Jaypee Institute of Information Technology

Minor Project Report

Application name - <u>Annapurna</u>

A canteen food ordering system application

Superviser - Dr. Prakash Kumar

Index

1. Acknowledgment
2. Abstract
3. Introduction
4. Literature Review
5. Requirements and specification
6. Design algorithm and Flowchart
7. Dependencies
    7.1 Flutter and Dart
    7.2 Firebase Firestore

8. MVVM Model

9. Implementation

9.1 User authentication

10.Result screenshots

11. Contribution of each

member 12.Future Work

13.Conclusion

14.References

1. Acknowledgment

The project was supported by the Department of Computer Science Engineering, Jaypee Institute of Information Technology. We would like to express our gratitude to our honorable mentor, Dr. Prakash Kumar (Associate Professor & Associate Dean of Students), JIIT for their advice and valuable comments regarding our project, which led us to the completion of the project. We also want to thank our classmates and seniors who helped us and encouraged us during the project.

2. Abstract

The proposed system will be a complete full-fledged app-based canteen management system which will be very feasible for the students, and the faculties to order food. On the canteen side, the staff will be able to view all the purchases from different users and analyze these purchases from the app through a tab. The app will be secured with login id- password during registration for the different users. We are planning to incorporate a payment gateway, so that there is no need for the user to carry money in hand. There will be an option to make online payments via debit/credit card, internet banking and other online services like PayTM and Gpay. If the user doesn't have an account for that particular payment gateway they can order food online and make payment at the counter in cash. This proposed system will help in removal of long queues in the busy hours of the restaurant and also promote social distancing protocol in these tough times. No contact delivery will be another benefit provided by the same.

3. Introduction

In a college canteen, there can be at the most two counters available, so in the lunchtime or the peak hour of the canteen the crowd gets bigger and students need to wait for a long time in the queue to place their order and at the same time it gets difficult for the canteen side to manage and sometimes it ends up with some error in the payment and exchanging of orders. A lot of chaos is generated in the kitchen too when the order is being told to prepare. So to make the work easy we decided to make an application in which the users need to register to make an account and then they can order their food through the app from anywhere within the campus and pay through the app or can also pay with cash.

From the canteen side, the user will be kept updated on their order status and when he/she gets a notification that their order is prepared, then students go to collect their order because of this student's no need to wait in a long queue to collect their order. A history of all the transactions will be maintained for the student as well as the canteen on their respective accounts.

4. Literature Review

In [1] they had proposed an android application system that would let the user order food from their house via their personal mobile phones so that they don't have to stand in long queues in the canteen to order food.

In [2] they worked on an android application system where they used Ewallet for the payment of food. They used various security algorithms to make the wallet secure. In our work we have referred to their work in terms of canteen management features and we thought of integrating a payment gateway, the feature of cash on delivery.

In [3] proposed a system for ordering food in restaurants. Here they stated that every table would have a tab through which the customers would order the food, but making tab available at each table becomes expensive for the restaurant. So, in our work instead of keeping tabs at each table in the canteen the application can be downloaded in android mobile phones.

In [4] they have proposed a system for the canteen management system to be used by students. Using this system the students will save their time and also have the correct change for each order while paying it through E-wallet. They have used ElGamal algorithm for the security of their wallet and used HTML, javascript, bootstrap for making the canteen management system.

In [5] this project, an online canteen system helps the users to book their food earlier. The users have to book their food on the menu card online. As soon as they book their food the order will be sent to the chef for preparing it. In the proposed system the payment is online and the e-menu will be available for the user. The users will have the username and the password through which they can book. This project helps in demonstrating the route from adapting materials to developing an online environment. This brings all necessities in one place that benefits both the user and the canteen owner smartly.

5. Requirements and specifications

Framework - Flutter

Language - Dart

Database - Firebase Firestore

IDE - Visual Studio Code

Version Control - Git

6. Design algorithm and flowchart

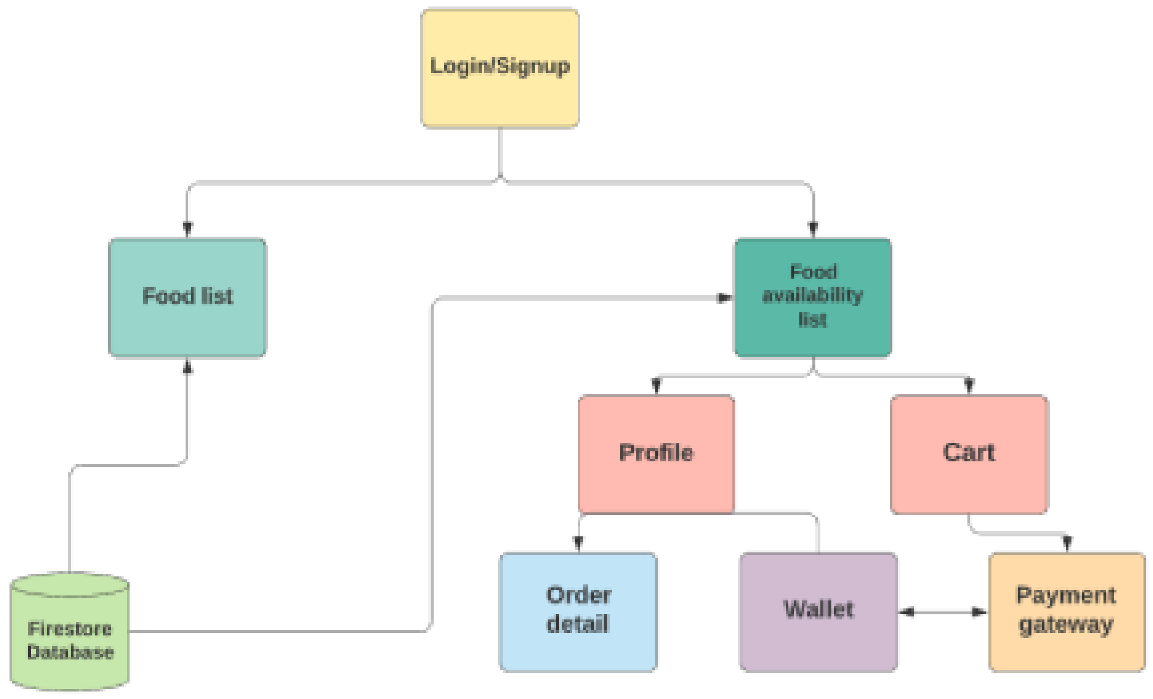


Fig. 6.1 Basic idea of the project

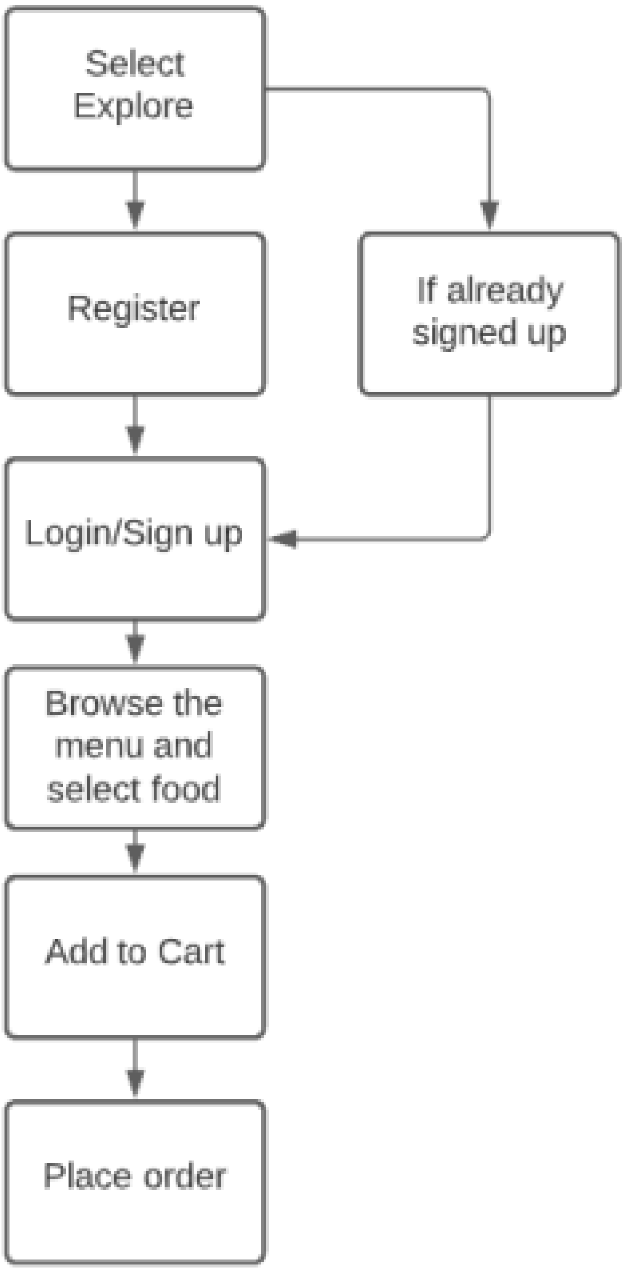


Fig. 6.2 Flowchart

7. Dependencies

7.1. Why Flutter and Dart?

Flutter is defined as the Google's UI toolkit for building beautiful, natively compiled applications for mobile (Android, iOS) desktop (Linux, Mac, Windows, Google Fuchsia) and the web from a single codebase by using a modern reactive framework.

Flutter apps are built using Dart, a simple object-oriented programming language. The central idea of Flutter revolves around widgets. The entire UI is made of combining different widgets, each of which defines a structural element (like a button or menu), a stylistic element (like a font or color scheme), an aspect of layout (like padding), and so on. Flutter does not use OEM widgets, but provides its own ready-made widgets which look native either to Android (Material Design) or iOS apps (Cupertino). It's also possible to create custom widgets.

High Productivity - Since it is cross platform, one can use the same codebase for iOS and Android applications.

Great performance - Dart converts into native code and there is no need to access OEM widgets as Flutter has its own.

Fast and simple development - Its fast reload allows us to instantly view changes made in the code on emulators, simulators and hardware. In less than a second, the changed code is reloaded without the need of restart.

Compatibility - Since widgets are part of the app, there are less or no compatibility issues on different OS platforms.This means that less time is spent on testing.

Open Source - Both Flutter and Dart are open source and free to use, and provide excellent and extensive documentation and community support to help with any issue that the user may encounter.

7.2. Firebase Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through real time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.

Benefits of using firestore -

- Flexible
- Expressive querying
- Realtime updates
- Offline Support
- Designed to scale

Implementation steps -

1. Integrate the Cloud and Firestore SDKs
2. Secure your data
3. Add data
4. Get data

Debug Mode: This mode enables debugging of apps on a physical device, emulator, or simulator. Assertions and service extensions are enabled here. Quick deployment is then achieved by optimizing compilation.

8. MVVM Model

The application we designed is based on the MVVM architecture.

Model–View–ViewModel (MVVM) is a very established architectural pattern when it's come to software development.

It's the architecture to be placed in the application which communicates between UI and business logic.

ViewModel is the mediator between View and Model which carry all user events and return back the result.

There are three key things that flow out of applying MVVM −

- Maintainability:- The presentation layer and the logic are loosely coupled, due to this code is easily maintainable and reusable. As the code base will increase over the course of time, this will help you to distinguish between them.
- Testability:- The ViewModel is easier to unit test than code-behind or event-driven code. Thanks to separation logic MVVM has.
- Extensibility:- This architecture gives you assurance which enables code to get extensible over the period of time. but keep in mind it's also an over job to keep components reusable.

9. Implementation

The implementation of this application has been divided into parts -

● Admin Module - This module allows the admin to login into the app and make changes to the menu (add or delete any food item, modify its price or change its availability status). ● User Module - This module allows the user to sign in as a customer, scroll and look through the menu, add items to the cart and place an order. Additionally it allows the user to keep a track of his past orders.

Directory structure:

• android - where Android-related files are stored.

- ios - where iOS-related files are stored.
- lib - this is where we worked most of the time. By default, it contains a main.dart file, this is the entry point file of the Flutter app.
- test - this is where we put the unit testing code for the app.
- pubspec.yaml - this file defines the version and build number of our app. It's also where we defined our dependencies. This file has the same job description as the package.json file so we can define the external packages (from the Dart packages website) we used in here.

The app is built by nesting widgets within each other. This means that the root of an app is a widget and everything below is a widget.

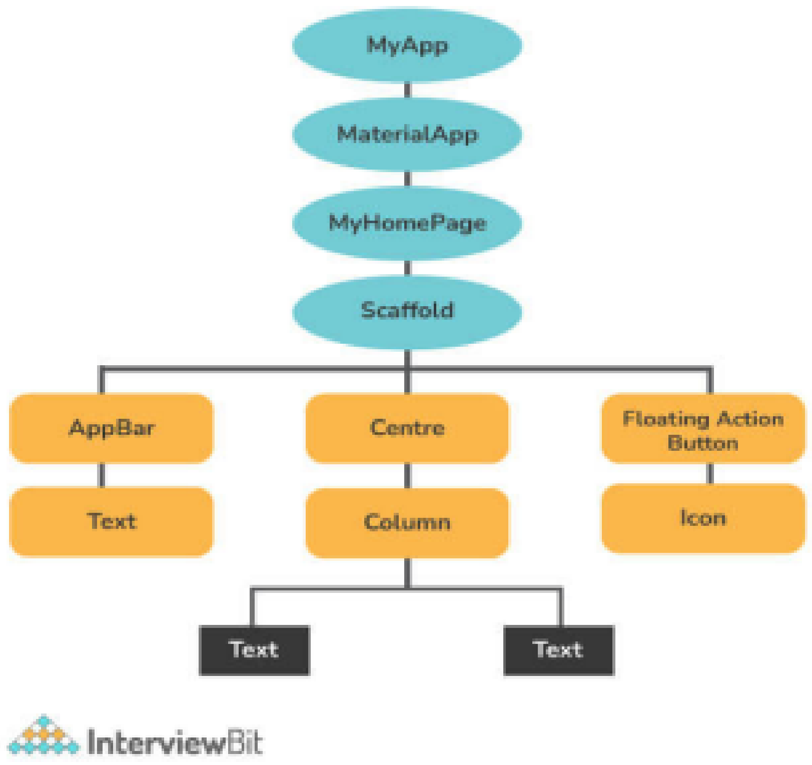


Fig. 9 Widget framework

First there is a widget for the company/organization/app name(Annapurna). Then about the screen itself, Sign in. Now, we have two text fields, username and password, to get login/sign-in credentials from the user. Then we have a Flat Button for the Forgot Password. After that, there is the Login button. If the user does not have an account, there is a provision for the Sign-up process, hence a Sign-up Flat Button.

Most of the widgets used like Text and Flat Button are surrounded in Container widgets and we have inputted the colour, padding, margin, backgrounds, according to the theme of our application.

First, we use an async widget to create the login page of the app. Then we use FirebaseAuth (which is the gateway to the Firebase authentication API) to manage user accounts and credentials. By doing this, we create the sign-in options for the users. If the email entered is not verified by FirebaseAuth, then it will return a message "*Email ID not verified*". If the email entered matches the admin account, then it will redirect the app to adminHomePage.
After the account is verified, the async widget uploads the user data to the Firebase and it prints a message saying 'user data uploaded successfully'.

A sign out button is created using async widget which redirects the user to the login page directly.

If the user accidentally forgets their password, then a ‘forgotPassword’ button is created to restore their password. Once the user clicks on that button, FirebaseAuth will send an email to reset their password. After that, the userData gets updated on Firebase.

Thereafter, after login successfully, the menu page loads up, which gets updated from the database stored in Firebase by Passing Future<QuerySnapshot> to future FirebaseFirestore.instance.collection('posts').get() .
If the data.documents.length >= 10, then, a message is displayed "Cart cannot have more than 10 times!" as the limit has been set to 10.

For the admin side, services like to edit cart items, delete cart items and add cart items are provided to provide more flexibility on the admin side. The changes reflected on the admin side reflect directly to the realtime database and continue to reflect on the user side of the application automatically.

9.1. User Authentication

Social authentication is an authentication technique in which one uses an existing account to log into or create a new account in another application. Besides levying a user on constantly remembering multiple passwords, social authentication eases the registration process and improves security.

Google provides a platform for integrating social authentication with Firebase. A programmer only needs to generate an SHA1 key and configure it with his/her machine. On the login screen,

a user is asked to login using his email id and password. The app calls a function to authorize the user and redirect him/her to a home screen where we display their name and profile image fetched from the social account selected.
The home screen has a logout button that logs the user out and takes them to the login screen once more.

Basic steps to add Firebase services to your app:

- Set up a user authentication flow with Authentication.
- Store data, like user information, with Cloud Firestore or Realtime Database.
- Trigger backend code that runs in a secure environment with Cloud Functions.
- Send notifications with Cloud Messaging.
- Find out when and why your app is crashing with Crashlytics.

We need the following dependencies for our project:

- **firebase_core**. This dependency will allow us to connect multiple Firebase apps to our Flutter project.
- **firebase_auth**. This plugin will enable us to use Firebase Authentication API.
- **google_sign_in.** To enable login with Google.

All these dependencies are added in the pubspec.yaml file.

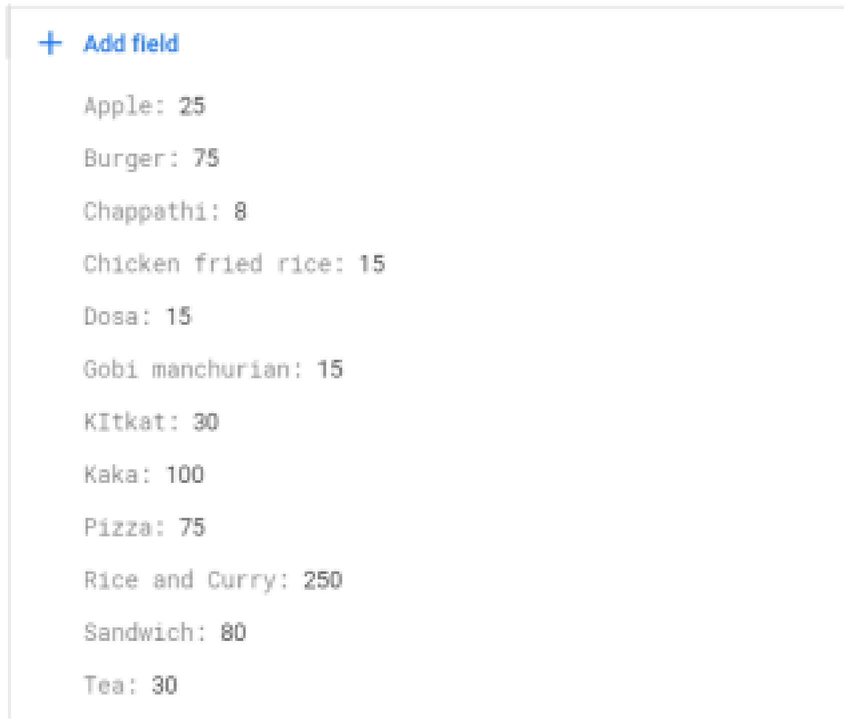
- **package:flutter/material.dart** - This import allows you to access Flutter's built-in widgets and other functionalities.
- **package:firebase_database/firebase_database.dart** - This import helps access the Firebase Realtime database features.
- **package:firebase_core/firebase_core.dart** - This dependency enables you to connect to different Firebase products.

We can verify our application's registered users. To do the verification, we must head over to the Firebase console for your project. Under authentication, clicking the users' tab, will show the list of registered users.

After adding the configuration files and dependencies to the project for the firebase_database dependency, we can add data, get data, update records and delete records whenever necessary. All these changes will occur in real time and will immediately be reflected in the application.

A data model is created to reflect the structure of your app messages. In addition to the data model, we make a Data Access Object (DAO) that can store and retrieve the messages from our Realtime Database.Add a new folder inside lib called data. We use this folder to store the data models and data access objects.

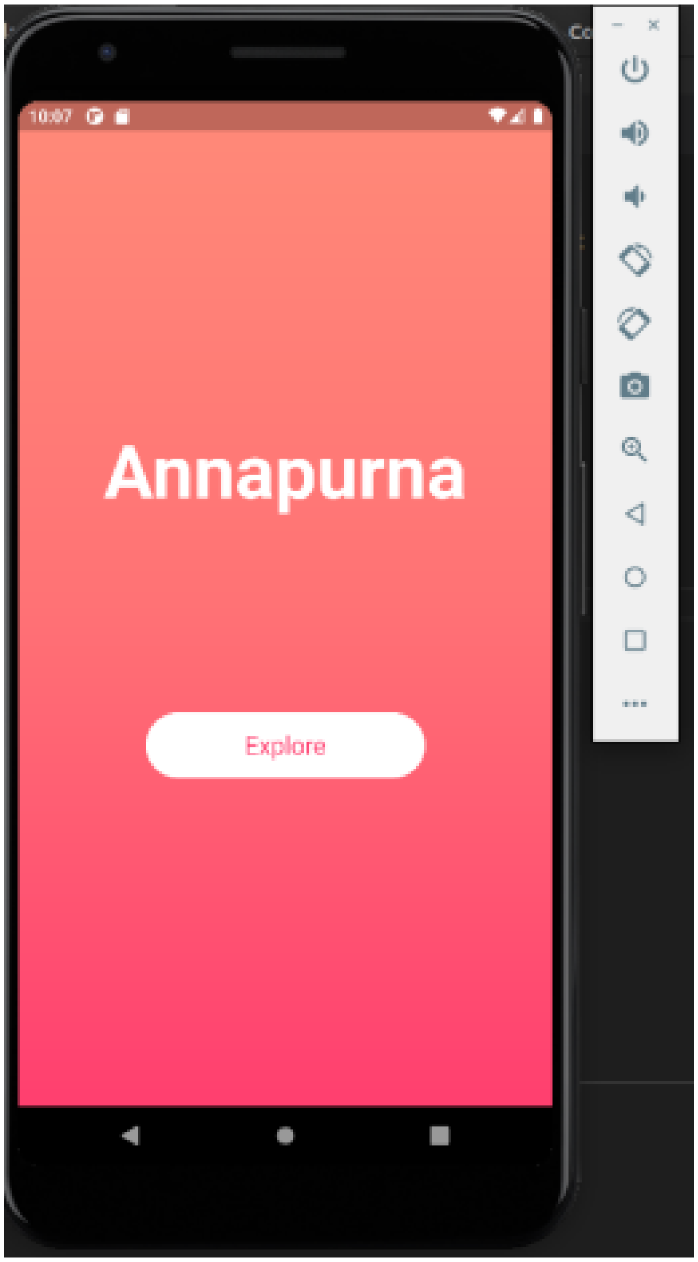
After the data is added to the database, the data looks as shown below:



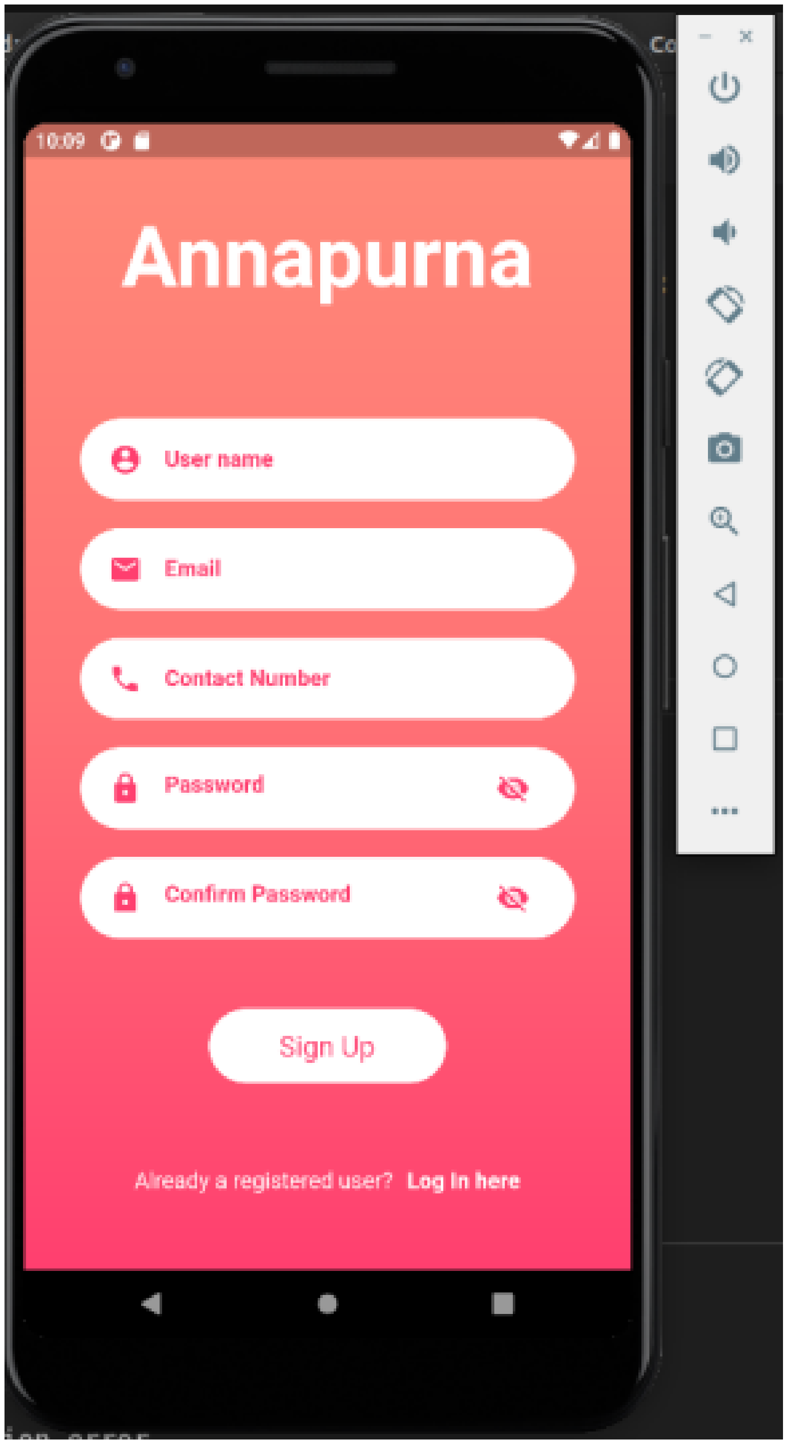
```
Apple: 25
Burger: 75
Chappathi: 8
Chicken fried rice: 15
Dosa: 15
Gobi manchurian: 15
KItkat: 30
Kaka: 100
Pizza: 75
Rice and Curry: 250
Sandwich: 80
Tea: 30
```

10. Result screenshots

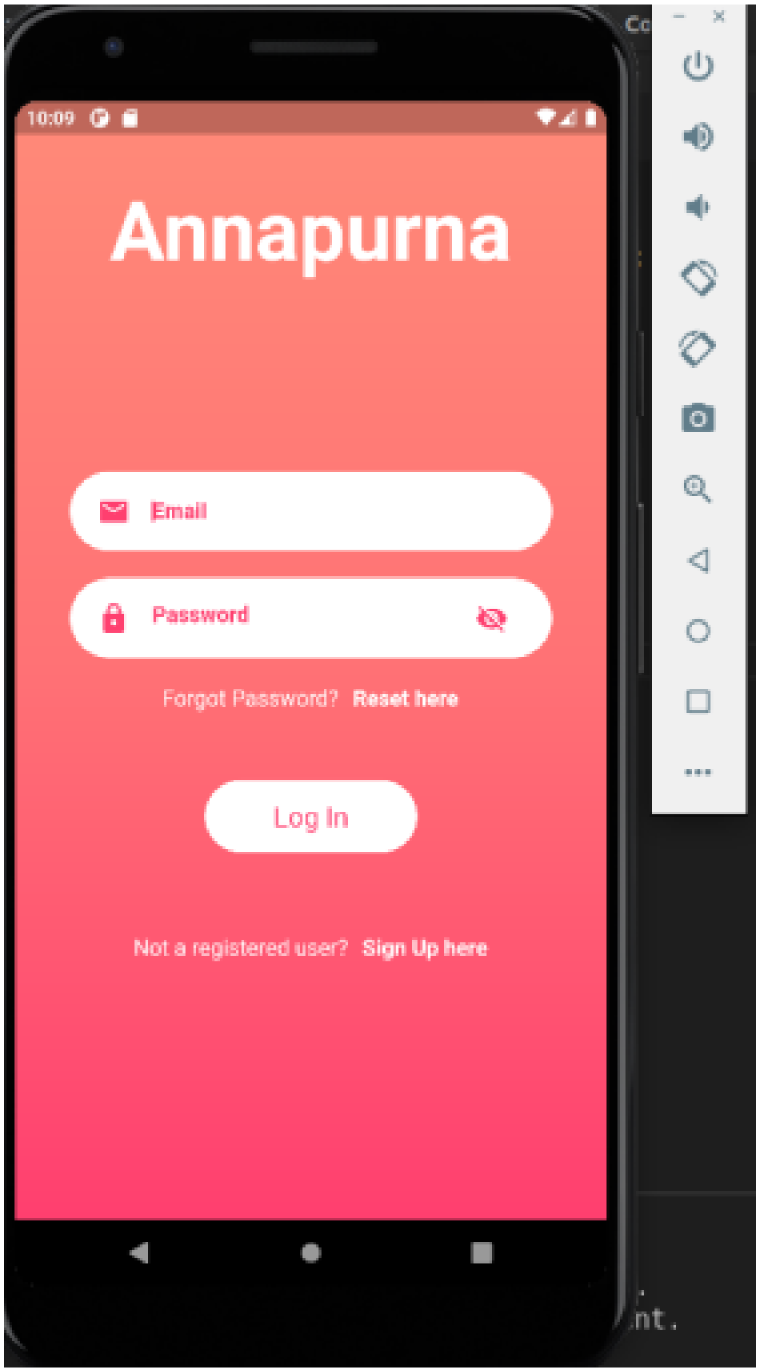
10.1. The Explore page



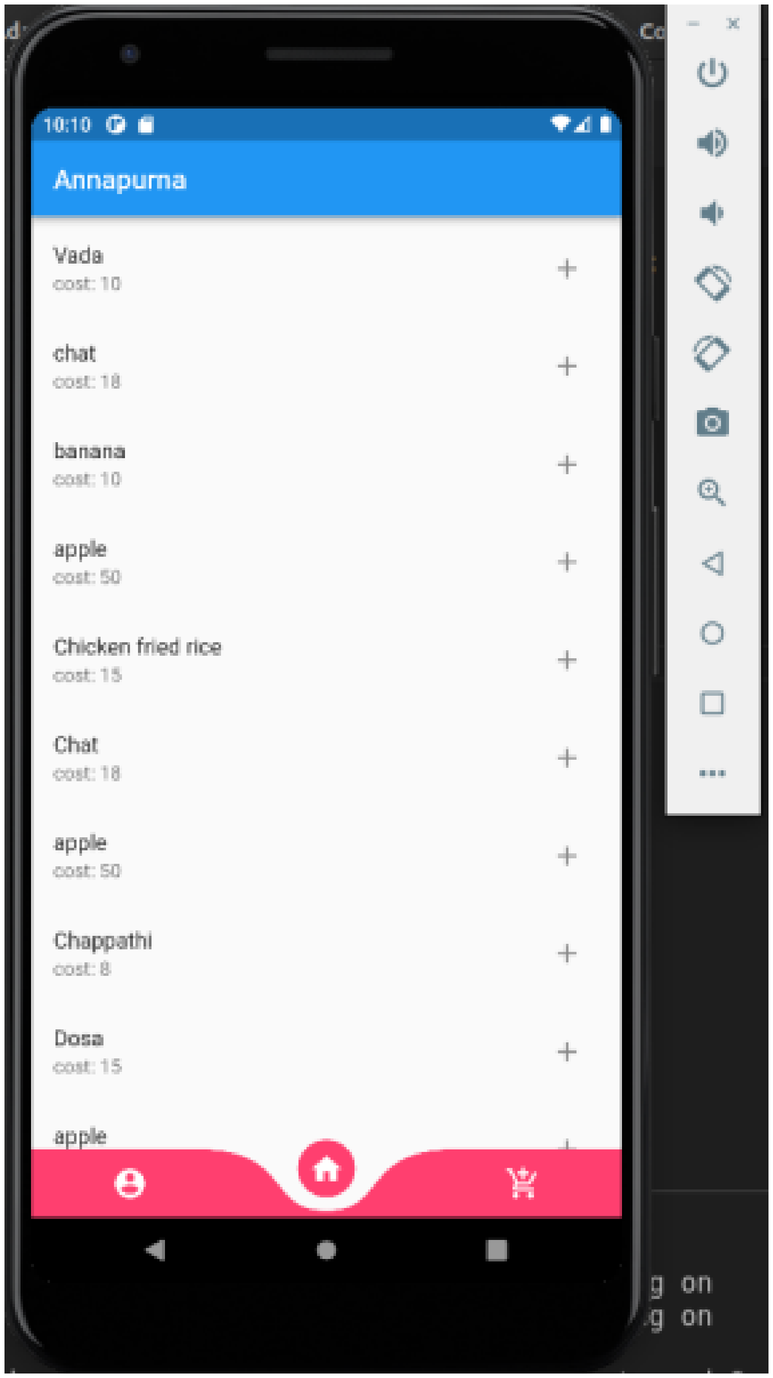
10.2. Register page



10.3. Login/Sign up


10:09
Annapurna
Email
Password
Forgot Password? Reset here
Log In
Not a registered user? Sign Up here

10.4. Food menu


10:10

Annapurna

Vada
cost: 10

chat
cost: 18

banana
cost: 10

apple
cost: 50

Chicken fried rice
cost: 15

Chat
cost: 18

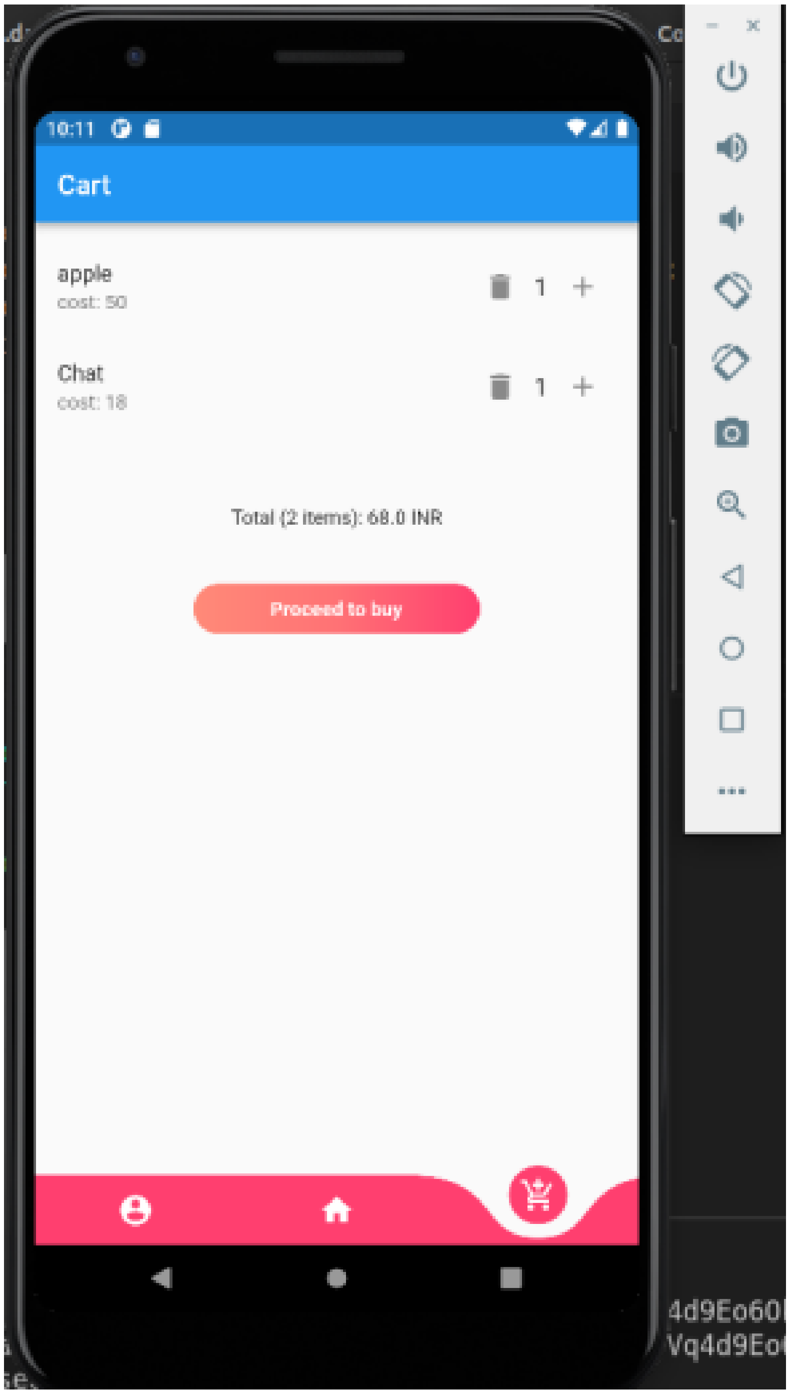
apple
cost: 50

Chappathi
cost: 8

Dosa
cost: 15

apple

10.5. Cart page



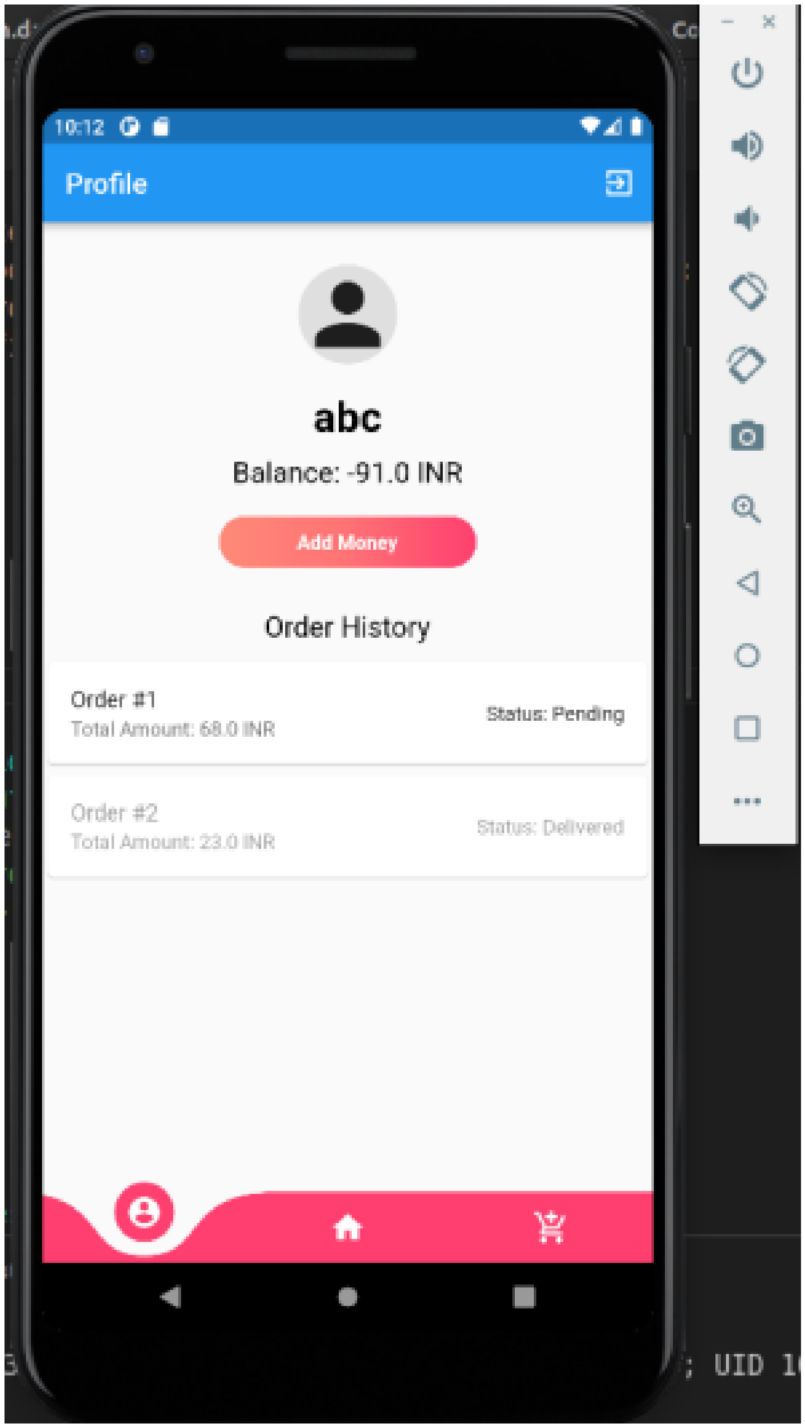
Cart

apple
cost: 50
1

Chat
cost: 18
1

Total (2 items): 68.0 INR

10.6. Profile page


Profile

abc

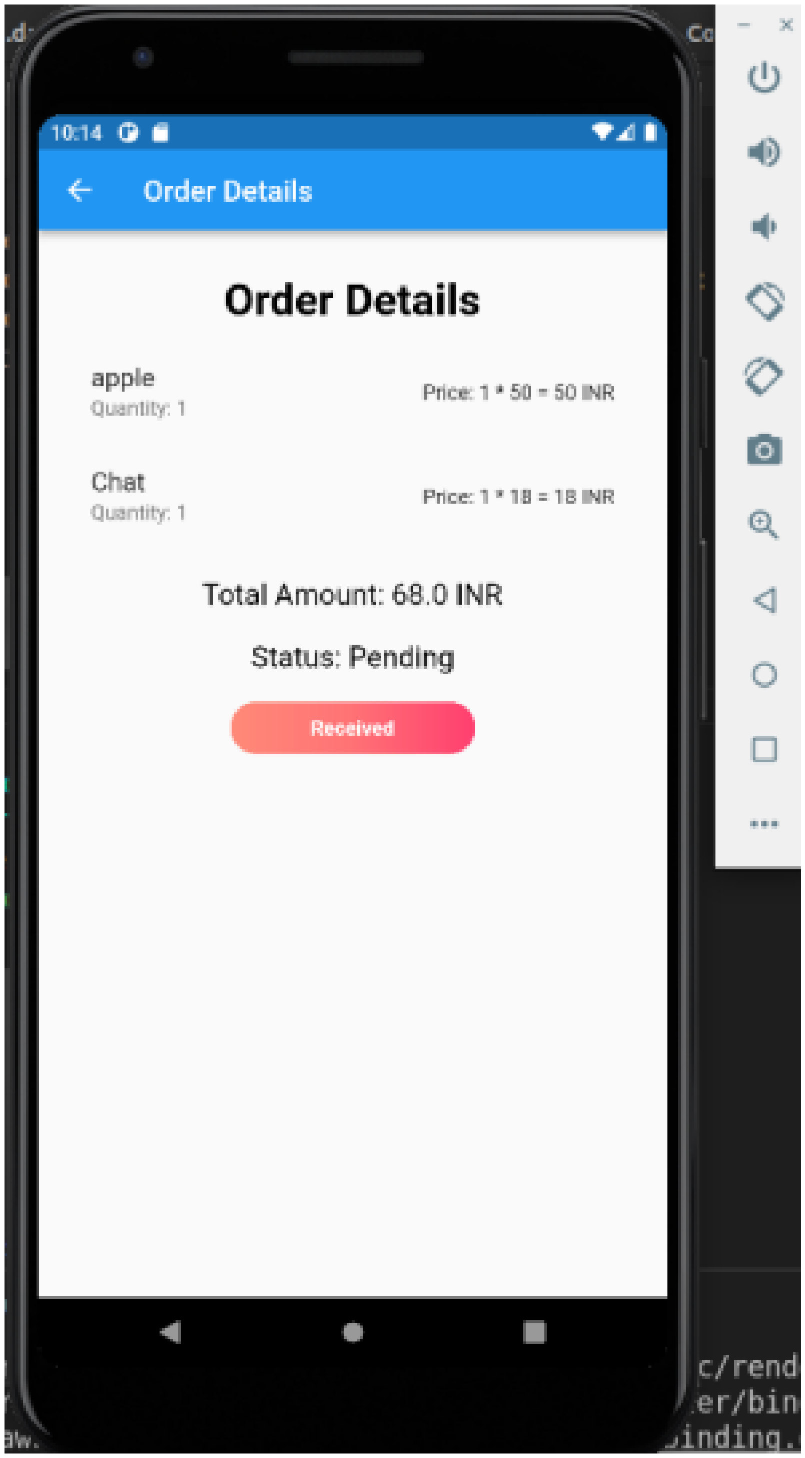
Balance: -91.0 INR

Order History

Order #1
Total Amount: 68.0 INR
Status: Pending

Order #2
Total Amount: 23.0 INR
Status: Delivered

10.7. Order details page



10:14

Order Details

Order Details

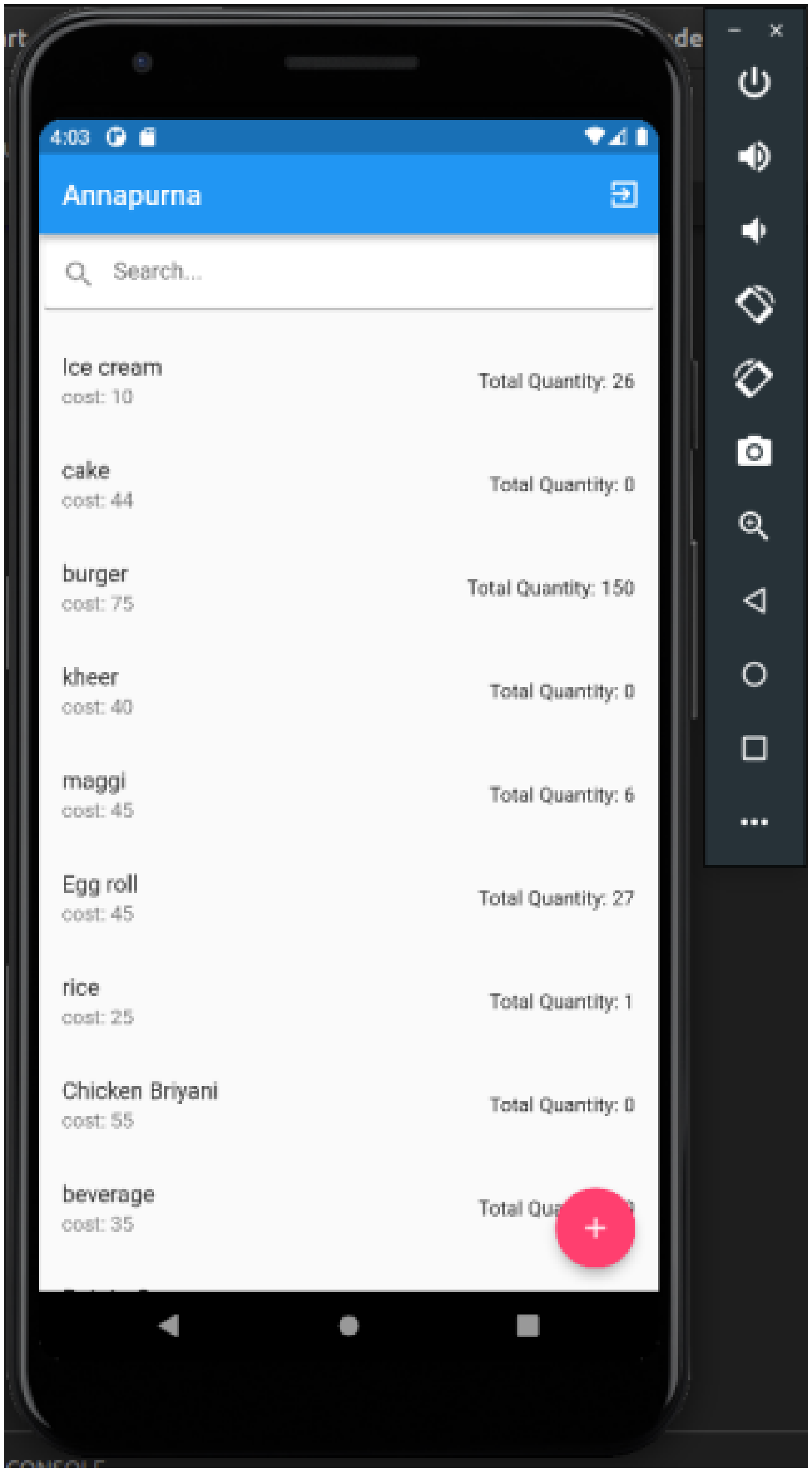
apple
Quantity: 1
Price: 1 * 50 = 50 INR

Chat
Quantity: 1
Price: 1 * 18 = 18 INR

Total Amount: 68.0 INR

Status: Pending

10.8. Cart details(Admin Module)

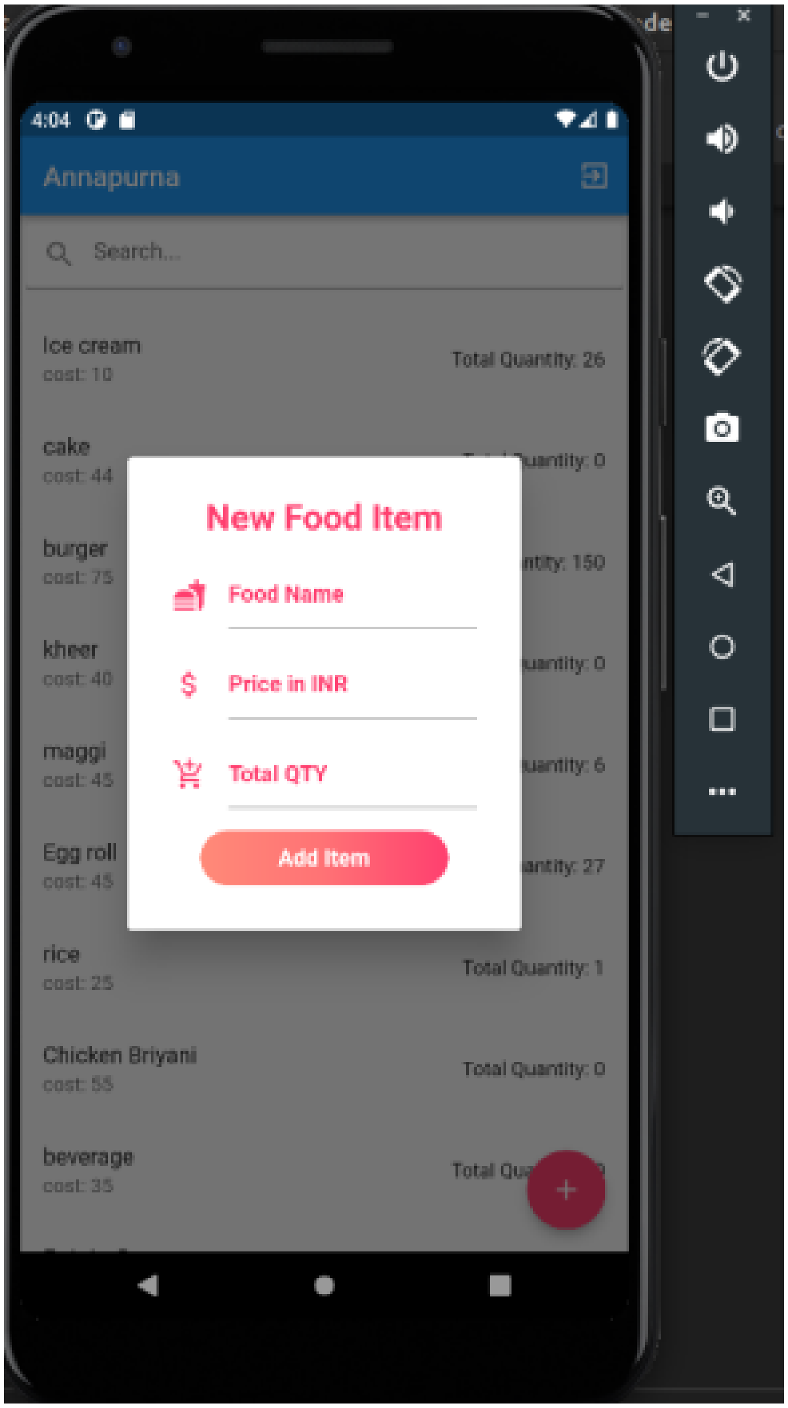

4:03

Annapurna

Search...

Ice cream cost: 10	Total Quantity: 26
cake cost: 44	Total Quantity: 0
burger cost: 75	Total Quantity: 150
kheer cost: 40	Total Quantity: 0
maggi cost: 45	Total Quantity: 6
Egg roll cost: 45	Total Quantity: 27
rice cost: 25	Total Quantity: 1
Chicken Briyani cost: 55	Total Quantity: 0
beverage cost: 35	Total Qua

10.9. Add new food item(Admin module)



4:04

Annapurna

Search...

Ice cream
cost: 10
Total Quantity: 26

cake
cost: 44
Total Quantity: 0

burger
cost: 75
Total Quantity: 150

kheer
cost: 40
Total Quantity: 0

maggi
cost: 45
Total Quantity: 6

Egg roll
cost: 45
Total Quantity: 27

rice
cost: 25
Total Quantity: 1

Chicken Briyani
cost: 55
Total Quantity: 0

beverage
cost: 35
Total Quantity: 0

New Food Item

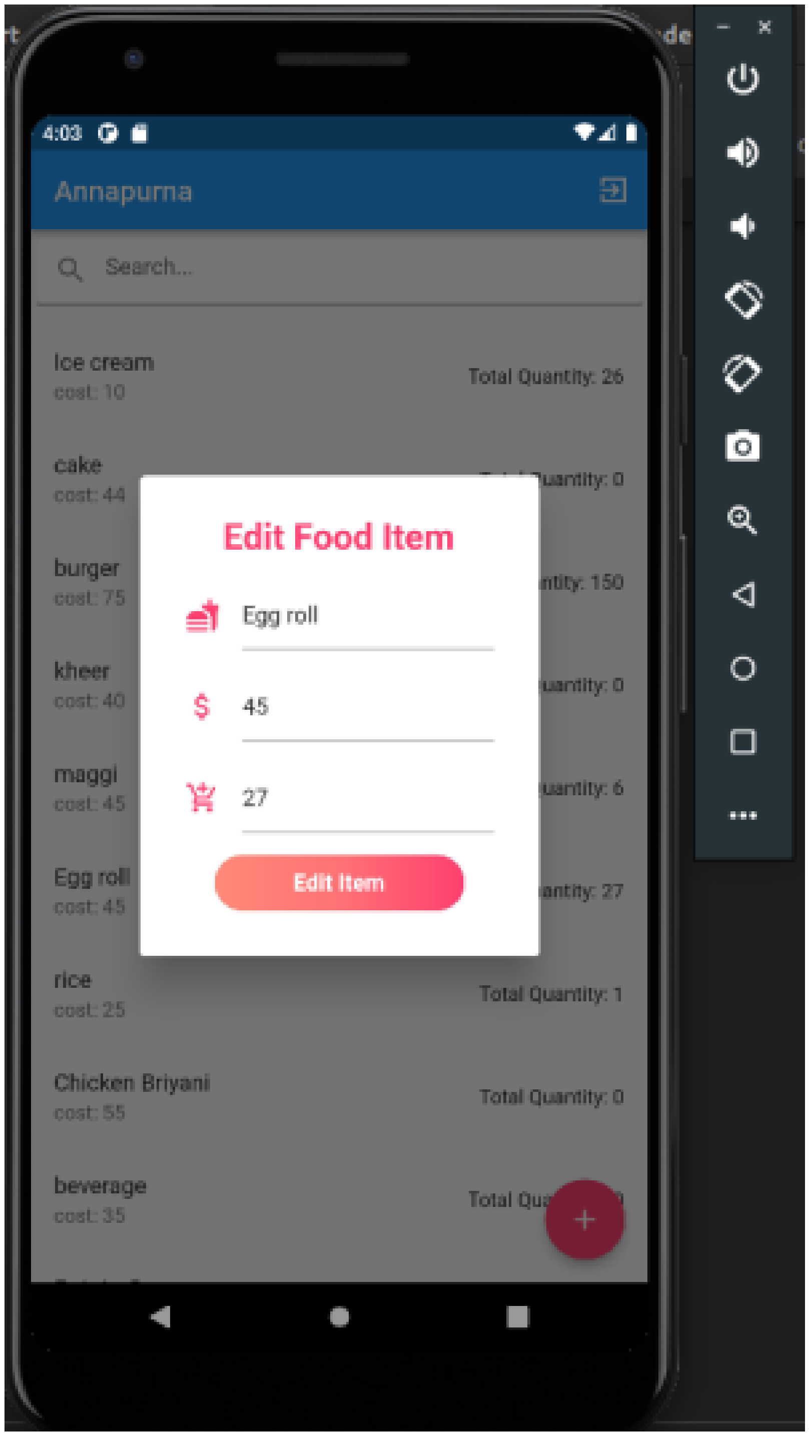
Food Name

Price in INR

Total QTY

Add Item

10.10. Edit existing food item (Admin module)



11. Future Work

1. Adding a payment gateway method to allow the customers to make online transactions in any mode, be it credit/debit card, net banking, or other online services like PayTM and Gpay. This system can be added using the Razorpay tool. Authenticating Razorpay and

giving this access to the users will save time and support the no contact delivery protocol.

2. The Canteen meal ordering mobile app was created to shorten line time and encourage e-payment to reduce cash usage. QR code generation is one of the future enhancements. Currently, the application has a "Received" button that is pressed after they deliver the food. This can be reduced by creating a QR code for each order placed that can be scanned in the Admin Application, which is equipped with a QR code scanner. The order status is marked as delivered once the code is scanned.
3. We are planning to add Facebook authentication to the application along with Google sign up authentication. This would provide more flexibility to the users and make login more easy for the users.

12. Conclusion

This automated food ordering system will allow people to place an order online.
It will promote no contact policy in the college canteen.
The online payment gateway will promote digital payment. It will also save time and make the work of a cashier easy.
The realtime database management system is flexible to maintain and any changes will be reflected in the application and thus informing the users in real time about the food availability in the canteen.
This system will provide full access to the admin and the customer side, just a matter of a few taps in the user's phone.

13. References

[1] Mittal, R., Abhishek Singh, Amit Tanwar, Aditya Sawant, Chaitanya Parulekar, Kunal Yadav –"Canteen Food Ordering Android System" International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 04 Issue: 04, 699, 2016

[2] Mittal, R., Rameshwari Fegade, Gaurav Nandge, Pranjal Patil, Tejas Gaikwad, Prof. P.P. Bastawade –" Canteen Management Android Application Using E-wallet"International Research Journal of Engineering and Technology, Volume: 06 Issue: 03, 2020

[3] Mittal, R., Anas Bin Amir, Mohd Syazwan Nazmi Bin Abul Kassim, Mohamad Ikhsan Bin Johari –"Automated Food Ordering System" School of Computer Sciences, Universiti Sains Malaysia.

[4] Mittal, R., Akash Katkar, Kalpesh Juvekar, Nitin Rohira, Smita Jangale –“Canteen management system using the Ewallet” International Journal of Advance Research, Ideas and Innovations in Technology , Volume: 04 Issue: 02, 2020

[5] Mittal, R., Ms. Minu, Kowshik Reddy, Sumanth, Ashik Teja, Gopi Kishan - “Online Canteen System”, Volume: 05 Issue: 10, 2018

[6] Accessed Date [07.10.21] URL:
https://www.section.io/engineering-education/implementing-firebase-in-flutter/

[7] Accessed Date [15.10.21] URL:
https://www.section.io/engineering-education/flutter-social-authentication/

[8] Accessed Date [17.10.21] URL:
https://firebase.flutter.dev/docs/overview
[9] Accessed Date [02.11.21] URL:
https://kodytechnolab.com/why-flutter-uses-dart#:~:text=Flutter%20uses%20Dart%20as%20Dart,it%20very%20easily%20and%20effortlessly

[11] Accessed Date [07.11.21] URL:
https://dev.to/kazuhideoki/how-to-get-data-from-firestore-and-show-it-on-flutterbuilder-or-streambuilder-e05

[12] Accessed Date [07.11.21] URL:
https://medium.com/flutterworld/flutter-mvvm-architecture-f8bed2521958

[13] Accessed Date [07.11.21] URL:
https://pusher.com/tutorials/login-ui-flutter/

[14] Accessed Date [07.11.21] URL:
https://medium.com/analytics-vidhya/login-page-design-with-flutter-28371e78f7a2

[15] Accessed Date [07.11.21] URL:

https://www.section.io/engineering-education/flutter-social-authentication/

[16] Accessed Date [07.11.21] URL:
https://blog.codemagic.io/what-is-flutter-benefits-and-limitations/#:~:text=On%20the%20official%20Flutter%20Website,web%20from%20a%20single%20codebase.&text=Since%20then%20Flutter%20has%20improved%20a%20lot%20in%20terms%20of%20performance.

[17] Accessed Date [07.11.21] URL:
https://kodytechnolab.com/why-flutter-uses-dart#:~:text=Flutter%20uses%20Dart%20as%20Dart,it%20very%20easily%20and%20effortlessly.