

**MOBILE APPLICATION FOR ORDER PLACING
PROJECT WORK I**

Submitted by
KRISHNA B
21CSR097

MANIPRABHA S
21CSR108

DANEESH M
21CSL256

*in partial fulfilment of the requirements for
the award of the degree
of*

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

MAY 2024

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KONGU ENGINEERING COLLEGE**

(Autonomous)

PERUNDURAI ERODE – 638 060

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that the Project report entitled **MOBILE APPLICATION FOR ORDER PLACING** is the bonafide record of the project work done by **KRISHNA B (Register No: 21CSR097), MANIPRABHA S (Register No: 21CSR108), DANEESH M (Register No:21CSL256)**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Science and Engineering** of Anna University Chennai during the year 2024.

SUPERVISOR

Date:

HEAD OF THE DEPARTMENT

(Signature with seal)

ABSTRACT

In an era where convenience is paramount, the restaurant industry is evolving to meet the demands of modern consumers. This project introduces a mobile application designed to streamline the ordering process and enhance the dining experience for both customers and restaurant staff. For enhanced efficiency, restaurant servers receive notifications once orders are ready for delivery to patrons. Upon completion of the dining experience, servers can easily close orders and enter the customer's mobile number for billing purposes.

The application, developed using Flutter framework, provides a user-friendly interface for customers to place orders directly from their smartphones. Orders are seamlessly transmitted to the kitchen, where chefs can view and manage incoming requests in real-time. Upon preparation completion, kitchen staff can mark orders as ready for serving.

Data management is facilitated through Firebase, ensuring secure storage and efficient retrieval of order information. Additionally, the application automates the billing process by generating invoices and sending them as SMS to the provided mobile numbers. This innovative solution not only improves operational efficiency for restaurants but also enhances customer satisfaction by offering a seamless and convenient dining experience. Through the integration of technology, the application aims to revolutionize traditional restaurant management practices and set new standards for the industry.

To ensure the success of the project, the Application development team will establish a clear project plan and will regularly communicate progress updates to the client. Quality assurance and testing will be conducted throughout the development process to ensure that the Application is fully functional and meets the client's requirements.

KONGU ENGINEERING COLLEGE**(Autonomous)****PERUNDURAI ERODE – 638 060****MAY 2024****DECLARATION**

We affirm that the Project Report titled **MOBILE APPLICATION FOR ORDER PLACING** being submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion this or any other candidate.

Date:**KRISHNA B****Reg No : 21CSR097****MANIPRABHA S****Reg No : 21CSR108****DANEESH M****Reg No : 21CSL256**

I certify that the declaration made by the above candidate is true to the best of my knowledge.

Date:**Name and Signature of the Supervisor**

ACKNOWLEDGEMENT

We express our sincere thanks and gratitude to **Thiru. A. K. ILANGO B.Com., M.B.A.,LLB.**, our beloved Correspondent and all other philanthropic trust members of Kongu Vellalar Institute of Technology Trust who have always encouraged us in the academic and co- curricular activities.

We are extremely thankful with no words of formal nature to the dynamic Principal **Dr. V. BALUSAMY, M.Tech., Ph.D.**, for providing necessary facilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department **Dr. S. MALLIGA M.E., Ph.D.**, for providing necessary facilities.

We extend our thanks to **Ms. P. KALAIVANI M.Tech.**, Assistant Professor, Computer Science Engineering, Project Coordinator for her encouragement and valuable advice that made us to carry out the project work successfully.

We extend our gratitude to our Supervisor **Ms. P. KALAIVANI M.Tech.**, Assistant Professor Computer Science Engineering, for her valuable ideas and suggestions, which have been very helpful in the project. We are grateful to all the faculty members of the Computer Science and Engineering Department, for their support.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	iii
	LIST OF FIGURES	viii
1.	INTRODUCTION	1
	1.1 EXISTING SYSTEM	1
	1.2 SYSTEM STUDY	2
	1.3 OBJECTIVE	3
	1.4 SCOPE	3
2.	GENERAL DESCRIPTION	4
	2.1 PROJECT PERSPECTIVE	4
	2.2 USER CHARACTERISTICS	4
	2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS	5
3.	REQUIREMENTS	6
	3.1 FUNCTIONAL REQUIREMENTS	6
	3.2 NON-FUNCTIONAL REQUIREMENTS	7
	3.3 DESIGN	8
4.	DETAILED DESIGN	9
	4.1 ARCHITECTURAL DESIGN	9
	4.2 INTERFACE DESIGN	11
	4.3 FLOW CHART	11
	4.4 USE CASE	12
	4.5 SEQUENCE DIAGRAM	12
	4.6 OUTPUT DESIGN	13

5.	TESTING	20
	5.1 UNIT TESTING	20
	5.2 REGRESSION TESTING	20
	5.3 VALIDATION TESTING	21
	5.4 VERIFICATION TESTING	21
	5.5 INTEGRATION TESTING	21
6.	CONCLUSION AND FUTURE SCOPE	22
	APPENDIX 1	24
	APPENDIX 2	72
	REFERENCES	74

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	BLOCK DIAGRAM	8
4.1	MENU	10
4.2	KITCHEN	10
4.3	FLOW CHART	12
4.4	USE CASE	12
4.5	SEQUENCE DIAGRAM	12
4.6	WELCOME PAGE	13
4.7	SIGNUP PAGE	14
4.8	SIGNIN PAGE	14
4.9	MENU PAGE	15
4.10	PROFILE PAGE	15
4.11	SEARCH PAGE	16
4.12	CART PAGE	16
4.13	TABLES PAGE	17
4.14	ORDER DETAILS PAGE	17
4.15	KITCHEN PAGE	18
4.16	KITCHEN DETAILS PAGE	18
4.17	BILL	19
A2.1	CLOSE ORDER	72
A2.2	FIREBASE AUTHENTICATION	72
A2.3	FIREBASE FIRESTORE	73
A2.4	FIREBASE STORAGE	73

CHAPTER 1

INTRODUCTION

Kumar Mess, a renowned hotel with a commitment to providing exceptional dining experiences to its customers, is seeking to elevate its customer experience through a digital platform. OrderEase, a bespoke mobile application, is being developed to revolutionize the way customers interact with Kumar Mess, offering an intuitive and convenient ordering experience. With the OrderEase app, waiters can browse the hotel's menu, place orders for customers, and make reservations with ease. The app aims to streamline the ordering process, increase customer satisfaction, and optimize hotel operations.

1.1 EXISTING SYSTEM

In the existing system, Kumar Mess relies primarily on traditional methods of receiving orders, such as in-person or paper-pen order taking. This approach limits the efficiency and convenience for customers, as well as the hotel's ability to handle orders effectively during peak hours. As a result, Kumar Mess may be missing opportunities to engage with customers digitally and offer them a seamless experience. The existing system lacks a digital platform that allows waiters to explore the menu and place orders conveniently from their mobile devices. Consequently, Kumar Mess is missing out on the potential to reach a wider audience, increase customer engagement, and enhance customer satisfaction.

1.2 SYSTEM STUDY

1.2.1 Understanding the Business Requirements

The first step in the system study is to grasp the business requirements of the OrderEase app. This involves understanding the goals of Kumar Mess, its target audience, and the specific features and functionalities required in the app.

1.2.2 Analyzing the Existing Systems and Processes

Next, it is essential to assess existing systems and processes within Kumar Mess, including its current ordering methods and any related software systems. This analysis identifies any limitations or opportunities for improvement that the OrderEase app can address.

1.2.3 Identifying User Needs

The system study involves identifying the needs and preferences of waiters who will use the app. This can be achieved through surveys, focus groups, or other research methods to gather insights on user expectations and challenges with the current system.

1.2.4 Developing Use Cases

Developing use cases helps illustrate how the OrderEase app will function and how customers will interact with it. This step provides a clear understanding of potential design and functionality considerations and helps address any potential issues early on.

1.3 OBJECTIVE

The primary objective of the project is to enhance Kumar Mess's customer experience by creating a user-friendly mobile application that allows waiters to browse the menu, place orders, and make reservations easily.

1.4 SCOPE

The scope of the project encompasses the creation of a highly functional and visually appealing mobile application for Kumar Mess. The app will offer customers a seamless ordering experience, including browsing the menu, placing orders and making reservations. Throughout the development process, the project team will work closely with Kumar Mess to understand their expectations and requirements. Regular updates will be provided to ensure the project aligns with the hotel's goals. The ultimate aim is to deliver an app that exceeds expectations, enhances the customer experience, and showcases Kumar Mess's commitment to quality service.

CHAPTER 2

GENERAL DESCRIPTION

2.1 PROJECT PERSPECTIVE

- The OrderEase app will serve as a primary tool for waiters to engage with Kumar Mess, providing them with a seamless and convenient way to place orders, make reservations, and browse the menu.
- The app will act as a marketing channel to showcase Kumar Mess's offerings, allowing the hotel to reach a wider audience and generate increased customer interest.
- The development team will leverage modern mobile app technologies to ensure the app's functionality, security, and performance align with industry standards and customer expectations.

2.2 USER CHARACTERISTICS

- **Waiters:** Waiters using the OrderEase app can browse the menu, place orders , make reservations, and provide feedback on dining experience. They may need to enter personal details such as their name, email address, and contact number when placing orders or making reservations.
- **Admin:** The admin team at Kumar Mess can access the app to manage menu items, track orders, monitor reservations, and review customer feedback. Admin users will have additional access and control over the app's settings and data.

2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS

2.3.1 Time

The development of the OrderEase app may be subject to specific timelines and deadlines, impacting the speed and efficiency of the design and implementation process.

2.3.2 Security

The app must adhere to high security standards to protect customer data, including secure login and payment features. Compliance with data protection regulations (such as GDPR or CCPA) may also be necessary.

2.3.3 Scalability

The app should be designed and developed with scalability in mind to accommodate future growth, such as an increased number of users, new features, or potential changes in the hotel's offerings.

2.3.4 User Experience

The app should offer an intuitive and seamless user experience, prioritizing ease of navigation and responsiveness across different devices and platforms. Accessibility standards must be considered to ensure inclusivity for all users.

CHAPTER 3

REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

3.1.1 User Registration and Login

- Users should be able to create an account and log in to access the app's features, such as placing orders, making reservations, and providing feedback.

3.1.2 Menu Browsing

- Users should be able to browse the hotel's menu, including viewing detailed descriptions, prices, and availability of menu items.

3.1.3 Order Placement

- Users should be able to place orders for pick-up or delivery through the app, specifying quantities and any customization options.

3.1.4 Order Status Tracking

- Users should be able to track the status of their orders in real-time, including estimated preparation time.

3.1.5 Suggestion

The customer can add their own suggestion to the waiter for their order so it will be a perfect customer experience.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 Performance

- The app should be optimized for fast loading times and smooth, responsive interactions to provide a positive user experience.
- Efficient use of resources should be maintained to avoid battery drain or excessive data usage.

3.2.2 Usability

- The app should feature an intuitive and user-friendly interface, making it easy for users to navigate the app and access its features.
- Clear and concise language should be used in the app to guide users and facilitate understanding.

3.2.3 Reliability

- The app should be reliable, with minimal downtime or interruptions to ensure continuous access for customers.
- Robust error handling should be in place to manage unexpected issues and maintain app functionality.

3.2.4 Maintenance

- The app should be designed to be easy to maintain and update, with a well-structured codebase and thorough documentation.

3.3 DESIGN

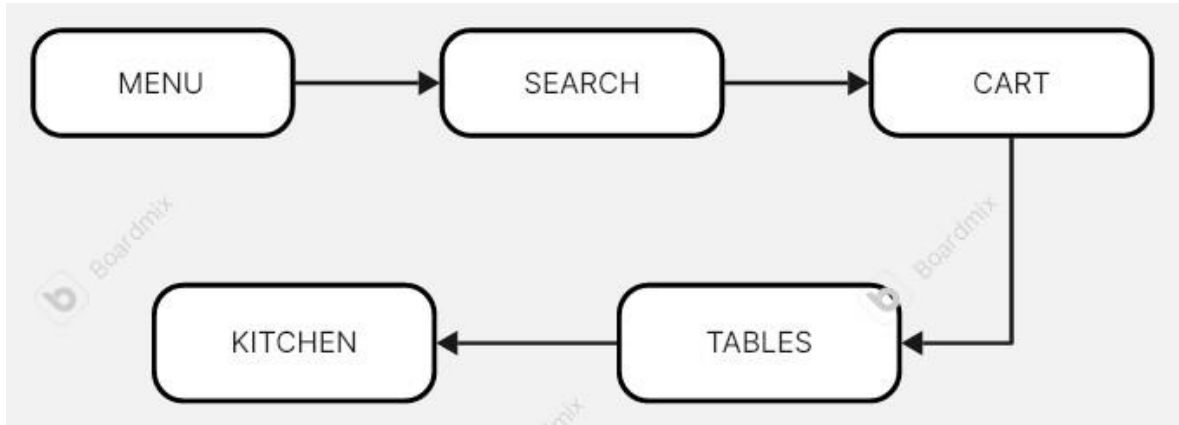


Fig.3.1.Block Diagram

CHAPTER 4

DETAILED DESIGN

4.1 ARCHITECTURAL DESIGN

The architectural design outlines the structure and behavior of the OrderEase mobile application for Kumar Mess. This includes different diagrams such as flow chart, that detail the interactions and workflows for all modules.

4.1.1 MODULE CHARACTERISTICS

The project contains four modules as follows and their description is given below.

- Menu module
- Search module
- Tables module
- Kitchen module

4.1.1.1 MENU MODULE

The Menu Module enables waiters to efficiently browse through the hotel's menu, providing access to detailed item descriptions and allowing seamless addition of items to their cart. It streamlines the ordering process, empowering waiters to provide quick and accurate service to customers.

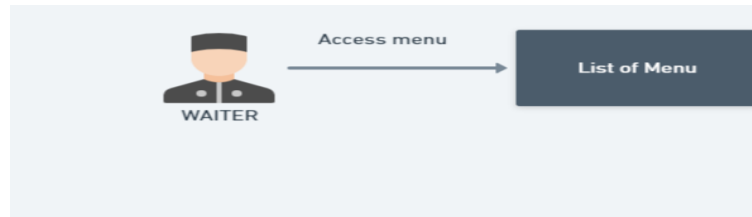


Fig.4.1.Menu

4.1.1.2 SEARCH MODULE

The Search Module empowers users to swiftly locate desired menu items or categories by employing keywords or filters, enhancing their browsing experience. It streamlines navigation, ensuring users can easily find what they're looking for, thereby facilitating efficient order placement.

4.1.1.3 TABLES MODULE

The Tables Module efficiently oversees table reservations and ongoing orders, providing seamless management for staff. It ensures smooth coordination between reservations and current dining activities, optimizing the dining experience for guests

4.1.1.4 KITCHEN MODULE

The Kitchen Module facilitates streamlined communication between kitchen staff and the front-end, managing order preparation and providing real-time status updates. It ensures efficient coordination, enhancing workflow and minimizing delays in food preparation.



Fig.4.2.Kitchen

4.2 INTERFACE DESIGN

The OrderEase mobile application provides a range of interfaces to enhance the user experience and make the app convenient and user-friendly. The design focuses on providing a clear and intuitive interface that is visually appealing and easy to navigate.

4.3 FLOW CHART

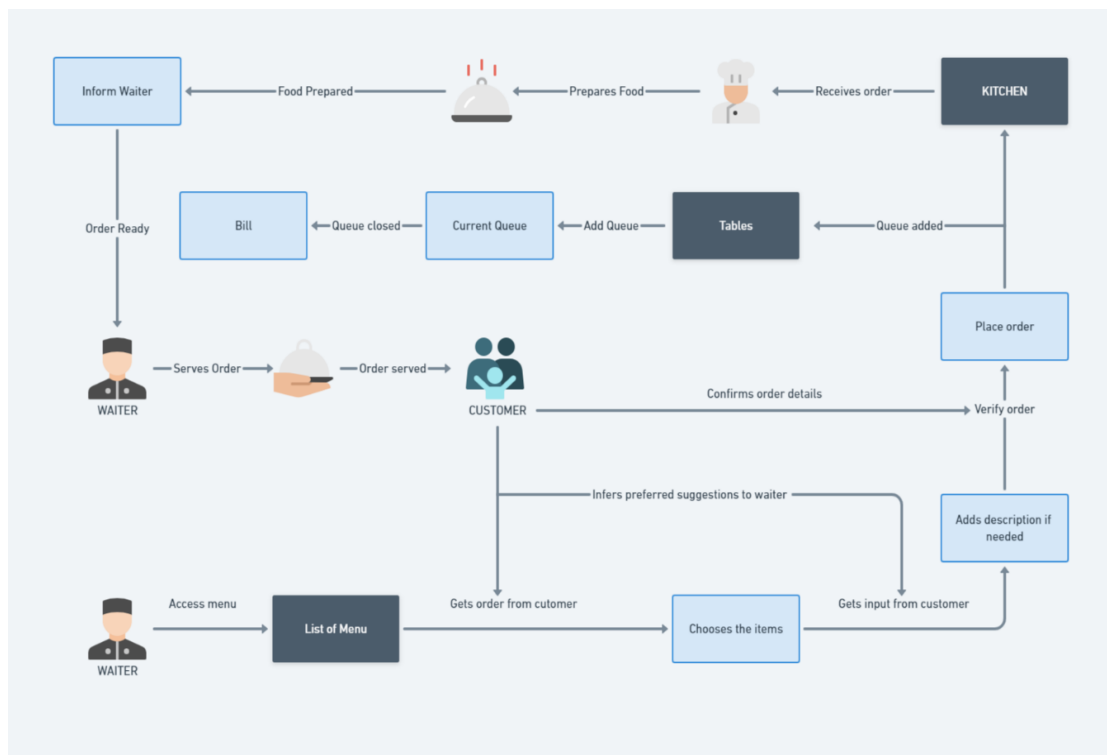


Fig.4.3.Flow chart

4.4 USE CASE

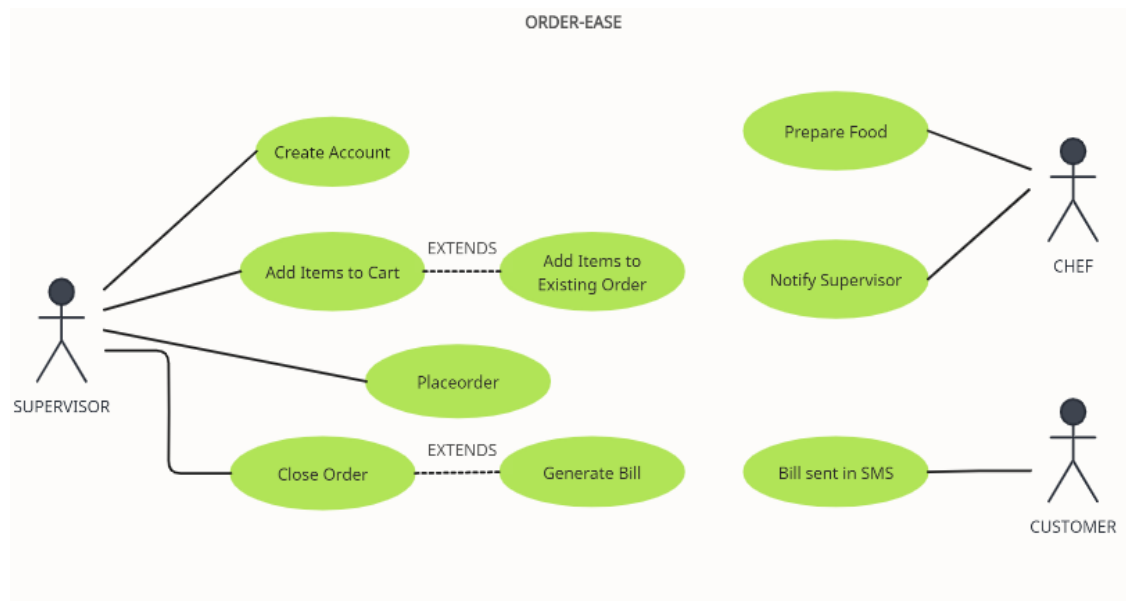


Fig.4.4.Usecase

4.5 SEQUENCE DIAGRAM

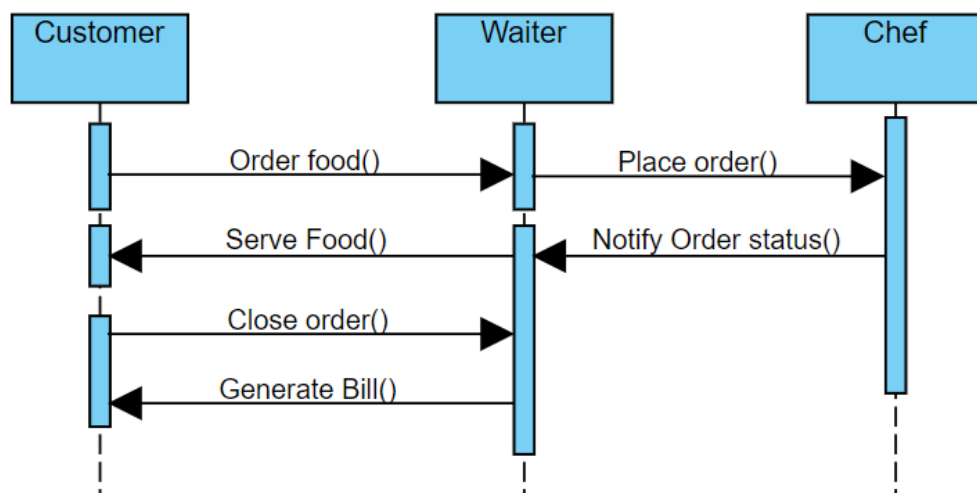


Fig.4.5.Sequence Diagram

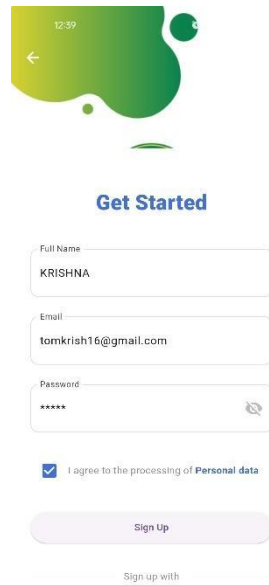
4.6 OUTPUT DESIGN

Output design refers to the results and information generated by the OrderEase mobile application for end-users, including both customers and administrators. The output design prioritizes understandability, visual appeal, convenience, and informativeness.

WELCOME PAGE:



Fig.4.6.Welcome Page

SIGNUP PAGE:

The mockup shows a mobile app interface for a signup page. At the top, there's a green header with a back arrow, a clock showing 12:39, and a profile icon. Below the header is a green decorative shape. The main heading is "Get Started" in blue. There are three input fields: "Full Name" with the value "KRISHNA", "Email" with the value "tomkrish16@gmail.com", and "Password" with masked characters "*****" and an eye icon. Below the password field is a checkbox labeled "I agree to the processing of Personal data". At the bottom is a "Sign Up" button and a "Sign up with" section.

12:39

←

Get Started

Full Name
KRISHNA

Email
tomkrish16@gmail.com

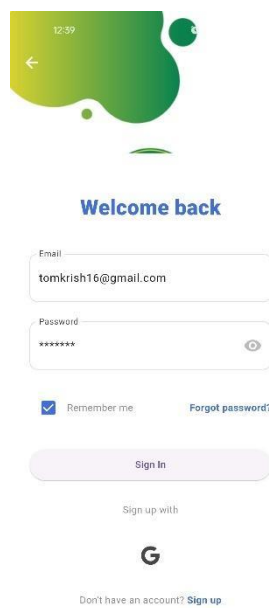
Password

☒ I agree to the processing of **Personal data**

Sign Up

Sign up with

Fig.4.7.Signup Page

SIGNIN PAGE:

The mockup shows a mobile app interface for a signin page. At the top, there's a green header with a back arrow, a clock showing 12:39, and a profile icon. Below the header is a green decorative shape. The main heading is "Welcome back" in blue. There are two input fields: "Email" with the value "tomkrish16@gmail.com" and "Password" with masked characters "*****" and an eye icon. Below the password field is a checkbox labeled "Remember me" and a link "Forgot password?". At the bottom is a "Sign In" button and a "Sign up with" section with a Google logo and a link "Don't have an account? Sign up".

12:39

←

Welcome back

Email
tomkrish16@gmail.com

Password

☒ Remember me [Forgot password?](#)

Sign In

Sign up with

G

Don't have an account? [Sign up](#)

Fig.4.8.Signin Page

MENU PAGE:

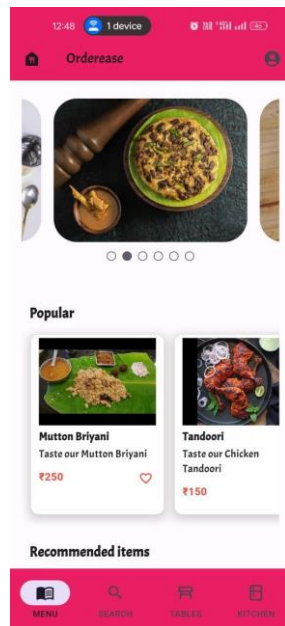


Fig.4.9.Menu Page

PROFILE PAGE:

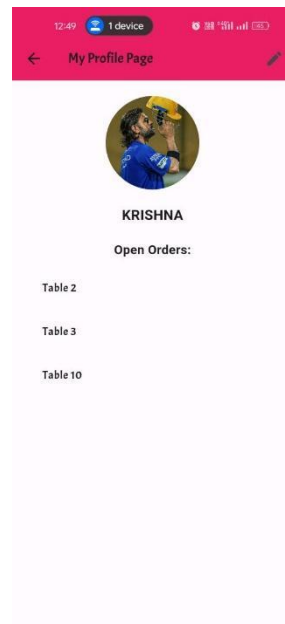


Fig.4.10.Profile Page

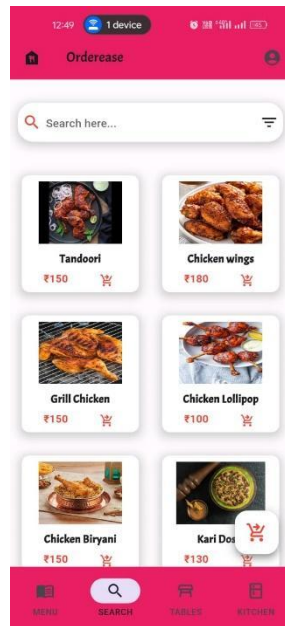
SEARCH PAGE:

Fig.4.11.Search Page

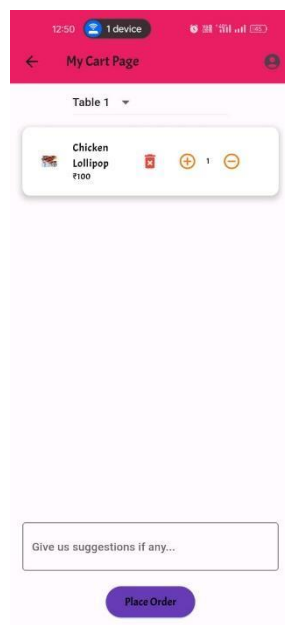
CART PAGE:

Fig.4.12.Cart Page

TABLES PAGE:

Fig.4.13.Tables Page

ORDER DETAILS PAGE:

Fig.4.14.Order Details Page

KITCHEN PAGE:

Fig.4.15.Kitchen Page

KITCHEN DETAILS PAGE:

Fig.4.16.Kitchen Details Page

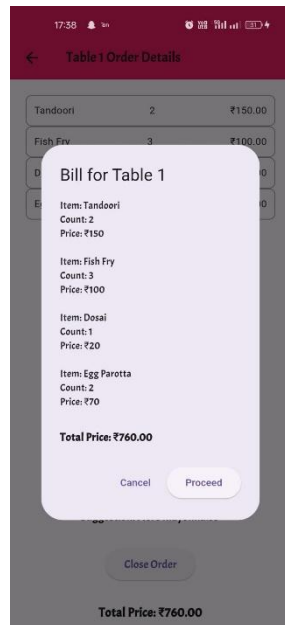
BILL DETAILS PAGE:

Fig.4.17.Bill

CHAPTER 5

TESTING

5.1 UNIT TESTING

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing can be done manually, but is often automated. In this process, each module is considered as individual units and are tested for proper operation. If each module meets up with the user's requirement, then it is subjected to integration testing where more than one module is integrated and tested.

5.2 REGRESSION TESTING

Regression testing helps in identifying any unintended side effects or issues that may arise due to code changes, updates to dependencies, or modifications in the underlying infrastructure. By rerunning test cases, the development team can ensure that previously implemented features and functionality have not been negatively impacted by recent updates. The primary objective of regression testing is to ensure that the website remains fully functional, reliable, and user-friendly, even after modifications or enhancements have been made.

Overall, regression testing serves as an essential quality assurance measure to maintain the integrity and performance of the website throughout its lifecycle. It helps in minimizing the risk of functional regressions, ensuring that the website continues to meet the expectations and requirements of the end users and stakeholders.

5.3 VALIDATION TESTING

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the service actually meets the client's needs. It can also be defined as to demonstrate that the service fulfils its intended use when deployed on appropriate environment.

5.4 VERIFICATION TESTING

Verification is the process of evaluating work-service of a development phase to determine whether they meet the specified requirements. When developing each module, the individuality of the service is checked at its development stage. Thus, the modules must be verified at the development stage.

5.5 INTEGRATION TESTING

Integration tests are designed to test the integrated software components to determine if they actually run as a program. It is specifically aimed at exposing problems that arise from the combination of components. Integration testing is done after integration of the model with the core service. The integration testing can be done for our project by integrating the user module.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The OrderEase mobile application for Kumar Mess represents a significant step forward in providing customers with a seamless and convenient dining experience. By offering a user-friendly interface for browsing the menu, placing orders, and making reservations, the app improves the overall customer experience and streamlines hotel operations. The app's robust features, such as secure payment integration, feedback and review options, and admin dashboard, contribute to enhancing Kumar Mess's efficiency and reputation. Moreover, the app's compatibility with different mobile platforms and devices ensures accessibility for a broad range of customers.

FUTURE WORK:

- **In-Table Ordering System:** As a future enhancement, the OrderEase app could be extended to include in-table ordering tablets for customers. These tablets would be placed at each dining table, allowing customers to browse the menu and place orders directly from their table.
- **Cashless Payment:** The in-table ordering system can be combined with cashless payment options, enabling customers to pay for their meals through the app after dining, further simplifying the dining experience and increasing operational efficiency.
- **Real-Time Updates:** Integrating real-time order updates into the in-table system can provide customers with live information on order status and estimated wait times, enhancing the dining experience and setting expectations.

- **Data Analytics:** The app can leverage data analytics to track customer preferences and behaviors, allowing Kumar Mess to tailor their offerings and marketing strategies to better suit their customer base.
- **Loyalty Programs:** Future iterations of the app could introduce customer loyalty programs, rewarding frequent diners with special discounts, promotions, or exclusive offers.

The OrderEase app has great potential for further development and enhancement to meet the evolving needs of Kumar Mess and its customers. By continuing to innovate and expand its capabilities, the app will solidify its role as an essential tool for the hotel and its patrons.

APPENDIX 1

GITHUB LINK:

https://github.com/Maniprabha06/Consultancy_Orderease.git

CODING:

welcome_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:orderease_new/screens/signin_screen.dart';
import 'package:orderease_new/screens/signup_screen.dart';
import 'package:orderease_new/theme/theme.dart';
import 'package:orderease_new/widgets/custom_scaffold.dart';
import 'package:orderease_new/widgets/welcome_button.dart';

class WelcomeScreen extends StatelessWidget {
  const WelcomeScreen({Key? key});

  @override
  Widget build(BuildContext context) {
    return CustomScaffold(
      child: Column(
        children: [
          Flexible(
            flex: 8,
            child: Container(
              padding: const EdgeInsets.symmetric(
                vertical: 0,
                horizontal: 40.0,
              ),
            child: Center(
```



```

child: Row(
  children: [
    Expanded(
      child: WelcomeButton(
        buttonText: 'Sign in',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => SignInScreen()),
          );
        },
        color: Colors.transparent,
        textColor: Colors.white,
      ),
    ),
    Expanded(
      child: WelcomeButton(
        buttonText: 'Sign up',
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => SignUpScreen()),
          );
        },
        color: Colors.white,
        textColor: lightColorScheme.primary,
      ),
    ),
  ],
)

```

```

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

```

cartPage.dart:

```

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
class CartPage extends StatefulWidget {
  final List<Map<String, dynamic>> selectedItems;
  const CartPage({Key? key, required this.selectedItems}) : super(key: key);
  @override
  _CartPageState createState() => _CartPageState();
}
class _CartPageState extends State<CartPage> {
  String selectedTable = 'Table 1';
  List<String> tableNumbers = List.generate(20, (index) => 'Table ${index + 1}');
  List<Map<String, dynamic>> uniqueItems = [];
  TextEditingController descriptionController = TextEditingController();
  @override
  void initState() {
    super.initState();

```

```

uniqueItems = widget.selectedItems.map((item) {
  return {
    'title': item['title'],
    'quantity': item['quantity'] ?? 1, // Default quantity to 1 if null
    'imagePath': item['imagePath'],
    'price': item['price'],
  };
}).toList();
}

// Function to place an order and save data to Firestore
Future<void> _placeOrder() async {
  // Prepare data to be saved to Firestore
  List<Map<String, dynamic>> newOrderDetails = uniqueItems.map((item) {
    return {
      'itemName': item['title'],
      'count': item['quantity'],
      'price': item['price'], // Include the price field
    };
  }).toList();

  // Retrieve suggestion from the text field
  String suggestion = descriptionController.text;

  // Query to find an existing order for the selected table
  final querySnapshot = await FirebaseFirestore.instance
    .collection('orders')
    .where('tableNumber', isEqualTo: selectedTable)
    .where('closed', isEqualTo: false) // Only consider open orders
    .get();

  if (querySnapshot.docs.isNotEmpty) {

```

```

// An existing order is found for the selected table
final existingOrderDoc = querySnapshot.docs.first;
final existingOrderData = existingOrderDoc.data();

// Retrieve existing order details
List<Map<String, dynamic>> existingOrderDetails = List<Map<String,
dynamic>>.from(existingOrderData['orderDetails']);

// Merge the existing order details with the new order details
for (var newItem in newOrderDetails) {
  bool itemExists = false;

  // Check if the new item already exists in the existing order
  for (var existingItem in existingOrderDetails) {
    if (existingItem['itemName'] == newItem['itemName']) {
      // Item exists, update its count and price
      existingItem['count'] += newItem['count'];

      // Ensure price is consistent
      existingItem['price'] = newItem['price'];

      itemExists = true;

      break;
    }
  }

  // If the item does not exist, add it to the existing order details
  if (!itemExists) {
    existingOrderDetails.add(newItem);
  }
}

// Update the existing Firestore document with the merged order details
await existingOrderDoc.reference.update({
  'orderDetails': existingOrderDetails,

```

```

        'suggestion': suggestion, // Update the suggestion as well
        'timestamp': Timestamp.now(),
    });
} else {
    // No existing order found, create a new order document
    await FirebaseFirestore.instance.collection('orders').add({
        'orderDetails': newOrderDetails,
        'tableNumber': selectedTable,
        'suggestion': suggestion,
        'timestamp': Timestamp.now(),
        'closed': false, // Ensure the order is marked as open
    });
}

// Clear the cart and reset the selected table and suggestion text field
setState(() {
    uniqueItems.clear();
    descriptionController.clear();
    selectedTable = 'Table 1';
});

// Show a success message
ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
        content: Text('Order placed successfully!'),
    ),
);
}

@override
Widget build(BuildContext context) {

```

```

return Scaffold(
  appBar: AppBar(
    title: Padding(
      padding: const EdgeInsets.only(right: 2.0),
      child: Text(
        'My Cart Page',
        style: GoogleFonts.acme(
          fontSize: 20.0,
        ),
      ),
    ),
    centerTitle: false,
    actions: <Widget>[
      IconButton(
        icon: const Icon(Icons.account_circle_sharp),
        onPressed: () {},
      ),
    ],
    backgroundColor: Colors.pink,
    leading: IconButton(
      onPressed: () {
        Navigator.of(context).pop();
      },
      icon: const Icon(Icons.arrow_back),
    ),
  ),
  body: Column(
    children: [

```

```

// Dropdown menu for selecting table numbers
Container(
  width: 200,
  child: DropdownButton<String>(
    value: selectedTable,
    onChanged: (String? newValue) {
      setState() {
        selectedTable = newValue!;
      };
    },
    items: tableNumbers.map<DropdownMenuItem<String>>((String value) {
      return DropdownMenuItem<String>(
        value: value,
        child: Text(value),
      );
    }).toList(),
  ),
),
// List of items in the cart
Expanded(
  child: ListView.builder(
    shrinkWrap: true,
    itemCount: uniqueItems.length,
    itemBuilder: (context, index) {
      return Container(
        margin: EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
        padding: EdgeInsets.all(8.0),
        decoration: BoxDecoration(

```



```

color: Colors.white,
borderRadius: BorderRadius.circular(10.0),
boxShadow: [
  BoxShadow(
    color: Colors.grey.withOpacity(0.5),
    spreadRadius: 2,
    blurRadius: 5,
    offset: Offset(0, 3),
  ),
],
),
child: ListTile(
  leading: Image.asset(
    uniqueItems[index]['imagePath'],
    width: 20,
    height: 20,
  ),
  title: Text(
    uniqueItems[index]['title'],
    style: GoogleFonts.acme(
      fontSize: 15.0,
      color: Colors.black,
    ),
  ),
  subtitle: Text(
    '₹${uniqueItems[index]['price']}',
    style: GoogleFonts.acme(
      fontSize: 12.0,

```

```

        color: Colors.black,
      ),
    ),
    trailing: Row(
      mainAxisAlignment: MainAxisAlignment.min,
      children: [
        // Delete item from the cart
        IconButton(
          onPressed: () {
            setState(() {
              uniqueItems.removeAt(index);
            });
          },
          icon: Icon(
            Icons.delete_forever_rounded,
            color: Colors.red,
          ),
        ),
        // Increase item quantity
        IconButton(
          onPressed: () {
            setState(() {
              if (uniqueItems[index]['quantity'] == null) {
                uniqueItems[index]['quantity'] = 1;
              }
              uniqueItems[index]['quantity']++;
            });
          },

```

```

    icon: Icon(
      Icons.add_circle_outline,
      color: Color.fromARGB(255, 248, 121, 11),
    ),
  ),
  // Display item quantity
  Text(
    '${uniqueItems[index]['quantity']}',
    style: GoogleFonts.acme(
      fontSize: 12.0,
      color: Colors.black,
    ),
  ),
  // Decrease item quantity
  IconButton(
    onPressed: () {
      setState() {
        if (uniqueItems[index]['quantity'] != null &&
            uniqueItems[index]['quantity'] > 1) {
          uniqueItems[index]['quantity']--;
        }
      };
    },
    icon: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Icon(
        Icons.remove_circle_outline,
        color: Color.fromARGB(255, 248, 121, 11),

```

```

        ),
      ),
    ),
  ],
),
),
);
},
),
),
  SizedBox(height: 16),
  // Text field for user suggestion
  Padding(
    padding: const EdgeInsets.all(16.0),
    child: TextField(
      controller: descriptionController,
      decoration: InputDecoration(
        hintText: 'Give us suggestions if any...',
        border: OutlineInputBorder(),
      ),
    ),
  ),
),
  // Button to place order
  ElevatedButton(
    onPressed: _placeOrder,
    style: ElevatedButton.styleFrom(
      shadowColor: Colors.blue,
      backgroundColor: Colors.deepPurple,

```

```

    ),
    child: Text(
      'Place Order',
      style: GoogleFonts.acme(
        fontSize: 16.0,
        color: Colors.black,
      ),
    ),
  ),
  SizedBox(height: 16),
],
),
);
}
}

```

firebase_options.dart:

```

class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:

```

```

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

static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyBInu7Gey0oNm90vqZnW0AHOM_-Z1eDzWM',
  appId: '1:991713497116:web:5a5c241d69e2ea18704fd9',
  messagingSenderId: '991713497116',
  projectId: 'orderease-832b3',
  authDomain: 'orderease-832b3.firebaseio.com',
  storageBucket: 'orderease-832b3.appspot.com',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyC1E2cH0Ku0innZJjdbszBIvYmuO1C2_fQ',
  appId: '1:991713497116:android:46c0aafc85e63366704fd9',

```

```

    messagingSenderId: '991713497116',
    projectId: 'orderease-832b3',
    storageBucket: 'orderease-832b3.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyBCcopsJJJa_y_sit2AgSy6aDHmnQr9KfZw',
    appId: '1:991713497116:ios:849d1b8ea3029ee8704fd9',
    messagingSenderId: '991713497116',
    projectId: 'orderease-832b3',
    storageBucket: 'orderease-832b3.appspot.com',
    iosClientId: '991713497116-
lppk16iptcvcs446c9qoie66edluq3a.apps.googleusercontent.com',
    iosBundleId: 'com.example.ordereaseNew',
  );

  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyBCcopsJJJa_y_sit2AgSy6aDHmnQr9KfZw',
    appId: '1:991713497116:ios:5f33f5d9c243b53f704fd9',
    messagingSenderId: '991713497116',
    projectId: 'orderease-832b3',
    storageBucket: 'orderease-832b3.appspot.com',
    iosClientId: '991713497116-
1b7ieul06ul43lb7mjfcg2kh2eut5m8o.apps.googleusercontent.com',
    iosBundleId: 'com.example.ordereaseNew.RunnerTests',
  );
}

```

HomePage.dart:

```
import 'package:flutter/material.dart';
```

```

import 'package:orderease_new/Carousel_Slider.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:orderease_new/popular.dart';
import 'package:orderease_new/NewItem.dart';
class HomePageWidget extends StatelessWidget {
  const HomePageWidget({ Key? key }) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: ListView(
        padding: EdgeInsets.all(15),
        children: [
          // SizedBox(height: 20), // Adding space between the search bar and carousel slider
          MyCarouselSlider(),
          Padding(
            padding: EdgeInsets.only(top: 20, left: 10),
            child: Text(
              "Popular",
              style: GoogleFonts.acme(
                textStyle: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 20,
                ),
              ),
            ),
            PopularItemPage(),
            Padding(

```



```

padding: EdgeInsets.only(top: 20, left: 10),
child: Text(
  "Recommended items",
  style: GoogleFonts.acme(
    textStyle: TextStyle(
      fontWeight: FontWeight.bold,
      fontSize: 20,
    ),
  ),
),
),
),
NewestItemsPage(),
],
),
);
}
}

```

kitchen.dart:

```

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:orderease_new/table_items_page.dart';
class KitchenPage extends StatelessWidget {
  const KitchenPage({Key? key}) : super(key: key);
  Future<bool> hasItems(int tableNumber) async {
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;
    try {

```

```

QuerySnapshot querySnapshot = await _firestore
    .collection('orders')
    .where('tableNumber', isEqualTo: 'Table $tableNumber')
    .orderBy('timestamp', descending: true)
    .limit(1)
    .get();

if (querySnapshot.docs.isNotEmpty) {
    final orderData = querySnapshot.docs.first.data() as Map<String, dynamic>;
    final orderDetails = orderData?['orderDetails'] as List<dynamic>? ?? [];
    return orderDetails.isNotEmpty;
}

} catch (e) {
    print('Error fetching order data for Table $tableNumber: $e');
}

return false; // No items if there was an error or no data
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        body: ListView.builder(
            itemCount: 20, // Number of tables
            itemBuilder: (context, index) {
                int tableNumber = index + 1; // Table number
                return FutureBuilder<bool>(
                    future: hasItems(tableNumber), // Check if the table has items
                    builder: (context, snapshot) {
                        if (snapshot.connectionState == ConnectionState.waiting) {
                            // Show a loading indicator while fetching data

```

```

    return Center(child: CircularProgressIndicator());}

// Determine the image based on the presence of items
final String imagePath = snapshot.data == true
    ? 'assets/reserved_table.jpg' // Reserved table image
    : 'assets/unreserved_table.jpg'; // Unreserved table image
return GestureDetector(
  onTap: () {
    // Navigate to the TableItemsPage when a table is pressed
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => TableItemsPage(tableNumber: tableNumber),
      ),
    );
  },
  child: Column(
    children: [
      SizedBox(
        height: 250, // Set the desired height
        child: Container(
          padding: EdgeInsets.all(10),
          margin: EdgeInsets.symmetric(vertical: 25, horizontal: 30),
          decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(10),
            boxShadow: [
              BoxShadow(
                color: Colors.grey.withOpacity(0.5),

```

```

        spreadRadius: 3,
        blurRadius: 10,
        offset: Offset(0, 3),
      ),
    ],
    image: DecorationImage(
      image: AssetImage(imagePath), // Use the determined image path
      fit: BoxFit.cover, // Cover the entire container
    ),
  ),
),
),
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Icon(Icons.table_restaurant_outlined), // Table icon
    SizedBox(width: 5),
    Text(
      'Table $tableNumber', // Table number
      style: GoogleFonts.acme(fontSize: 16),
    ),
  ],
),
),
);
},
);

```

```

    },
  ),
);
}
}

```

main.dart:

```

import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:orderease_new/firebase_options.dart';
import 'package:orderease_new/navigation_bar.dart';
import 'package:orderease_new/cartPage.dart';
import 'package:orderease_new/screens/signin_screen.dart';
import 'package:orderease_new/screens/welcome_screen.dart';
import 'package:orderease_new/ProfilePage.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform
  );
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(

```

```

debugShowCheckedModeBanner: false,

routes: {

  '/': (context) => WelcomeScreen(), // Route for the welcome screen

  'navBar': (context) => Navigationbar(), // Route for the navigation bar screen

  'signin': (context) => SignInScreen(), // Route for the sign-in screen

  //'cartPage': (context) => CartPage(), // Route for the cart page

  'profilePage': (context) => ProfilePage(),

},

);

}

}

```

navigation_bar.dart:

```

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:orderease_new/HomePage.dart';

import 'package:google_fonts/google_fonts.dart';

import 'package:orderease_new/tables.dart';

import 'package:orderease_new/kitchen.dart';

import 'package:orderease_new/SearchPage.dart';

class Navigationbar extends StatelessWidget{

  const Navigationbar({super.key});

  @override build(BuildContext context){

    final controller = Get.put(Navigationcontroller());

    return Scaffold(

      appBar: AppBar(

        title: Padding(

```

```

padding: const EdgeInsets.only(right: 2.0),
child: Text(
  'Orderease',
  style: GoogleFonts.acme( // Example: Using Open Sans font
    fontSize: 20.0,
    // fontWeight: FontWeight.bold,
  ),
),
),
centerTitle: false,
actions: [
  IconButton(
    icon: Icon(Icons.account_circle_sharp),
    onPressed: () {
      Navigator.pushNamed(context, 'profilePage');
    },
  ),

],
backgroundColor: Colors.pink,
leading: IconButton(
  onPressed: () {},
  icon: const Icon(Icons.food_bank,
),
),
),
bottomNavigationBar: Obx(
  ()=> NavigationBar(

```

```

        height: 80,
        backgroundColor: Colors.pink,
        elevation: 0,
        selectedIndex: controller.selectedindex.value,
        onDestinationSelected: (index)=> controller.selectedindex.value = index,
        destinations: [
            NavigationDestination(icon: Icon(Icons.menu_book), label: 'MENU'),
            NavigationDestination(icon: Icon(Icons.search), label: 'SEARCH'),
            NavigationDestination(icon: Icon(Icons.table_restaurant_outlined), label:
'TABLES'),
            NavigationDestination(icon: Icon(Icons.kitchen_outlined), label: 'KITCHEN'),
        ],
    ),
),
body: Obx(()=> controller.screens[controller.selectedindex.value]),
);
}
}

class Navigationcontroller extends GetxController{
    final Rx<int> selectedindex = 0.obs;
    final screens =[HomePageWidget(),
        SearchPage(),
        TablePage(),
        KitchenPage(),
    ];
}

```


orderdetailspage.dart:

```

import 'package:flutter/material.dart';

import 'package:google_fonts/google_fonts.dart';

class OrderDetailsPage extends StatelessWidget {

  final String tableNumber;

  final List<dynamic> orderDetails;

  const OrderDetailsPage({Key? key, required this.tableNumber, required
this.orderDetails}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

          'Order Details ($tableNumber)',

          style: GoogleFonts.acme(fontSize: 20.0),

        ),

        centerTitle: false,

        backgroundColor: Colors.pink,

        leading: IconButton(

          onPressed: () {

            Navigator.of(context).pop();

          },

          icon: const Icon(Icons.arrow_back),

        ),

      ),

      body: ListView.builder(

```

```

    itemCount: orderDetails.length,
    itemBuilder: (context, index) {
      final item = orderDetails[index];
      return ListTile(
        title: Text(
          item['itemName'],
          style: GoogleFonts.acme(fontSize: 16),
        ),
        subtitle: Text(
          'Count: ${item['count']}, Price: ₹${item['price']}',
          style: GoogleFonts.acme(fontSize: 12),
        ),
      );
    },
  ),
);
}
}

```

ProfilePage.dart:

```

import 'dart:io';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:permission_handler/permission_handler.dart';

```

```

import 'package:orderease_new/orderdetailspage.dart';

class ProfilePage extends StatefulWidget {
  const ProfilePage({Key? key}) : super(key: key);

  @override
  State<ProfilePage> createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final FirebaseStorage _storage = FirebaseStorage.instance;
  File? _imageFile;
  String _name = "";
  String? _imageUrl;
  bool _isPermanentProfilePicture = false; // Added to track the permanent statu
  bool _isEditing = false;
  final _nameController = TextEditingController();
  List<Map<String, dynamic>> openOrders = []; // List to hold open orders
  String selectedTable = ""; // Selected table number for order details

  @override
  void initState() {
    super.initState();
    _loadUserData();
    _retrieveOpenOrders();
  }

  Future<void> _loadUserData() async {
    final user = _auth.currentUser;
    if (user != null) {
      final userDoc = await _firestore.collection('users').doc(user.uid).get();

```

```

if (userDoc.exists) {
  setState(() {
    _name = userDoc['fullName'];
    _imageUrl = userDoc['profileImageUrl'];
    _isPermanentProfilePicture = userDoc['isPermanentProfilePicture'] ?? false;
    _nameController.text = _name;
  });
}
}
}

Future<void> _retrieveOpenOrders() async {
  final querySnapshot = await _firestore.collection('orders').where('closed', isEqualTo:
false).get();
  if (querySnapshot.docs.isNotEmpty) {
    setState(() {
      openOrders = querySnapshot.docs.map((doc) => doc.data() as Map<String,
dynamic>).toList();
    });
  }
}

Future<void> _checkPermission() async {
  final permissionStatus = await Permission.photos.request();
  if (permissionStatus.isGranted) {
    _pickImage(); // Permission granted, proceed with image picking
  } else if (permissionStatus.isPermanentlyDenied) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text(

```

```

        'Permission permanently denied. Please enable photo access in settings.',
    ),
    ),
);
await openAppSettings();
} else {
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Permission denied. Cannot access photos.')),
    );
}
}

Future<void> _pickImage() async {
    final picker = ImagePicker();
    final pickedFile = await picker.pickImage(source: ImageSource.gallery);
    if (pickedFile != null) {
        setState(() {
            _imageFile = File(pickedFile.path);
        });
        await _uploadImage();
    } else {
        print('No image selected.');
```

```
    }
```

```
}
```

```
Future<void> _uploadImage() async {
```

```
    final user = _auth.currentUser;
```

```
    if (user != null && _imageFile != null) {
```

```
        final storageRef = _storage.ref().child('profile_images/${user.uid}');
```

```
        final uploadTask = storageRef.putFile(_imageFile!);
```

```

    await uploadTask.whenComplete(() async {
      final downloadUrl = await storageRef.getDownloadURL();
      setState(() {
        _imageUrl = downloadUrl;
      });
      await _updateUserProfile();
    });
  } else {
    print('Error: User not signed in or image file not available.');
```

```

  }
}

Future<void> _updateUserProfile() async {
  final user = _auth.currentUser;
  if (user != null) {
    await _firestore.collection('users').doc(user.uid).update({
      'fullName': _name,
      'profileImageUrl': _imageUrl,
      'isPermanentProfilePicture': true, // Set isPermanentProfilePicture to true
    });
  } else {
    print('Error: User not signed in.');
```

```

  }
}

void _toggleEditMode() {
  setState(() {
    _isEditing = !_isEditing;
  });
}

```

```

    });
}

void _saveChanges() async {
  setState(() {
    _name = _nameController.text;
  });
  await _updateUserProfile();
  _toggleEditMode();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(
        'My Profile Page',
        style: GoogleFonts.acme(fontSize: 20.0),
      ),
      centerTitle: false,
      backgroundColor: Colors.pink,
      leading: IconButton(
        onPressed: () {
          Navigator.of(context).pop();
        },
        icon: const Icon(Icons.arrow_back),
      ),
      actions: [
        if (!_isEditing)

```

```

        IconButton(
            icon: const Icon(Icons.edit),
            onPressed: _toggleEditMode,
        ),
    ],
),
body: SingleChildScrollView(
    padding: const EdgeInsets.all(20.0),
    child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
            // Profile Image
            Stack(
                children: [
                    CircleAvatar(
                        radius: 60,
                        backgroundImage: _imageUrl != null && _isPermanentProfilePicture
                            ? NetworkImage(_imageUrl!)
                            : null,
                        child: _imageUrl == null || !_isPermanentProfilePicture
                            ? const Icon(Icons.account_circle, size: 80)
                            : null,
                    ),
                    if (_isEditing)
                        Positioned(
                            bottom: 0,
                            right: 0,
                            child: IconButton(

```



```

        onPressed: _pickImage,
        icon: const Icon(Icons.edit),
      ),
    ),
  ],
),
const SizedBox(height: 20),
// Name Input
if (_isEditing)
  TextFormField(
    controller: _nameController,
    decoration: const InputDecoration(label: Text('Name')),
  )
else
  Text(
    _name,
    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
  ),
const SizedBox(height: 20),
// Open Orders Section
const Text(
  'Open Orders:',
  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
),
const SizedBox(height: 10),
// List of open orders
ListView.builder(
  shrinkWrap: true,

```

```

itemCount: openOrders.length,
itemBuilder: (context, index) {
    final orderData = openOrders[index];
    final tableNumber = orderData['tableNumber'];

    return ListTile(
        title: Text(
            '$tableNumber',
            style: GoogleFonts.acme(fontSize: 16),
        ),
        onTap: () {
            // Navigate to a detailed order view for the selected table
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => OrderDetailsPage(
                        tableNumber: tableNumber,
                        orderDetails: orderData['orderDetails'] as List<dynamic>,
                    ),
                ),
            );
        },
    );
},
);
if (_isEditing)
    ElevatedButton(
        onPressed: _saveChanges,

```

```

        child: const Text('Save'),
      ),
    ],
  ),
),
);
}
}

```

tables.dart:

```

import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:twilio_flutter/twilio_flutter.dart';

class TablePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        scrollDirection: Axis.vertical,
        child: ListView(
          shrinkWrap: true,
          physics: NeverScrollableScrollPhysics(),
          children: List.generate(
            20, // Assuming there are 20 tables
            (index) {
              int tableNumber = index + 1;

```

```

return Container(
  width: MediaQuery.of(context).size.width,
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: InkWell(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => OrderedFoodDetailsPage(tableNumber:
tableNumber),
          ),
        );
      },
      child: Column(
        children: [
          Card(
            elevation: 4.0,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(12.0),
            ),
            child: Container(
              height: 200, // Increased container height
              width: 400, // Increased container width
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(12.0), // Add border radius
              ),
              child: ClipRRect(

```



```

}

class OrderedFoodDetailsPage extends StatelessWidget {
  final int tableNumber;

  OrderedFoodDetailsPage({required this.tableNumber});

  @override
  Widget build(BuildContext context) {
    // Initialize TwilioFlutter with your Twilio account SID, auth token, and phone number
    final twilioFlutter = TwilioFlutter(
      accountSid: '+',//SID
      authToken: '+',//AUTH TOKEN
      twilioNumber: '+*',//NUMBER
    );
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Table $tableNumber Order Details',
          style: GoogleFonts.acme(fontSize: 20.0),
        ),
        backgroundColor: Colors.pink,
        leading: IconButton(
          icon: Icon(Icons.arrow_back),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
      body: FutureBuilder<DocumentSnapshot>(
        future: _fetchOrderData(tableNumber),

```

```

builder: (context, snapshot) {
  if (snapshot.connectionState == ConnectionState.waiting) {
    return Center(child: CircularProgressIndicator());
  } else if (snapshot.hasError) {
    return Center(
      child: Text('Error loading order details: ${snapshot.error}'),
    );
  }

  final orderData = snapshot.data?.data() as Map<String, dynamic>;

  if (orderData == null || orderData.isEmpty) {
    return Center(
      child: Text('Currently no orders for Table $tableNumber'),
    );
  }

  final orderDetails = orderData['orderDetails'] as List<dynamic>? ?? [];
  final suggestion = orderData['suggestion'] as String?;

  if (orderDetails.isEmpty) {
    return Center(
      child: Text('Currently no orders for Table $tableNumber'),
    );
  }

  final totalPrice = _calculateTotalPrice(orderDetails);

```

```

return Column(
  children: [
    Expanded(
      child: ListView.builder(
        padding: const EdgeInsets.all(20.0),
        itemCount: orderDetails.length,
        itemBuilder: (context, index) {
          final item = orderDetails[index];
          final itemName = item['itemName'] as String?;
          final count = item['count'] as int?;
          final price = item['price']?.toString() ?? '0';

          return _buildFoodRow(context, itemName ?? 'Unknown Item',
count?.toString() ?? '0', price);
        },
      ),
    ),
    Padding(
      padding: const EdgeInsets.all(10.0),
      child: Text(
        'Suggestion: $suggestion',
        style: GoogleFonts.acme(
          fontSize: 16.0,
          color: Colors.black,
        ),
      ),
    ),
    Padding(

```



```

padding: const EdgeInsets.all(16.0),
child: ElevatedButton(
  onPressed: () => _closeOrder(context, tableNumber, twilioFlutter, orderDetails,
totalPrice),
  child: Text(
    'Close Order',
    style: GoogleFonts.acme(fontSize: 16.0),
  ),
),
),
),
Padding(
  padding: const EdgeInsets.all(10.0),
  child: Text(
    'Total Price: ₹${totalPrice.toStringAsFixed(2)}',
    style: GoogleFonts.acme(fontSize: 18, fontWeight: FontWeight.bold),
  ),
),
],
);
},
),
);
}

```

```

Future<DocumentSnapshot> _fetchOrderData(int tableNumber) async {
  final querySnapshot = await FirebaseFirestore.instance
    .collection('orders')
    .where('tableNumber', isEqualTo: 'Table $tableNumber')

```

```

        .orderBy('timestamp', descending: true)
        .limit(1)
        .get();

    if (querySnapshot.docs.isNotEmpty) {
        return querySnapshot.docs.first;
    } else {
        throw Exception('No data found');
    }
}

Widget _buildFoodRow(BuildContext context, String itemName, String quantity, String
price) {

    final NumberFormat currencyFormatter = NumberFormat.currency(

        locale: 'en_IN', // Use 'en_IN' for Indian locale

        symbol: '₹'); // Indian Rupee symbol

    final formattedPrice = currencyFormatter.format(double.parse(price));

    return Container(

        padding: const EdgeInsets.all(10.0),

        decoration: BoxDecoration(

            border: Border.all(color: Colors.grey),

            borderRadius: BorderRadius.circular(8.0),

        ),

        child: Row(

            mainAxisAlignment: MainAxisAlignment.spaceBetween,

            children: [

```

```

        Text(itemName),
        Text(quantity),
        Text(formattedPrice),
    ],
),
);
}

double _calculateTotalPrice(List<dynamic> orderDetails) {
    double totalPrice = 0.0;
    for (var item in orderDetails) {
        final price = double.tryParse(item['price']?.toString() ?? '0') ?? 0;
        final count = item['count'] as int?;
        totalPrice += price * (count ?? 1);
    }
    return totalPrice;
}

Future<void> _closeOrder(
    BuildContext context,
    int tableNumber,
    TwilioFlutter twilioFlutter,
    List<dynamic> orderDetails,
    double totalPrice) async {
    final result = await showDialog<bool>(
        context: context,
        builder: (context) {
            return AlertDialog(
                title: Text('Close Order'),
                content: Text('Are you sure you want to close this order?'),
            );
        },
    );
    if (result == true) {
        twilioFlutter.closeOrder(tableNumber);
    }
}

```

```

actions: [
  TextButton(
    onPressed: () {
      Navigator.of(context).pop(false);
    },
    child: Text('No'),
  ),
  TextButton(
    onPressed: () {
      Navigator.of(context).pop(true);
    },
    child: Text('Yes'),
  ),
],
);
},
);

if (result == true) {
  final phoneNumber = await _askForPhoneNumber(context);
  if (phoneNumber != null) {
    final orderDoc = await FirebaseFirestore.instance
      .collection('orders')
      .where('tableNumber', isEqualTo: 'Table $tableNumber')
      .orderBy('timestamp', descending: true)
      .limit(1)
      .get()
      .then((querySnapshot) => querySnapshot.docs.first.reference);

```

```

await orderDoc.update({
  'orderDetails': [],
  'suggestion': null,
  'closed': true,
});

await FirebaseFirestore.instance.collection('phoneNumbers').add({
  'phoneNumber': phoneNumber,
  'tableNumber': tableNumber,
  'timestamp': FieldValue.serverTimestamp(),
});

final orderDetailsMessage = orderDetails.map((item) {
  final itemName = item['itemName'] as String?;
  final count = item['count'] as int?;
  final price = double.tryParse(item['price']?.toString() ?? '0') ?? 0;

  return 'Item: $itemName\nCount: $count\nPrice: ₹$price\n';
}).join('\n');

try {
  await twilioFlutter.sendSMS(
    toNumber: phoneNumber,
    messageBody: 'KUMAR MESS PVT LTD\n'+ 'YOUR BILL\n'+ 'Order details for
Table $tableNumber:\n' + orderDetailsMessage + '\nTotal Price:
₹${totalPrice.toStringAsFixed(2)}\n'+ 'Thanks for visiting Kumar Mess.',
  );

```

```

        print('SMS sent successfully');
      } catch (e) {
        print('Failed to send SMS: $e');
      }
      Navigator.pop(context);
    }
  }
}

Future<String?> _askForPhoneNumber(BuildContext context) async {
  final phoneController = TextEditingController();
  final result = await showDialog<String?>(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: Text('Enter Phone Number'),
        content: TextField(
          controller: phoneController,
          keyboardType: TextInputType.phone,
          decoration: InputDecoration(
            hintText: 'Phone Number',
          ),
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(null);
            },
            child: Text('Cancel'),

```

```
    ),  
    ElevatedButton(  
      onPressed: () {  
        Navigator.of(context).pop(phoneController.text);  
      },  
      child: Text('Submit'),  
    ),  
  ],  
);  
},  
);  
return result;  
}  
}
```

APPENDIX 2

SNAPSHOTS

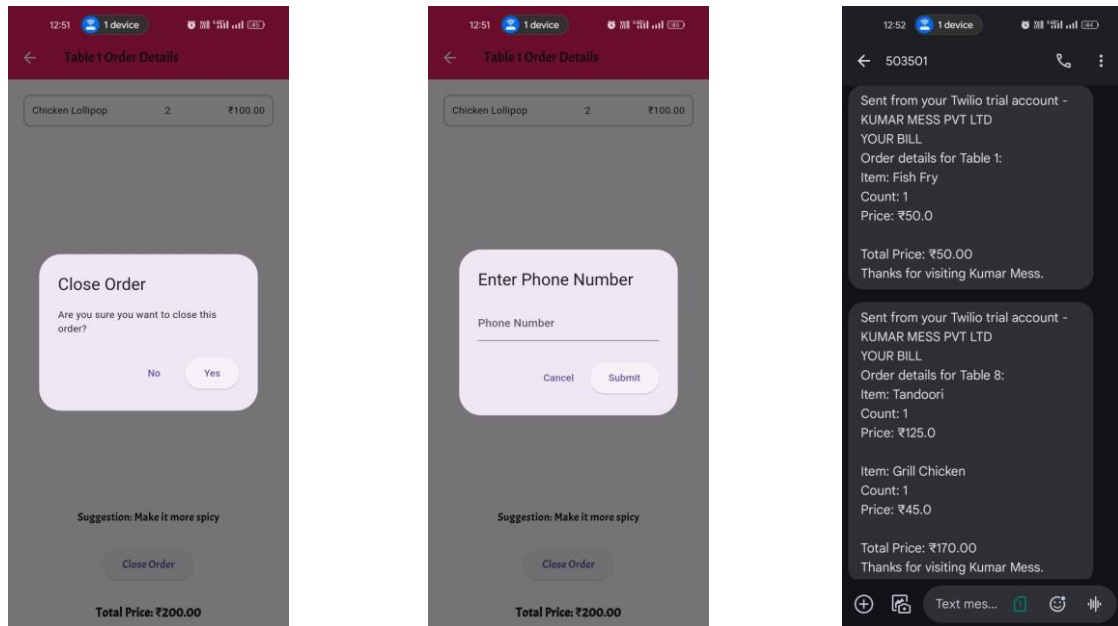


Fig.A2.1.Close Order

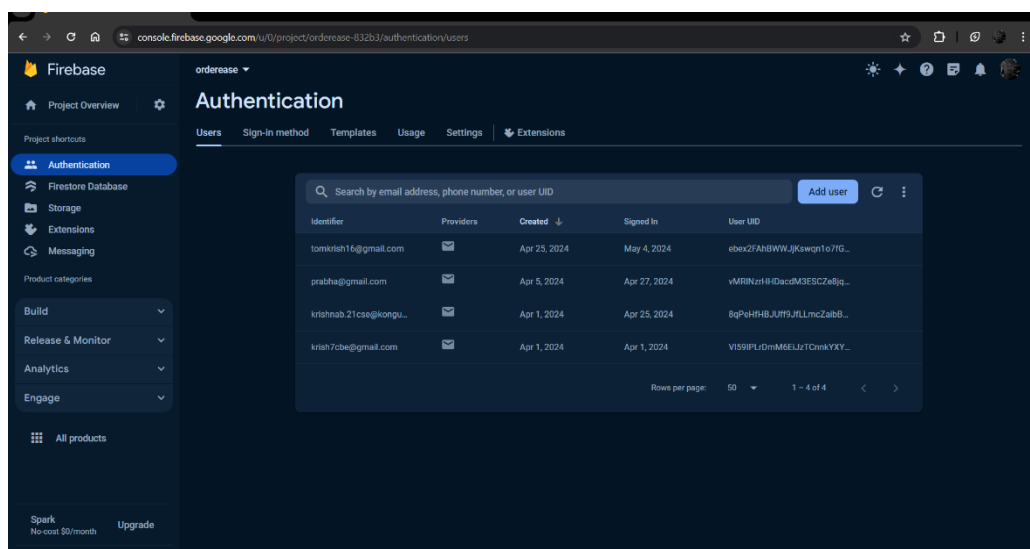


Fig.A2.2.Firebase–Authentication

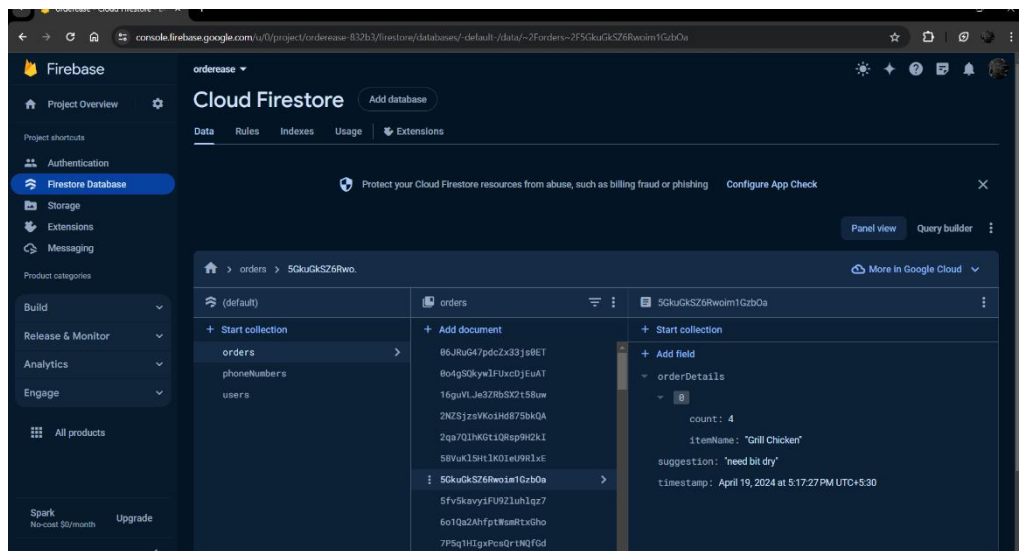


Fig.A2.3.Firebase–Firestore

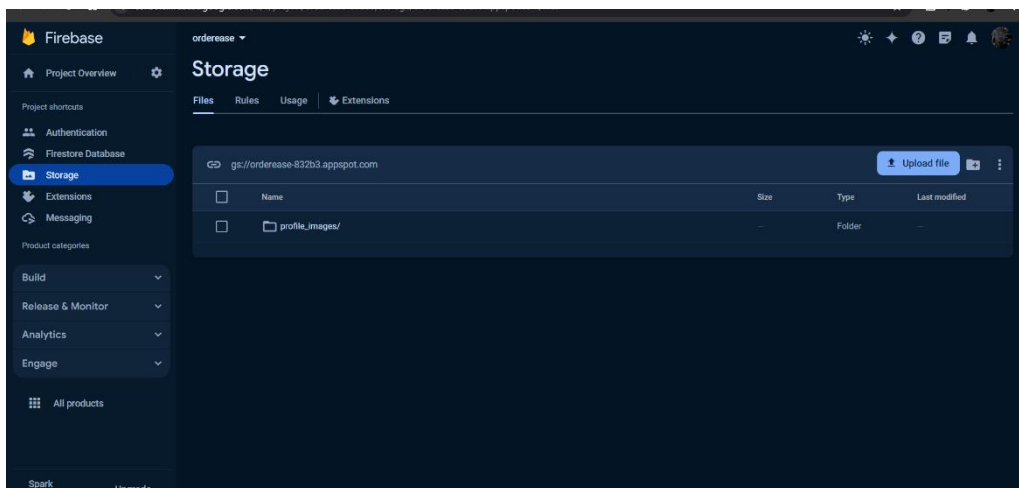


Fig.A2.4.Firebase–Storage

REFERENCES

- [1] Zomato Restaurant Partner(<https://play.google.com/store/apps/details?id=com.application.services.partner>)
- [2] Restaurant & Café BillingPOS(<https://play.google.com/store/apps/details?id=com.microtech.microtechcaptain.posCloud>)
- [3] Zomato(<https://play.google.com/store/apps/details?id=com.application.zomato>)
- [4] Restaurant Order-Taking App(<https://play.google.com/store/apps/details?id=com.globalfoodsoft.restaurantapp>)

CONSULTANCY REQUEST LETTER



Kumar Mess Private Limited

Regd. Office : RMG Illam, 8A/1, New Natham Road, Race Course,
Madurai - 625 002. Tamilnadu, India. Mobile : + 91 98403 64646, 98421 44576
Email : kumarmesspvtltd@gmail.com www.maduraimess.com
GST : 33AAFCCK2698Q1ZT PAN : AAFCCK2698Q
Admin. Office : 56, Alagar koll Road, Madurai - 625 002. Tamilnadu, India.
Tel.: + 91 452 2534575

03/02/2024

Dear Sir/Madam,

We have privilege in requesting the faculty **Ms. P KALAIVANI** of Department of Computer Science and Engineering at Kongu Engineering College, Perundurai to develop the business application for our concern. Below are the features that need to be included in the application.

- MENU
- PLACE ORDER
- KITCHEN
- TABLES
- BILL GENERATION

We will pay Rs.10,000/- and Rs.1800/- (18% GST) towards the software development. Further maintenance, if needed, will be requested on chargeable basis, as per your standard rates. We are confident that your team will be able to deliver an application that aligns with our requirements and represents our business effectively.

Thanking you for considering our request. We look forward to the opportunity to collaborate with you and your team on this project.

Regards

For KUMAR MESS PVT LTD


Director

BRANCHES @ MADURAI
TALLAKULAM - TOWNHALL ROAD - MATTUTHAVANI - BY PASS ROAD

BRANCHES @ CHENNAI
VADAPALANI - TAMBARAM RLY. STATION - T. NAGAR

CONSULTANCY COMPLETION LETTER



Kumar Mess Private Limited

Regd. Office : RMG Illam, 8A/1, New Natham Road, Race Course,
Madurai - 625 002. Tamilnadu, India. Mobile : + 91 98403 64646, 98421 44576
Email : kumarmesspvtltd@gmail.com www.maduralkumarmess.com
GST : 33AAFCK2698Q1ZT PAN : AAFCK2698Q
Admin. Office : 56, Alagar koil Road, Madurai - 625 002. Tamilnadu, India.
Tel.: + 91 452 2534575

DATE : 02/05/2024

Dear Sir / Madam,

We have great privilege to announce that the students under the guidance of **Ms. P KALAIVANI** of Department of Computer Science and Engineering. Kongu Engineering College, Perundurai have completed development of the Application for "**KUMAR MESS PRIVATE LIMITED**" with the requirements what we had given. We really satisfied with your software. We also request you to extend your support in future.

Thanking You,

Regards,

For KUMAR MESS PVT LTD


Director

BRANCHES @ MADURAI
TALLAKULAM - TOWNHALL ROAD - MATTUTHAVANI - BY PASS ROAD

BRANCHES @ CHENNAI
VADAPALANI - TAMBARAM RLY. STATION - T. NAGAR