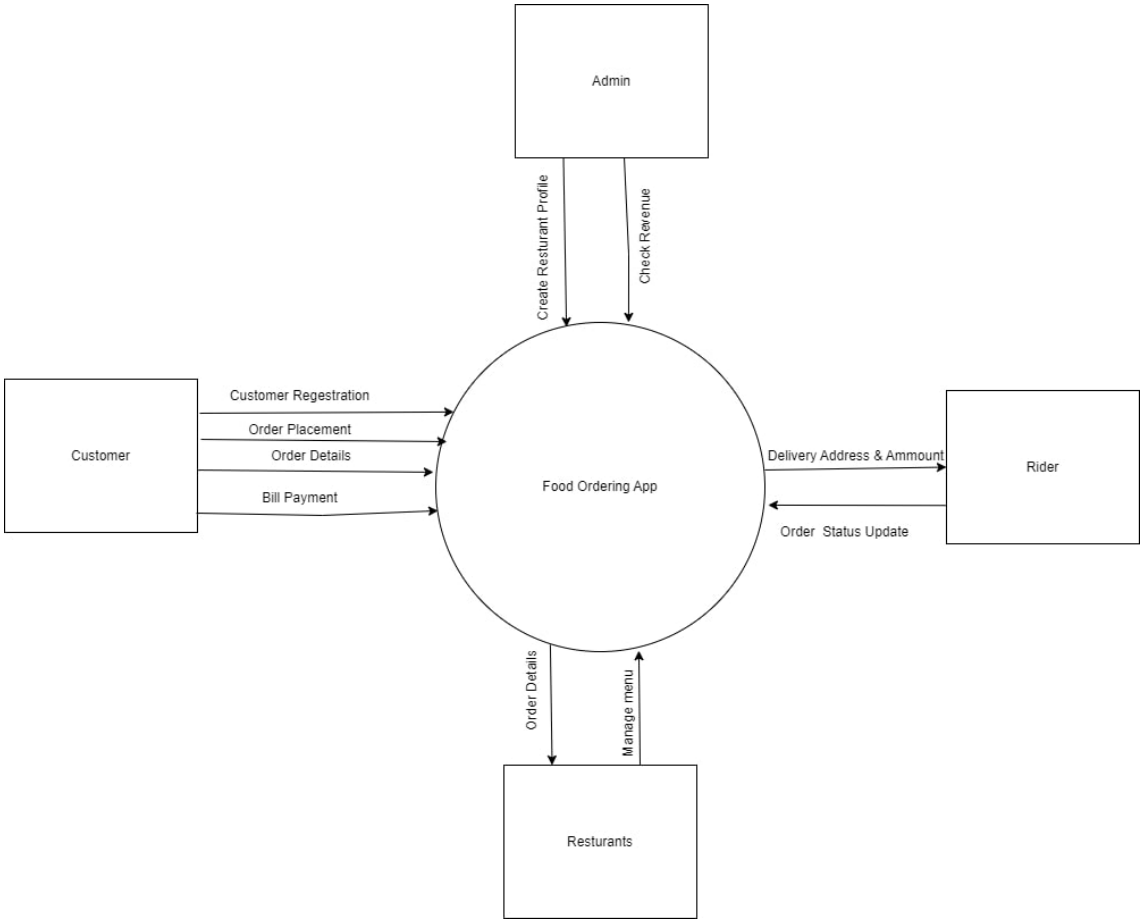

Figure 8.2: Data Flow Diagram Level 0

Food Ordering Application (Middle Circle):

Represents the primary process, which is the food ordering application. It handles the core functions: of food ordering, order tracking, payment processing, and communication with customers.

Outside Entities [Boxes around the Middle Circle]:

Customer:

The customer accesses the application to browse through restaurants, place food orders, and manage personal information.
Customers can register, log in, place an order, make payments, and track their orders through the application.

Restaurants:

Restaurants receive orders from customers via the app, prepare the meals, and update the app with the status of the order.

Rider:

Riders deliver food from the restaurant to the customer. They use the app to view delivery details and update the order status once the delivery is complete.

Admin:

The system administrator monitors the activities of all users (customers and riders), tracks system performance, and resolves issues to ensure the smooth functioning of the app.

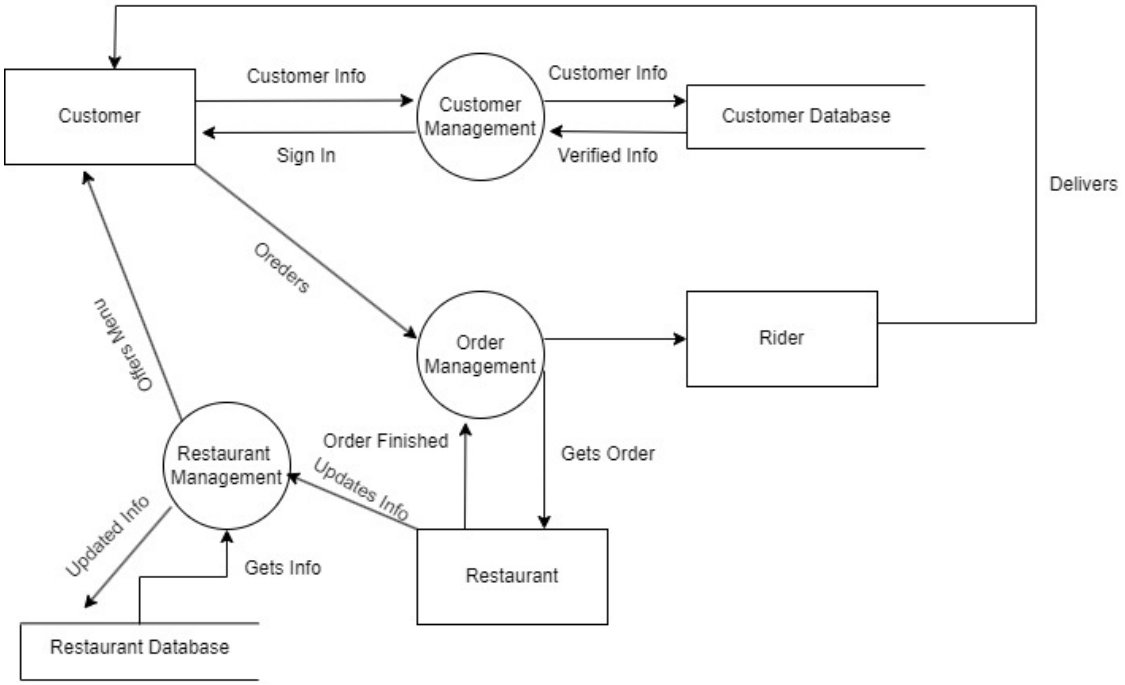

Figure 8.3: Data Flow Diagram Level 1

Information Flow:

Customer to Food Ordering App:

Customers interact with the app by registering, logging in, browsing restaurants, placing food orders, and making payments. The app responds by displaying menus, confirming orders, processing payments, and allowing customers to track their orders.

Restaurants to Food Ordering App:

Restaurants receive order details via the app, prepare the food, and update the app with the preparation status and estimated readiness times.
The app forwards this information to both the customer and the rider.

Rider to Food Ordering App:

Riders receive delivery instructions and customer location updates via the app. They update the system once the delivery is completed, and the app informs the customer.

Admin to Food Ordering App:

The admin manages and monitors all activities within the app, from customer registration to order tracking, ensuring smooth operations and addressing any issues that arise.

Activity Diagram

User

- **Search Restaurant:**
  It filters restaurants by location, type of food available, ratings, and delivery time.
  If suitable options are found, it uses search filters and maps. It can view the restaurant details.
  Menu items are viewed along with their description, prices, and ratings.
  Read the customer reviews and the popularity of the restaurant.
  The restaurant's current ongoing promotions and discounts can be viewed.
- **Add Items to Cart:**
  The customer identifies the items to be ordered along with the quantity.
  Adds items to his cart with options to check out later.
- **Manage Cart:**
  Modify the quantity of each item remove an item/clear the cart.
  Applies coupon or promo code, if available. The total cost of the order would be viewed, inclusive of all taxes and delivery charges.
- **Place Order:**
  Delivery address, contact information, and mode of payment provided.
  Summary of an order viewed and confirmed.
  Special instructions or requests are added for the rider.
- **Track Order:**
  The customer tracks, through the application or website, in real-time, the status of their order.
  The customer gets notified when the order is being prepared, picked up, and delivered.
- **Rate and Review:**
  Feedback of restaurant, quality of food, service of rider
  Rating to a restaurant with comments for other customers.

System

- **Restaurant Search and Filter:**
  Retrieve restaurant data from the database.
  Filter restaurants based on the search applied by the customer.
  Rank restaurants based on the relevance and popularity of restaurants.
- **Order Information Validation:**
  The cart is not empty, the address is valid, mode of payment is provided.
  A restaurant selected by a customer must be opened and receiving orders.
- **Total Amount Calculation:**
  Calculates the total price, inclusive of item prices, taxes, delivery costs, and all available discounts or promotions.

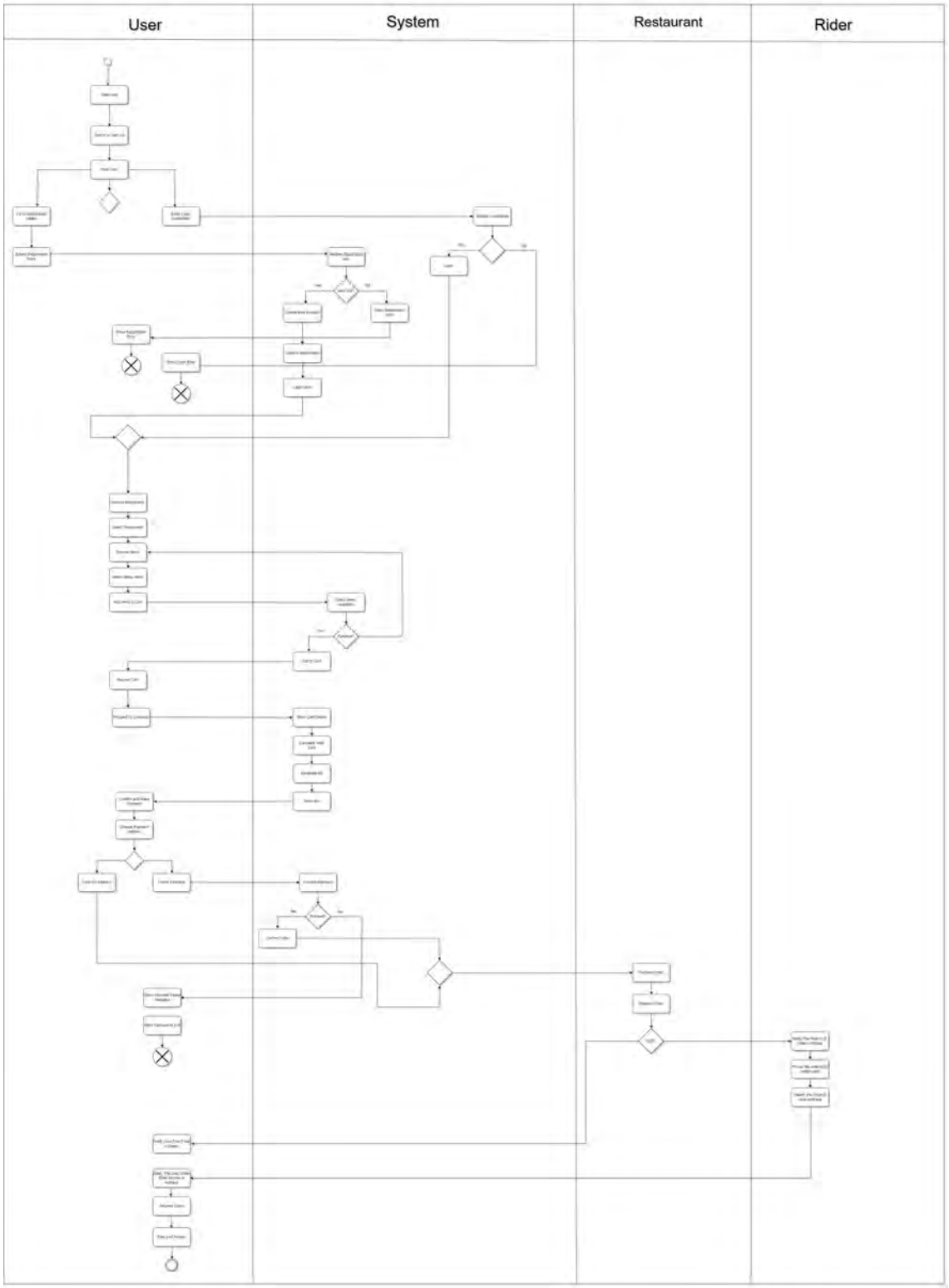

Figure 8.4: Activity Diagram of the Food Ordering App

- **Notify Restaurant:**
  Forward details related to an order to a selected restaurant via API or messaging.

- **Assign Rider:**
  Executes algorithms for assigning the most suitable rider concerning location, availability, and ratings.
  Will consider some of the factors that include distance, delivery time, and workload of riders.

- **Update Order Status:**
  Synchronizes, in real-time, the status of an order upon moving from pending to accepted, preparing, out for delivery, or delivered.

- **Process Payments:**
  Pays online through secure gateways.
  Clears the cash on delivery against the payment.

- **Send Notification:**
  Sends push notifications or 'in-app' notifications to the user, restaurant, and rider regarding updates on the order.

Restaurant

- **Receive Order Notification:**
  To get notifications from the system regarding new order arrivals.

- **Prepare Order:**
  Prepare the food items according to the order.
  Ensure quality and accuracy in the order.

- **Mark Order as Ready:**
  Notify the system when the order is ready.

Rider

- **Accept Order:**
  The Rider gets the assignment of the order and thus accepts the same.

- **Navigate to Restaurant:**
  GPS navigation towards the location of the restaurant.

- **Pick Up Order:**
  Picks up the order prepared from the restaurant.
  Check whether the order type and quantity are correct.

- **Navigate to Delivery Location:**
  GPS navigation to reach the address of the customer.

- **Deliver Order:**
  Delivery of the order at the front door steps of the customer.
  Attends to any special instructions or requests.

- **Mark Order as Delivered:**
  Updates, in the system, order status as delivered.

Decision Points and Conditions

- **Restaurant Availability:**
  It checks whether the chosen restaurant is available or open to take in orders.

- **Rider Availability:**
  A rider is assigned depending on his availability and proximity to the restaurant.

- **Payment Success:**
  It checks whether the payment has been successfully processed or not.

- **Order Acceptance:**
  This checks whether the order has been accepted or denied by the restaurant due to any capacity or staffing issues.

- **Order Cancellation:**
  Provision for cancellation requests by a user or restaurant.

- **Dispute Resolution:**
  In case disputes arise related to order accuracy, payment problems, or issues of delay in delivery.

Sequence Diagram

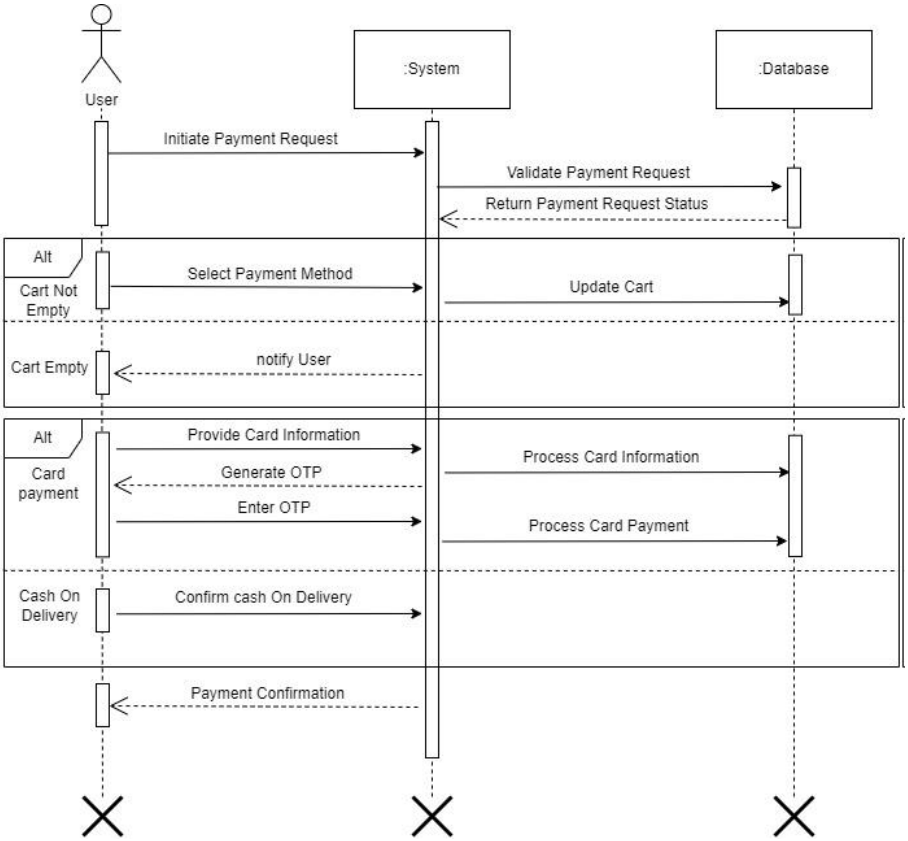

Figure 8.5: Sequence Diagram of user registration

User Inputs: Name, email, username, password.
System: The system checks whether the input meets the criteria or not. For example, the appropriate format of the email, a strong password.
System Generates OTP: In case of successful verification, the system generates one one-time password known as OTP, and it sends it to the users' emails.
User Input OTP: The user enters the OTP received.
System verify OTP: systems verify the OTP entered by the user with the previously generated one.
System saves register information: If the OTP verification succeeds, the system will save the registered information of the user to the database.

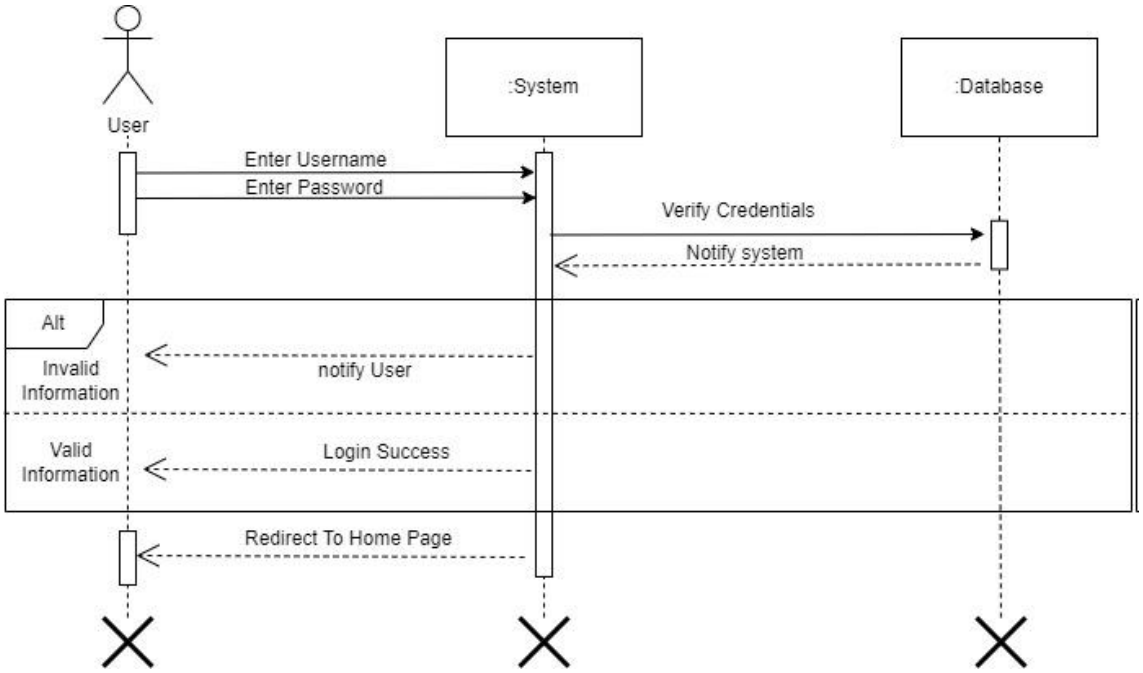

Figure 8.6: Sequence Diagram of user login

Enter credentials: User enters username and password.
System checks the credentials: System forwards credentials to the database.
Database responds: Database tells if credentials are valid or not.
System notifies user: In case of invalid credentials, the System notifies the user.
System proceeds to home page: If valid credentials, the system goes to the home page.

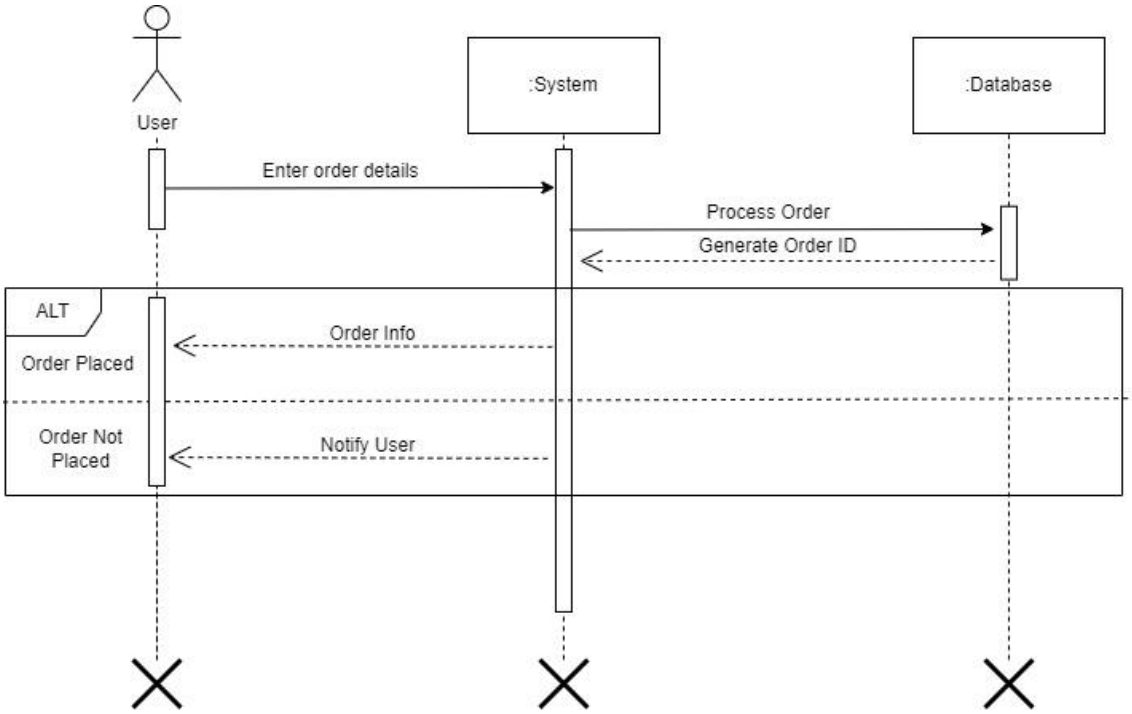

Figure 8.7: Sequence Diagram of placing Order

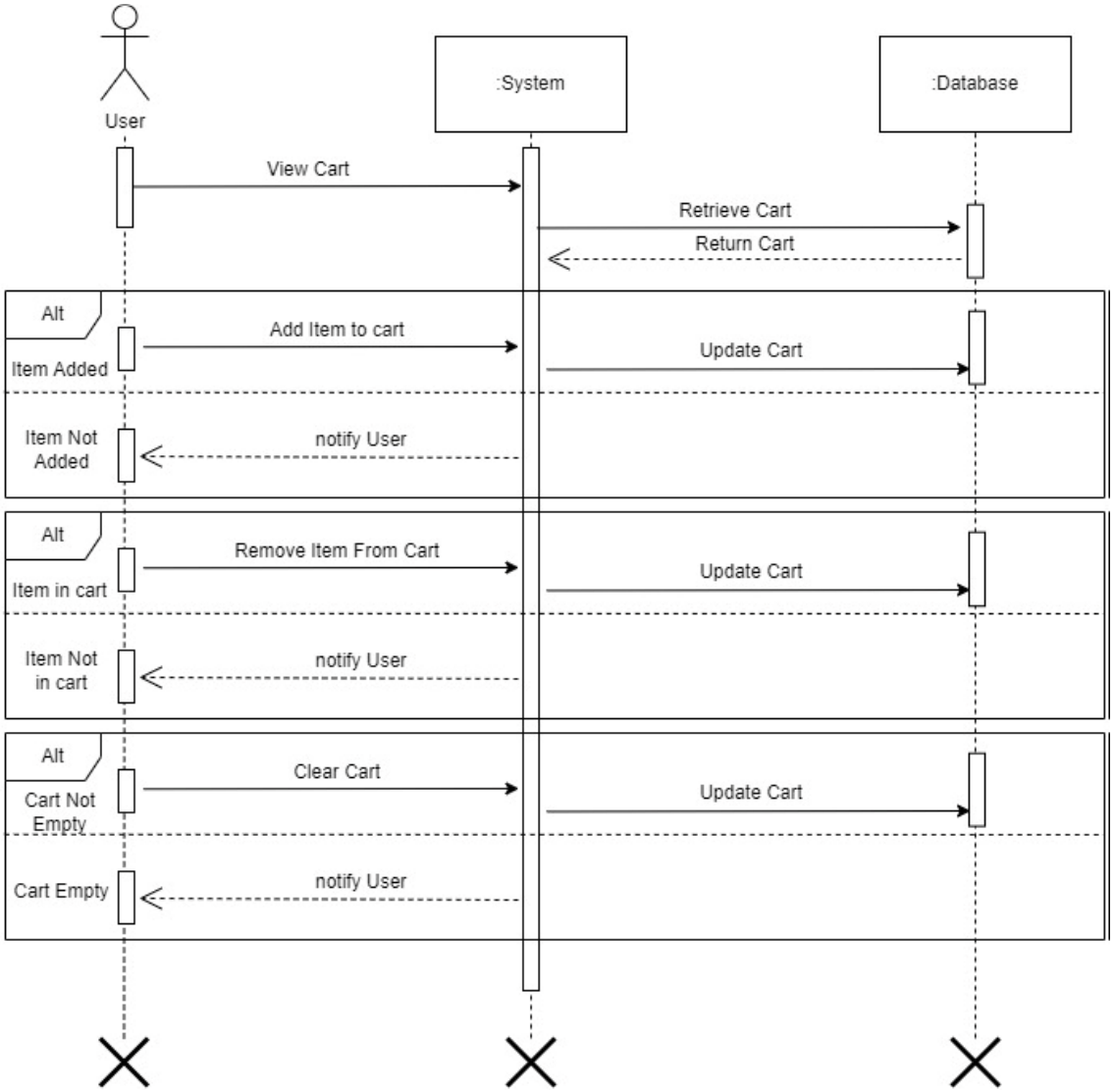

Figure 8.8: Sequence Diagram of cart

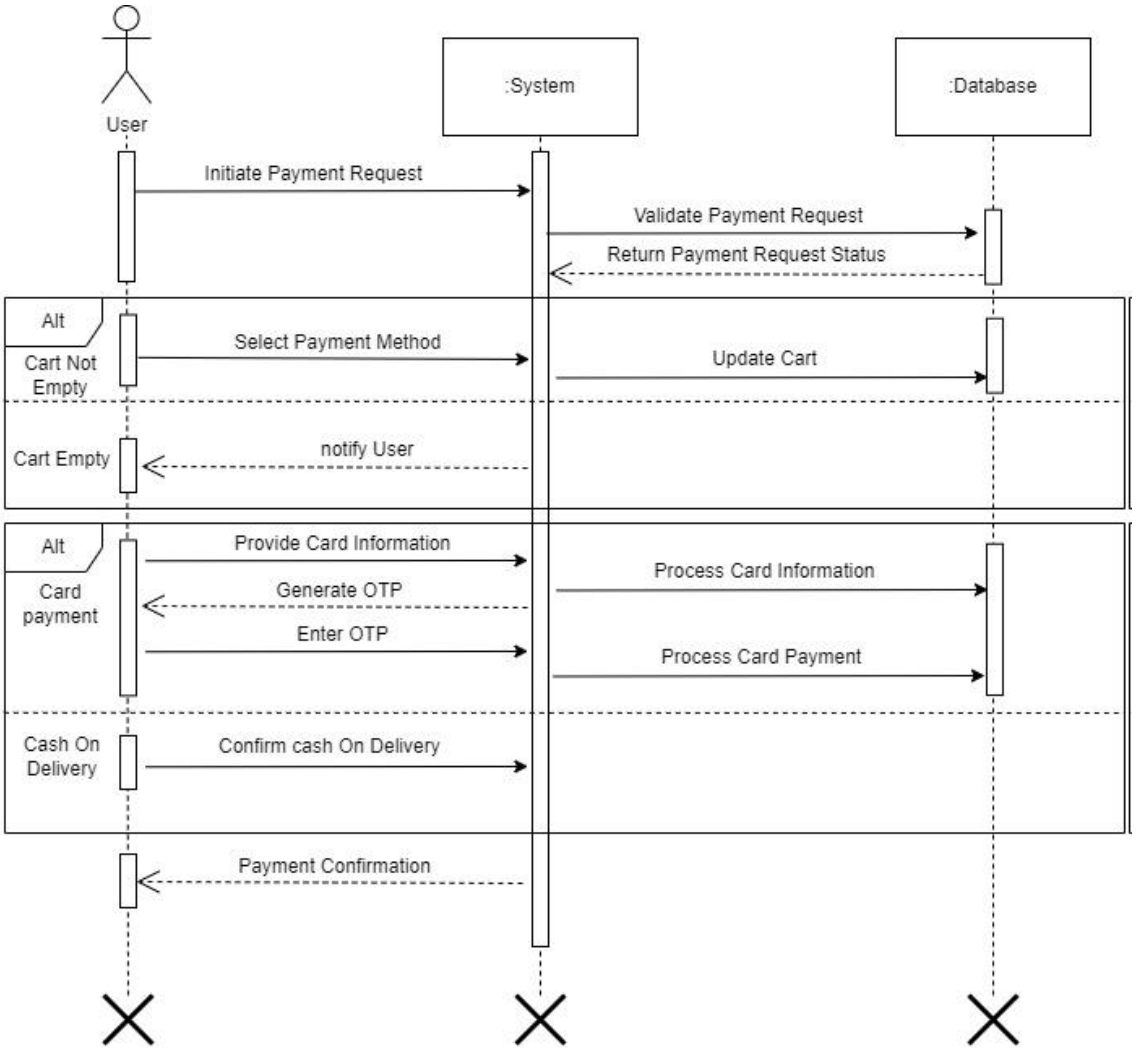

Figure 8.9: Sequence Diagram of Payment

Payment Flow

- **Pay:** User pays.
- **System checks:** System checks. The cart is not empty.
- **System returns:** If the cart is empty, the system tells the user and returns an error. If the cart is not empty, the system goes to the next step.
- **Choose payment:** User chooses payment (card or cash on delivery).
- **System updates cart:** System updates cart.

- **Online Payment:**
  - System generates OTP and sends it to the user.
  - User enters OTP.
  - System processes card and payment.
  - If payment is successful, the system confirms payment and redirects to the order confirmation page. If payment fails, the system tells the user and allows the user to retry.
- **If cash on delivery:**
  - User confirms cash on delivery.
  - System confirms payment and redirects to the order confirmation page.

Chapter 9

Development of the Project

9.1 User Point of View

The mobile app's frontend is built using JavaScript in combination with the React Native framework, significantly enhanced due to the addition of the Expo platform, enabling rapid development, navigation via the Expo Router, and client-side security utilizing crypto-js and expo-crypto. It also uses Supabase's JavaScript client directly to handle user authentication and profile data.
The underlying infrastructure includes a Python application that utilizes asyncio to reinforce its asynchronous features. It communicates with a Llama 3 large language model, that is hosted by RunPod, in order to enable end-to-end conversational AI processes. In addition, it processes all data-related operations, such as user authentication, using Supabase, using httpx for making external API calls, and runs on an underlying web framework like FastAPI.
A few demo images are available to give an idea of how the is currently looking, while the project is currently in development and the designs have not been completed. As the project develops, these demos, which provide an initial visual depiction, might be modified.

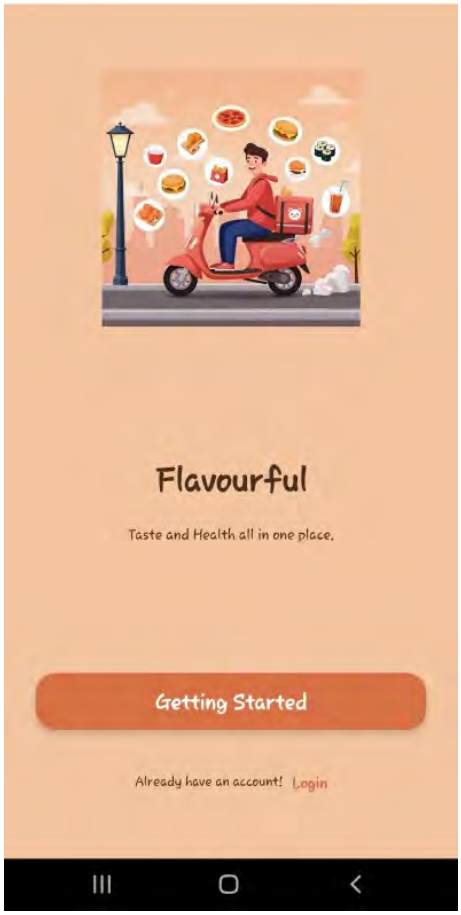

Figure 9.1: Welcome Screen

The Welcome Screen in React Native creates a friendly first impression of the application. It includes necessary dependencies, like expo-router for navigation and expo-status-bar for status bar customization. The ScreenWrapper ensures a consistent layout, while hp/wp enable a flexible layout. The component begins with the use of useRouter() for navigation. Inside a View, an Image is prominently displayed at the top, followed by the application name "Flavourful" and the tagline "Taste and Health in one location," displayed in a structured manner. The footer contains a "Getting Started" button, displayed in an orange color , which leads users to the Sign Up page. Below this, a login prompt is a Pressable text component with "Login," which leads users to the Login page. The StyleSheet.create function organizes styles in a systematic way with regard to layout, typography, and spacing. The background is a warm peach color, with items centered to provide a clean and welcoming look. The login link is displayed in an orange color to make it more visible. This screen is created to provide an uninterrupted user experience through a simple and organized interface.

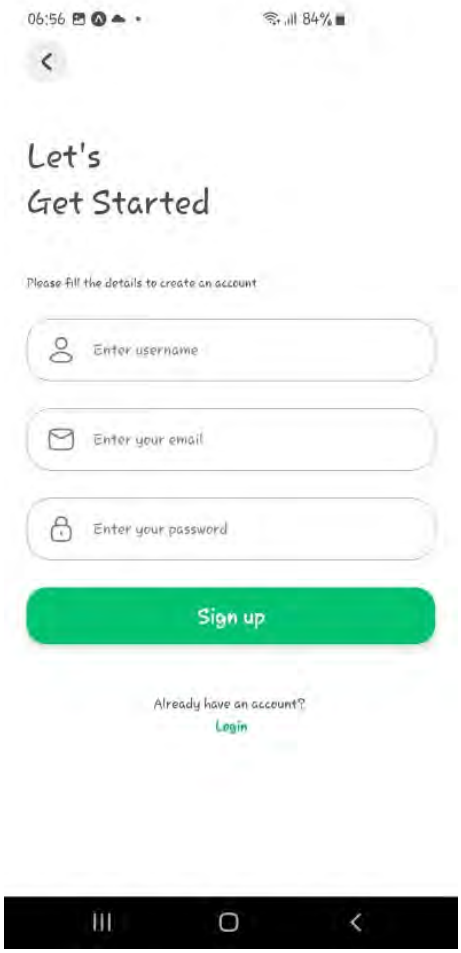

Figure 9.2: SignUp Screen

The SignUp Screen assists in creating a new account for a user. There is a field to enter a username, an email, and a password, and when "Sign Up" is clicked, it validates that all fields have been filled in. It then posts the user information to Supabase in an attempt to sign up, and a loading sign is displayed to inform that it is in progress. The screen contains a BackButton for returning and a footer with a link to the Login view in case a user already possesses an account. Main components include:User information field (username, email, password). Button for form submission and initiation of sign up. An error message in case sign up fails or in case any compulsory field hasn't been filled. Styling is accomplished with StyleSheet, and responsiveness utilizes helper function such as hp and wp.

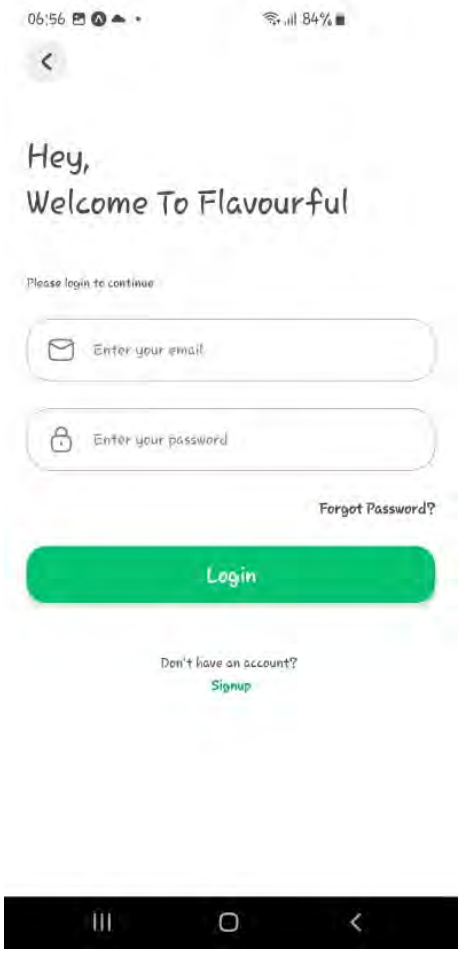

Figure 9.3: Login Screen

The Login Screen enables app users to sign in. There is a field for entering an email and a password, and a "Login" button to submit them. In case a field is not filled, an alert prompts users to fill in all fields. Once a field is submitted, password and email go to Supabase to authenticate with the signInWithPassword function. On failure, an error message appears in an alert when logging in fails. The screen comes with a BackButton for returning and a footer with a sign-up for a new account for new users. Main sections: Email and password fields. A button triggers a start for logging in when filled and a loading sign when in use. An alert for an error message when logging in fails. Designed with StyleSheet, utilizing responsive function hp and wp for fitting in with the screen size.

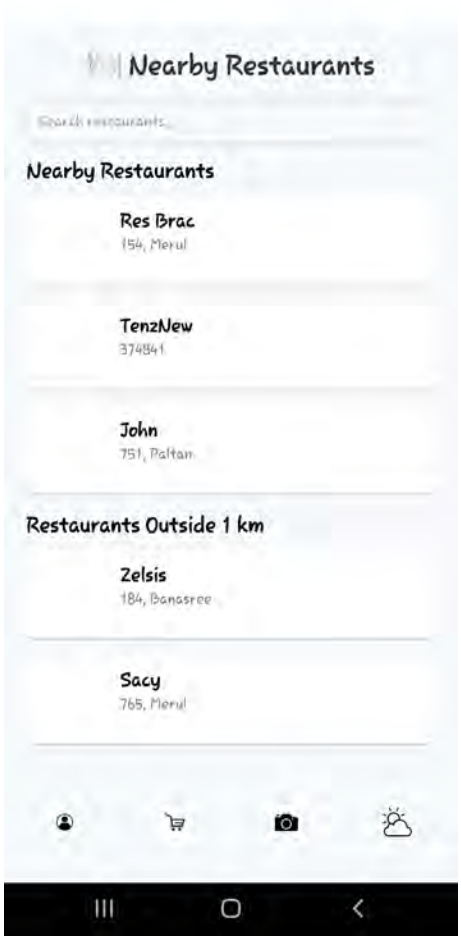

Figure 9.4: Home Screen

The RestaurantList screen displays a collection of restaurants in the vicinity pulled from Supabase, along with a search bar to help users find places by name or location. When someone picks a restaurant, they can see its menu, and the app saves their choice in AsyncStorage.

This screen has: A search bar to narrow down restaurant options. A FlatList to show restaurant cards with pictures, names, and addresses. A profile button that takes you to the editProfile screen. An ActivityIndicator that appears while the app loads restaurant information. When you tap on a restaurant, the app takes you to the RestaurantMenu screen. If your search doesn't match any restaurants, you'll see a message saying "No restaurants available".

Figure 9.5: Sequence Diagram of admin interaction with system

The EditProfile screen gives users the ability to change their profile details. These include their name address, and phone number. It has an option to select "Custom Radius" that allows them to select a certain radius and the home screen will show them a few restaurants within the range. It also has an option to log out and the get Location button is used to get the users current location.

Encryption in Edit Profile: This interface allows for registering a new user by collecting different kinds of information, such as the desired username, email address, geographic location, phone number, and radius—presumably the distance within which the app recognizes nearby restaurants. Two types of data—namely, address and phone number—are deemed to be sensitive; as a result, the application encrypts them before they are stored. This operation involves changing the data in a particular way that makes it unreadable without the respective secret key. All such precautions are taken in order to protect the privacy of the user. For safe operations, each unique field has a randomly derived code, an initialization vector (IV), created by the software. IVs add extra security to the encryption process and are retained with the encrypted data. After all preparations have been completed, the application stores all relevant data in the Supabase database.

Getting User Info: When you first open the screen, it gets the user's ID from AsyncStorage then pulls their profile from Supabase.

Saving Updates: When you hit the "Save" button, the form info gets updated in the Supabase database.

Logout: You can sign out, which wipes your user ID from AsyncStorage and logs you out of Supabase.

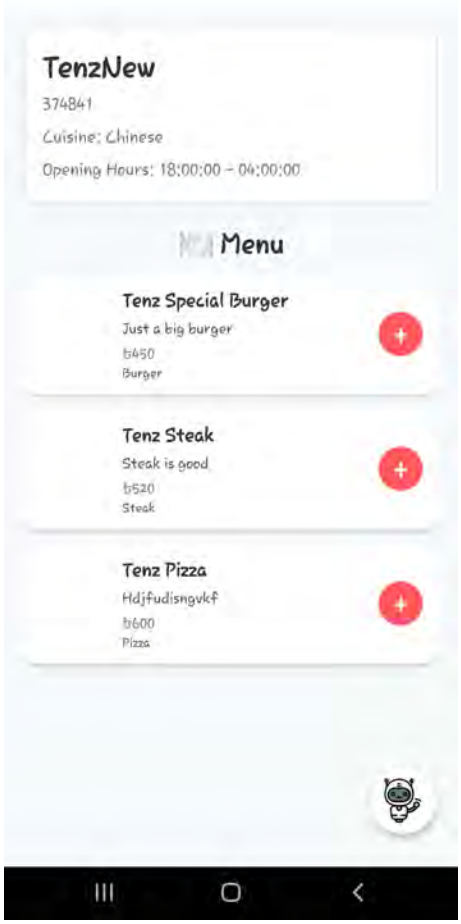

Figure 9.6: Menu of a Restaurant

The RestaurantMenu component lets users see a specific restaurant's menu and put items in their cart. Here's a rundown of how it works:

Getting Restaurant Info: Grabs the chosen restaurant ID from AsyncStorage and uses it to get both the restaurant details and menu items from Supabase.

Showing the Menu: The menu items it gets are shown in a FlatList, with each item's name, description, price, and category on display. If there aren't any menu items, it shows a message saying so.

Adding to Cart: Users can put items in their cart by hitting the + button. The cart is kept in AsyncStorage, and items are either added or their number goes up if

they're already in the cart.

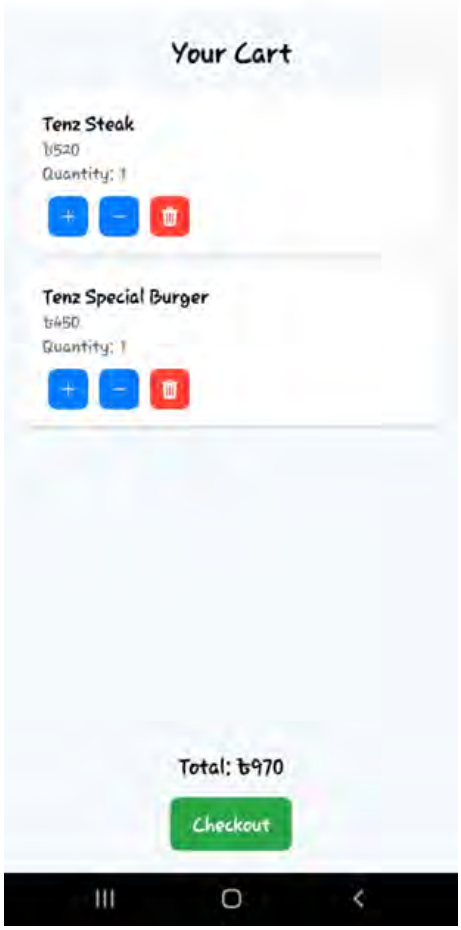

Figure 9.7: Cart

Key Features: Grabs and shows restaurant info (name, address, cuisine when it's open). Gets the menu for the restaurant you pick. Puts items in a cart and keeps them in AsyncStorage. Shows error messages if it can't get the data.
The Kart component displays cart items and allows users to change quantities, delete items in their cart, and checkout. Features Include:

Cart Items: It fetches items from AsyncStorage, displaying their names, prices, and quantities. Users can modify or delete items.

Quantity and Removal: Users will be able to add to or subtract from the quantity of items in their carts, or remove an item altogether.

Checkout: Takes the order through Supabase and completely resets the cart in AsyncStorage after a successful checkout.

Total: Calculates the complete account of items in the cart and makes it visi-

ble.

UI: Clean, card-style layout with action buttons and ionicons for quantity adjustment and removal.

Checkout Button at the bottom to finalize the purchase. Once the checkout button is pressed,it gives the user two options.

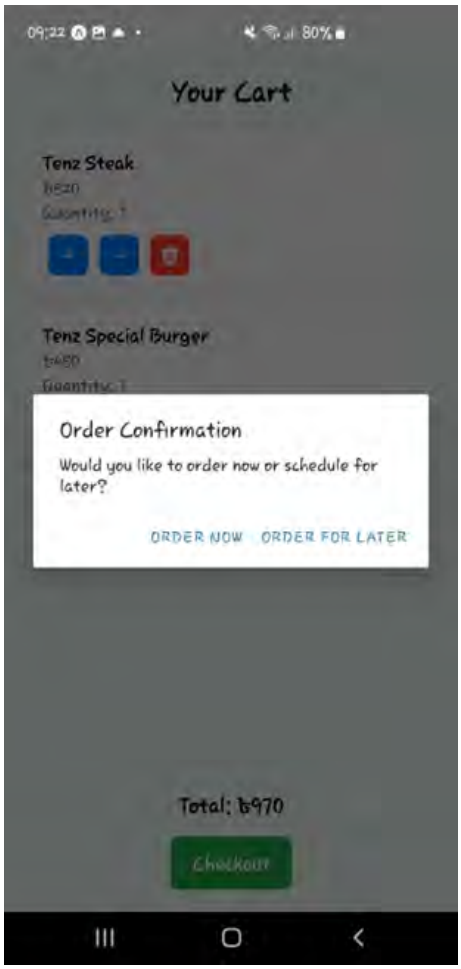

Figure 9.8: Schedule Ordering

Order now: it stores the user id, restaurant id, restaurants menu id, with quantity and additional information in order table.
Order for later: it does the same thing together with the desired order time and date of the users on scheduling table. After time and date is selected, the user is redirected to bkash payment system.

LLM In Flavourful

The frontend of react native of the and backend python enables interaction with a Large Language Model (LLM) called **Llama 3**, hosted on **RunPod**, using asynchronous programming techniques for efficient communication.

Coordinating LLM Requests (`_call_llama_api`)

At the core of LLM interaction is the `_call_llama_api` function, which:

- Sends an HTTP POST request to `RUNPOD_LLAMA_ENDPOINT_URL` with:
  - The prompt (input text)
  - Inference parameters like:
    - `temperature` (controls randomness)
    - `max_new_tokens` (limits output length)
    - `top_p` (sampling strategy)
    - `repetition_penalty` (discourages repetition)
- Receives a `job_id` and status (e.g., `IN_QUEUE`)
- Enters a polling loop to check job status via `RUNPOD_STATUS_BASE_URL`
- Waits until the job status is `COMPLETED` or `FAILED`
- Returns the generated tokens (LLM output)

Structuring LLM Input (`_format_llama_messages`)

The `_format_llama_messages` function formats the conversation history and user message into a structured prompt for Llama 3 Instruct Chat. It uses role tokens like:

- `<|system|>`
- `<|user|>`
- `<|assistant|>`

These help the LLM identify role boundaries and preserve dialogue context.

Schema-Conscious Generation (e.g., `classify_intent`, `extract_order_details`)

For structured tasks, the system uses **schema-guided generation**, similar to function calling:

- A JSON schema is embedded in the prompt (e.g., `CLASSIFY_INTENT_SCHEMA`)
- Detailed examples are included to demonstrate correct output structure
- A low temperature (e.g., `0.01`) ensures deterministic, parseable responses
- The `_parse_json_from_llama_response` function extracts JSON, even when markdown (e.g., ```` ```json ````) or extra text is present

Generating Dialogic Responses

Several functions generate natural language replies using the LLM:

- `generate_response`
- `generate_confirmation_or_next_step`
- `answer_info_query`
- `browse_menu_response`

They include:

- A `system_prompt` describing the restaurant (`restaurant_name`, `address`, `opening_time`, `menu_data`)
- The `conversation_history` for coherence
- A higher `temperature` (e.g., 0.7) to encourage creative and conversational responses

The Backend of LLM

Every conversation with the chatbot requires user inputs and chatbot responses to be stored. This is known as **state management**.
`ChatSessionState` serves as a log for recording conversation information. It tracks:

- The **user ID**
- The **restaurant ID**
- The list of selected food items (`current_order`)
- The user's most recent intent (`last_intent`)

Most importantly, it stores the entire `conversation_history`. This is essential because advanced language models (LLMs) require previous conversation turns to maintain context and relevance in ongoing dialogue.

Figuring Out What You Want: `process_chat_message`

When a new user message arrives, it is handled by the main function `process_chat_message`. The function:

1. Sends the user message and conversation history to the LLM.
2. The LLM performs **Intent Classification**, deciding what the user wants (e.g., to place an order, ask for info, confirm an order).
3. The intent is returned as structured **JSON output** based on a predefined schema, which makes it easier for the backend to interpret.

Rationale Based on Purpose: Responding to the Inquiry

Once the intent is known, the chatbot behaves like a smart manager:

- **Start an Order**: The system sends the user's message and menu data to the LLM for **Entity Extraction** (e.g., items, quantities like "2 pizzas, 1 coke"). It matches those with menu items and adds them to the active order.
- **Confirm an Order**: It compiles all order items, calculates the total, assigns a unique order ID (using `uuid.uuid4()`), stores the order in a mock database (`MOCK_ORDERS_DB`), and resets the current order.
- **Cancel an Order**: Simply clears the `current_order`.
- **Get Info or Browse Menu**: Asks the LLM to respond based on `restaurant_data` or `menu_data`.
- **General Greeting or Confusion**: If the user message is casual or unclear, the LLM generates a friendly fallback response.

Keeping the Memory Updated

After every interaction, both the user's message and the bot's reply are appended to the `conversation_history` in `ChatSessionState`. This is critical for maintaining long-term context across messages.

How LLM and Backend Logic Work togather

The Backend

The Python code in question (`process_chat_message`, `ChatSessionState`) acts as the orchestrator layer. It serves as the cognitive foundation that oversees business logic and organizes the system in a coherent manner.

Memory Retention (State Management)

Like a manager remembering your unique requests, the backend leverages `ChatSessionState` to:

- Track conversation history
- Record previous orders
- Maintain the user's last intent

This feature is crucial, as Large Language Models (LLMs) lack inherent memory. The backend provides the contextual backdrop for continuity.

Understanding Objectives (Initial Intent Identification)

When a message is received, the backend's first task is to determine the user's overall goal. It sends the message to the LLM and asks, "What does this client want to achieve?"
The LLM returns the identified intent (e.g., `place order`, `make inquiry`) in structured JSON format, simplifying interpretation.

Entity Extraction

Entity extraction refers to collecting specific data points. For food ordering, the backend sends the message and the menu to the LLM, which extracts:

- Food item(s)
- Quantity
- Size
- Modifiers or special requests

Management of Business Rules

The backend validates the extracted information by applying rules:

- Is the food item part of the menu?
- What is the price?
- What is the total cost?

Valid items are added to the `current_order`, and `total_price` is calculated.

Processing Transactions

When the order is complete:

- A unique `order_id` is generated
- The order is stored in a mock database (`MOCK_ORDERS_DB`)
- The temporary order is cleared

Identification of Next Steps (Dialogue Management)

The backend determines the appropriate response based on the:

- Current state of the conversation
- User's intent

It then instructs the LLM on how to continue the dialogue.

Interacting with the LLM

The backend acts as a middleman:

- Formats questions for the LLM
- Interprets the LLM's responses

The Large Language Model (LLM)

LLMs such as LLaMA 3 (running on RunPod via `llm_utils`) are powerful platforms for Natural Language Processing (NLP).

Natural Language Understanding (NLU)

Upon receiving input from the backend, the LLM:

- Understands idiomatic expressions and context
- Performs intent classification
- Executes entity extraction

Natural Language Generation (NLG)

After the backend has processed business logic, it may ask the LLM to generate a natural, human-like response. The LLM is responsible for:

- Maintaining conversational flow
- Summarizing order or menu details
- Answering general restaurant questions
- Generating polite confirmations

Collaborative Flow: Backend and LLM

1. **User**: "I want to order two large pizzas."
2. **Backend**: Receives message.
3. **Backend to LLM**: "Identify the intent from this message."
4. **LLM**: Returns intent = `start_order`.
5. **Backend**: Sends the same message and menu data to the LLM for entity extraction.
6. **LLM**: Extracts item = `pizza`, quantity = `2`, size = `large`.
7. **Backend**: Validates extracted items against the menu and updates `ChatSessionState`.
8. **Backend to LLM**: "Generate a polite confirmation and ask if the user needs anything else."
9. **LLM**: Returns reply = "Great! I've added 2 large pizzas to your order. Anything else?"
10. **Backend**: Adds both the user message and LLM reply to `conversation_history` and sends the reply back to the user.

Once the user confirms the order, the order information are saved in the orders table. The chatbot can not handle schedule order as for it to happen, the user needs to pay half of the total ammount

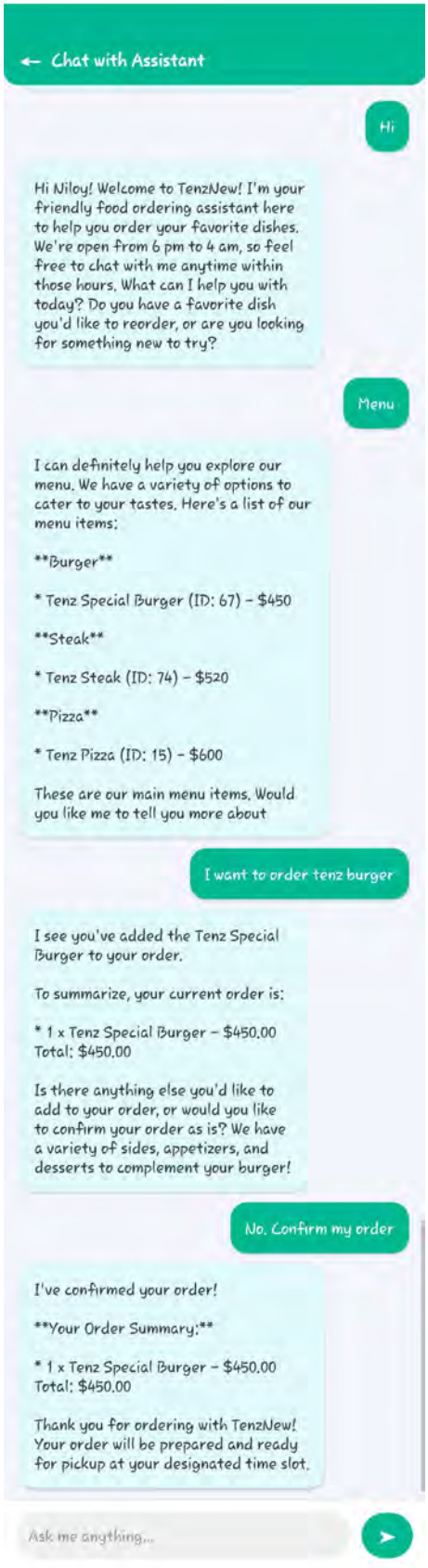

Figure 9.9: Chatbot Using LLM

9.1.1 Admin Point of View

Figure 9.10: Login of Admin

Purpose: A login page where the user can authenticate against the Supabase database using their email and password.
Key Features: Input for Email and Password: Fill in the fields used for the email and password.
Password Visibility Toggle: Button to toggle password visibility.
Login Validation: Checks whether the email exists and whether the password is correct in the admins-table.
Loading State: A loading spinner while logging in.
Error Handling: Alerts for missing fields, invalid credentials, or database error issues.
Key Code: supabase.from('admins-table') to pull data from your database for email and password validation. router.push to redirect to the dashboard page when a valid login is made.

Figure 9.11: Dashboard of Admin

Navigation Buttons: Add Restaurant, User Management, Restaurant Management **Navigation:** Uses router.push() to navigate between admin pages.

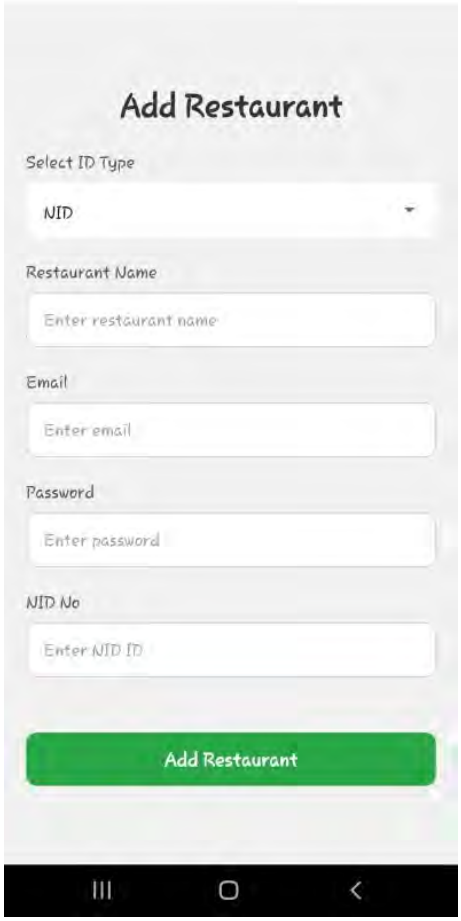

Figure 9.12: Adding Restaurant

This React Native component accepts the user input regarding restaurant name, email, password, and an ID (NID or TIN) to create an account for the restaurant. It employs the state hook useState to manage the form inputs and the routing hook useRouter from the expo-router to manage navigation. After the form submission, all the fields are validated, the email is transformed into an appropriate case-sensitive format, and, finally, the password is hashed using CryptoJS.SHA256. The component checks if the email already exists in Supabase's registered-res table, and if it exists, error message appears to notify the user. If the email does not exist, then restaurant details are inserted into registered-res, and without any password field, an entry is created in restaurant-profiles. To avoid multiple submissions of the form, a loading state is provided, which then triggers a success message before redirecting the user towards their dashboard. The UI incorporates the KeyboardAwareScrollView to handle keyboard interactions, text inputs for fields, and a dropdown to select the type of ID, and the submit button will show a loading indicator while processing. StyleSheet gives the application a stylish, modern, and friendly interface: rounded input fields with suitable spacing and a green submit button.

Figure 9.13: Users List

The component uses Supabase as a backend to manage a list of restaurants in React Native. It initializes state variables for restaurant data, filtered search results, status of loading, and search text. On mounting, it fetches restaurant data from the registered-res table in Supabase while handling errors and updating the state accordingly. The handleSearch function filters restaurants by ID, name, or email whenever a user types in the search bar. The handleDelete function confirms with the user whether to delete a restaurant from the database, and update the state to reflect that change. If data is not loading yet, it shows an activity indicator. Once it loads, the FlatList renders the restaurants in styled cards with the option to delete them. The component is built using StyleSheet for a clean and user-friendly interface.

9.1.2 Restaurants Point of View

Figure 9.14: Restaurant Login

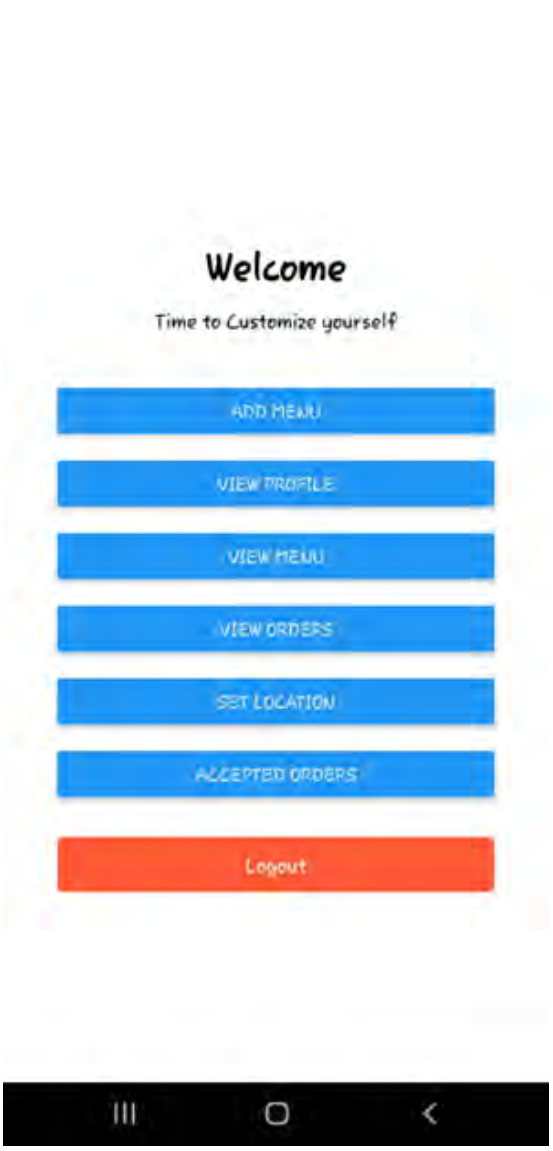

Figure 9.15: Restaurant Dashboard

Figure 9.16: View Profile

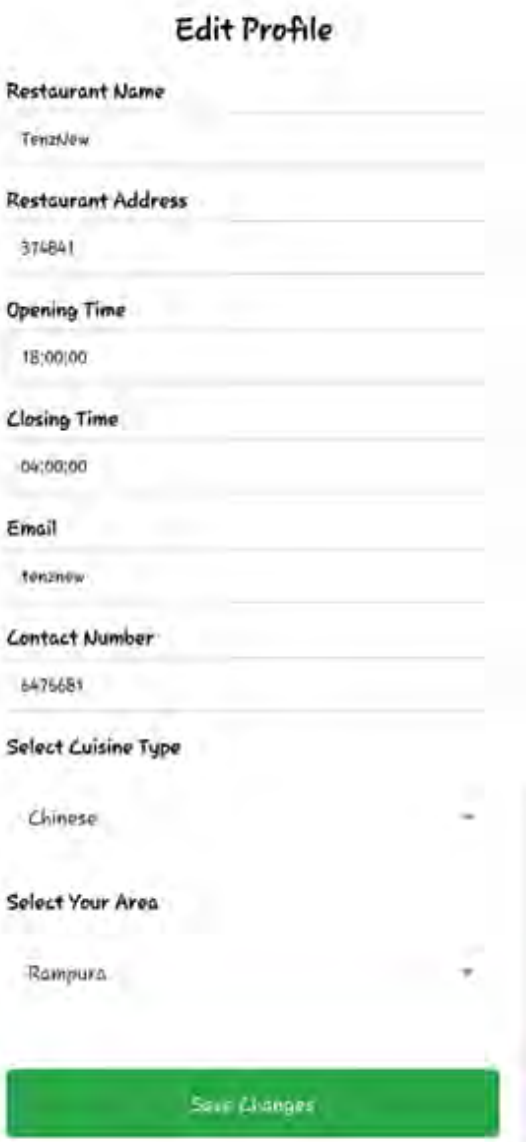

Figure 9.17: Edit Profile

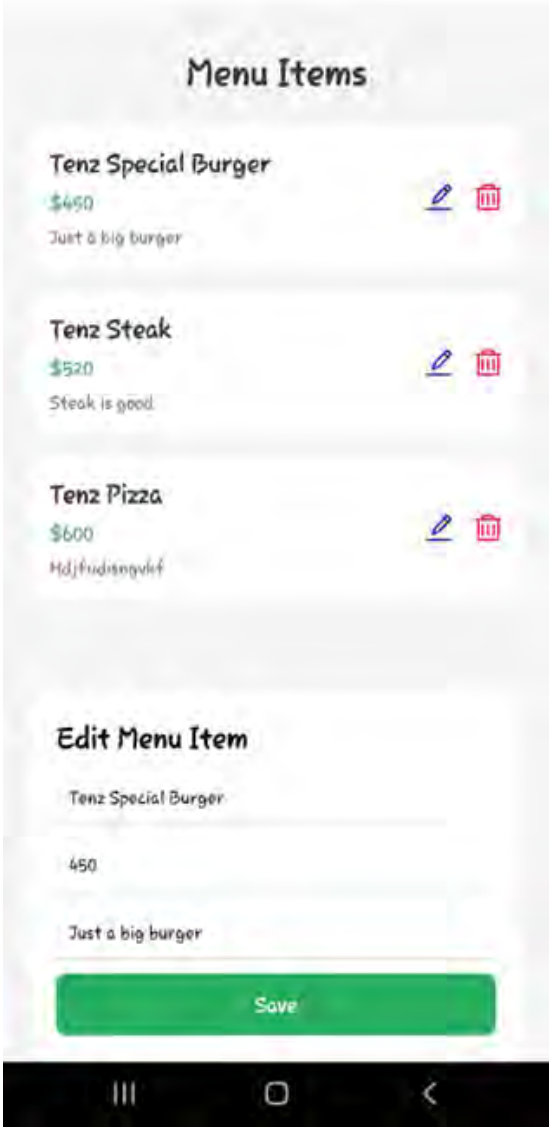

Figure 9.18: View and Edit Menu

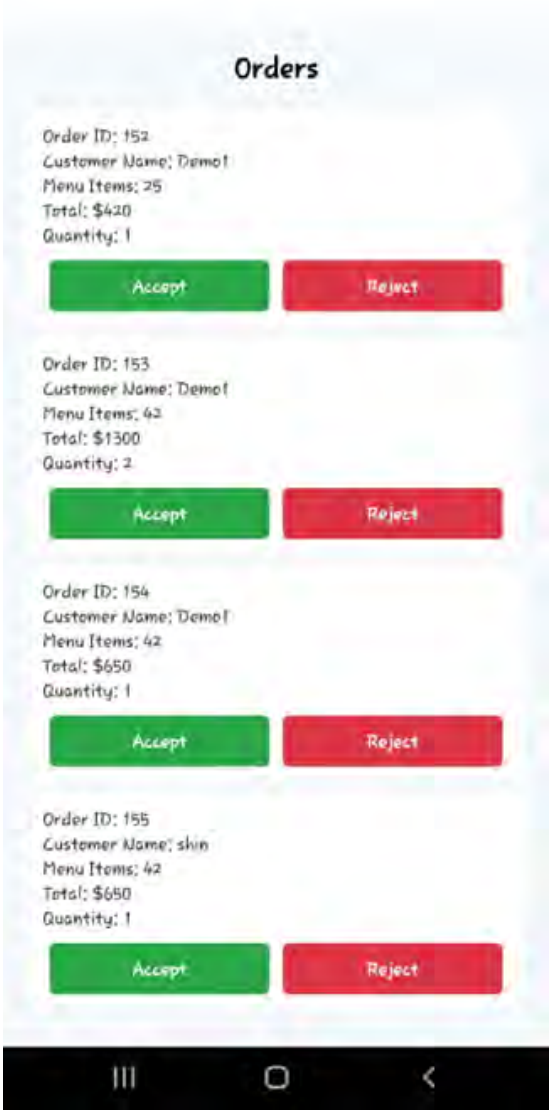

Figure 9.19: Orders

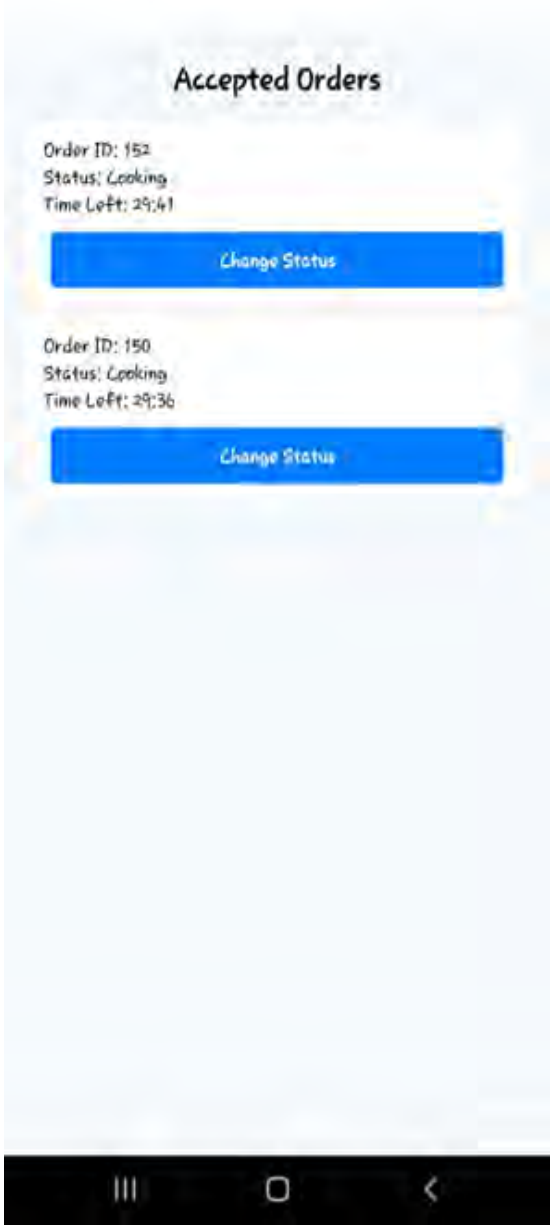

Figure 9.20: Accepted Orders

Chapter 10

Conclusion

The emergence of meal ordering and delivery companies has come from developments in technology and shifts in consumer tastes that have taken place over a decade. In other words, with the rapid changes in technology, it is smartphone apps that make it possible to order your favorite dishes at ease and enjoy them at any place you like. Everyone can now have access to their favorite food without leaving their own house, just by making a few taps on their smartphone. But with potential come obstacles, ranging from technological and operational limits to economic and societal consequences. App crashes, low user engagement, expensive commissions, and regulatory challenges are among the industry's significant concerns, which must be addressed to ensure long-term success. Besides, the competitive scene is changing very fast with major companies like Uber Eats, DoorDash, Grubhub, etc., coming to dominate the industry doing all things possible including creating a monopoly to increase costs on both restaurants and consumers even when it increases market share. However, a lot of positive trends and solutions lie ahead, such as artificial intelligence adoption, cloud kitchen growth, an emphasis on sustainability, etc., which would help restaurants and delivery applications become more efficient in providing a better user experience and even reduce some of the harmful effects caused by them. Ultimately, innovation, sustainability, and customer satisfaction must be the prime focus of any food delivery business if it wishes to sustain itself in this ever-changing landscape. Addressing the issues and adopting the upcoming trends, the food ordering and delivery industry can surely move ahead to meet the changing consumer needs in this digital era.