MINI PROJECT REPORT
on

GITZZERIA

Submitted by

KAVYA RAJ.P -MGP21UCS087

NOEL MATHEWS -MGP21UCS118

PARVATHY.S -MGP21UCS119

To APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

Department of Computer Science and Engineering
Saintgits College of Engineering (Autonomous)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
June 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SAINTGITS COLLEGE OF ENGINEERING (Autonomous)

2023-2024
CERTIFICATE

Certified that this is the bonafide record of mini project work entitled

GITZZERIA

Submitted by

KAVYA RAJ.P -MGP21UCS087

NOEL MATHEWS -MGP21UCS118

PARVATHY.S -MGP21UCS119

Under the guidance
of
Er. HARI.M

In partial fulfilment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering under the APJ Abdul Kalam Technological University during the year 2023-2024.

HEAD OF DEPARTMENT	PROJECT COORDINATOR	PROJECT GUIDE
Dr. Arun Madhu	Dr.Reni K Cherian	Er.Hari.M
	Er.Sania Thomas	

DECLARATION

We undersigned hereby declare that the project report ‘GITZZERIA’ submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by our under supervision of Er.Hari.M. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place:	Name of the students:
Date:	Signature:

ACKNOWLEDGEMENT

We express our gratitude to **Dr. Sudha T**, Principal, Saintgits College of Engineering for providing with excellent ambiance that laid potentially strong foundation for this work.

We express our heartfelt thanks to **Dr. Arun Madhu**, Head of the Department of Computer Science and Engineering, Saintgits College of Engineering who has been a constant support in every step of our seminar and the source of strength in completing this mini project.

We express our sincere thanks to **Er.Hari.M**, Computer Science and Engineering Department for providing with all the facilities, valuable and timely suggestions and constant supervision for the successful completion of my mini project.

We are highly indebted to project coordinators, **Dr.Reni K Cherian** and **Er.Sania Thomas** and all the other faculties of the department for their valuable guidance and instant help and for being with us. We extend our heartfelt thanks to our parents, friends and well-wishers for their support and timely help.

Last but not the least we thank Almighty God for helping us in successfully completing this mini project.

ABSTRACT

The conventional operations of a Cafeteria lead to shortfall in efficiency and convenience, creating long queues, manual payment processes, and limited customer engagement. To address these issues, there is a need to digitalize the canteen system, offering a seamless and cashless experience to customers while optimizing operations for the management. Therefore, the digitalization of the canteen system will help the management to provide a best-in-class service to the customers and which will also contribute in better time management.

In this proposed platform, we aim to provide a cashless counter by designing a user-friendly interface that is intuitive and visually appealing, considering the target audience of college students and faculty. Initially the user must login to the platform. A digital menu interface will be provided from which the customer can choose the desired item and proceed to payment. Also user can schedule the order whenever needed. Virtual queue management system is also implemented allowing users to place orders in advance, and receive notifications when their orders are ready and integrate secure payment gateways to facilitate cashless transactions, ensuring compliance with relevant security standards. At last, the customers can also share their feedback and queries. Reward system is also implemented to encourage the use of the platform. Thus the online system will be helpful for both customers and the canteen.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Our proposed platform is aimed to revolutionize the dining experience within college communities, offering a dynamic solution tailored to meet the evolving demands of modern campus life by seamlessly integrating cutting-edge technology and innovative features.

Key among these features is the ability for users to schedule their orders in advance, providing unparalleled convenience and flexibility to students and faculty members juggling busy schedules. With just a few clicks, users can plan their meals ahead of time, ensuring a seamless dining experience without the hassle of long wait times or last-minute rushes.

Furthermore, our platform prioritizes security and convenience through the implementation of secure payment gateways, facilitating cashless transactions that not only enhance user safety but also streamline the checkout process.

In addition to order scheduling and secure payments, our platform boasts advanced queue management systems that optimize order fulfillment and minimize wait times. Through virtual queue management, users receive real-time updates on the status of their orders, allowing for efficient pickup and a smoother overall dining experience.

Also, we've incorporated interactive feedback channels, enabling users to share their thoughts and experiences directly with canteen management. This two-way communication fosters a sense of collaboration and ensures that the platform continues to evolve and improve based on user input.

With a focus on convenience, efficiency, and innovation, we're confident that our platform will redefine the dining experience for students, faculty, and staff alike, ushering in a new era of campus dining excellence.

1.1 Project Objective

Our project aims to develop a cashless canteen platform that revolutionizes the dining experience within college communities. By integrating features such as order scheduling, secure payment gateways, virtual queue management, and interactive feedback channels, we seek to streamline operations, minimize wait times, and enhance overall user satisfaction. Our objective is to provide a seamless and convenient dining experience for students, faculty, and staff while optimizing efficiency for canteen management.

1.2 Project Scope

The scope of our project encompasses the development and implementation of a comprehensive cashless canteen platform tailored to meet the needs of college communities. Key features include order scheduling, secure payment gateways, virtual queue management, and interactive feedback channels. Our focus is on enhancing the dining experience for students, faculty, and staff while optimizing operational efficiency for canteen management. By embracing technology and innovation, we aim to address common pain points in traditional dining settings and provide a seamless, convenient, and user-friendly platform for all stakeholders involved.

1.3 Project Overview

Our project involves creating a cutting-edge cashless canteen platform for college communities. With features like order scheduling, secure payments, virtual queue management, and interactive feedback channels, we aim to revolutionize the dining experience. Our platform focuses on enhancing convenience, efficiency, and user satisfaction while streamlining operations for canteen management. Through innovative technology and user-centric design, we aspire to set a new standard for campus dining.

CHAPTER 2

LITERATURE REVIEW

The evolution of technology has revolutionized various aspects of our lives, and the dining experience within educational institutions is no exception. Campus canteens, once characterized by long queues, inefficient order processing, and manual errors, are now witnessing a paradigm shift towards the integration of mobile computing and automation technologies. This comprehensive review explores the burgeoning field of campus canteen management systems using mobile computing, delving into existing research, key insights, emerging trends, and technological advancements. By analysing a diverse range of studies, this review aims to provide a comprehensive understanding of how mobile computing is reshaping the landscape of campus dining, with a focus on enhancing efficiency, improving user experience, and optimizing operational processes.

The adoption of mobile computing in canteen management systems has emerged as a transformative force, enabling educational institutions to modernize their dining facilities and meet the evolving needs of their campus communities. Malla and Shah (2021) provide valuable insights into the potential of mobile applications in optimizing canteen operations, emphasizing the importance of user-friendly interfaces and efficient management systems. Their study underscores the role of mobile computing in facilitating seamless order placement, secure payment processing, and interactive feedback collection, thereby enhancing customer satisfaction and operational efficiency.

Additionally, Yang and Ting (2022) propose a canteen food ordering and managing system based on an Android application, aiming to streamline the ordering process and improve user convenience. Their study focuses on integrating features such as menu browsing, order customization, and real-time updates to provide a seamless and efficient dining experience for users. By leveraging mobile technology, Yang and Ting demonstrate the potential to reduce wait times, enhance order accuracy, and optimize resource utilization within campus canteens.

A user-centric approach is paramount in the design and development of campus canteen management systems, ensuring that the platform meets the diverse preferences and requirements of its users. George et al. (2020) delve into the development of a food ordering application tailored specifically for canteens, emphasizing the importance of user feedback and customization options. Their study highlights the role of personalized recommendations, dietary preferences, and order history in enhancing user experience and satisfaction. By incorporating user-centric design principles, George et al. aim to create a platform that fosters engagement, promotes loyalty, and exceeds user expectations.

Furthermore, Monik Shah et al. (2018) stress the benefits of automation in improving efficiency and reducing manual errors in canteen operations. Their study explores the implementation of a comprehensive canteen automation system, integrating features such as inventory management, order tracking, and reporting tools. By automating repetitive tasks and streamlining operational processes, Monik Shah et al. demonstrate the potential to enhance productivity, minimize costs, and optimize resource allocation within campus canteens.

Automation lies at the heart of modern campus canteen management systems, driving efficiency, accuracy, and operational excellence. Gowthami et al. (2020) present a mobile application for canteen automation system using Android, focusing on providing convenient access to canteen services and enhancing user engagement. Their study underscores the role of mobile technology in automating repetitive tasks such as order processing and payment handling, thereby improving efficiency and reducing operational costs. By leveraging automation, Gowthami et al. aim to create a platform that optimizes resource utilization, enhances operational performance, and meets the growing demands of campus dining.

Additionally, the integration of mobile computing and automation technologies enables educational institutions to gain valuable insights into user behaviour, preferences, and trends. By analysing data collected through mobile applications, canteen management can identify patterns, anticipate demand, and tailor offerings to meet the specific needs of their campus communities. This data-driven approach empowers educational institutions to make informed decisions, enhance customer satisfaction, and drive innovation within campus dining.

The integration of mobile computing and automation technologies is revolutionizing campus canteen management, offering a wide array of benefits including enhanced efficiency, improved user experience, and optimized operational processes. From streamlining order placement to automating payment processing, mobile applications are reshaping the way students, faculty, and staff interact with dining facilities within educational institutions. As technology continues to evolve, there is immense potential for further innovation and advancement in the field of campus canteen management. By embracing mobile computing and automation, educational institutions can create dining experiences that are seamless, convenient, and tailored to the needs of their campus communities.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Feasibility Study

The feasibility study for the proposed cashless canteen platform demonstrates its viability across technical, operational, economic, and legal dimensions. Technically, the required technologies and expertise are readily available, ensuring seamless development and integration. Operationally, the platform's user-friendly design and accessibility features make it suitable for tech-savvy college students, faculty, and staff, with management support achievable through engagement and training. Economically, the potential for increased efficiency, cost savings, and additional revenue streams outweighs the initial and ongoing costs, promising a favourable return on investment. Legally, compliance with data protection and payment regulations is manageable. Overall, the study indicates that the project is practical and beneficial, warranting further development and implementation.

3.2 Software Requirements

3.2.1 Frontend Development

- HTML (Hypertext Markup Language)
  HTML will be used to structure the web pages of the cashless canteen platform. This includes defining the layout and elements of each page, such as headers, footers, forms, buttons, and navigation bars. HTML semantic elements improve the accessibility and SEO of the platform, ensuring that users and search engines can easily navigate and understand the content.

- CSS (Cascading Style Sheets)
  CSS will be employed to style the web pages, ensuring a visually appealing and responsive design. Modern CSS features such as Flexbox and Grid will be utilized to create a layout that adapts to various screen sizes, enhancing the user experience on both desktop and mobile devices. CSS preprocessors like SASS or LESS can be used to streamline and manage styles efficiently.

- JavaScript
  JavaScript, will be used to add interactivity to the web pages. This includes handling dynamic content updates, form validations, and other interactive features. Modern JavaScript syntax, such as arrow functions, destructuring, template literals, and async/await, will be employed to write clean and efficient code.

- ReactVite
  React will be the primary library for building a dynamic and responsive user interface Vite will be used as the build tool and development server for the React application. It offers fast build times and hot module replacement, significantly improving the development experience. Vite's optimized build process ensures that the application is production-ready with minimal overhead.

3.2.2 Backend and Database

Firebase will provide the backend services and real-time database functionalities. Securely manage user login and registration processes using Firebase Authentication. Server less functions to handle backend logic such as order processing and virtual queue management. Send push notifications to users about their order status and updates.

3.2.3 Payment Integration

The Razorpay API will facilitate secure and reliable payment processing. It will be integrated into the platform to handle various payment methods, including credit/debit cards, UPI, and net banking. Razorpay's features for creating and managing orders, handling payment transactions, and processing callbacks ensure a seamless and secure checkout experience for users.

3.2.4 Development and Deployment Tools

Git will be used for version control, with repositories hosted on platforms like GitHub, GitLab, or Bitbucket. Version control ensures that all changes are tracked, facilitating collaboration and preventing code conflicts.

3.2.5 Development Environment

Node.js and npm (Node Package Manager) will be essential for managing project dependencies and running development servers. Node.js provides a runtime environment for server-side JavaScript, while npm handles the installation and management of libraries and packages.

3.2.6 Code Editor

Visual Studio Code will be the primary integrated development environments (IDEs) for coding. These editors offer extensive support for JavaScript, React, and Firebase development, including syntax highlighting, debugging, and version control integration.

3.3 Hardware Requirement

3.3.1 Development Machines

For the development team, high-performance computer or laptops are essential. These machines should have multi-core processors (e.g., Intel i5/i7), at least 16GB of RAM, and SSD storage to handle code compilation, running development servers, and multitasking efficiently.

3.3.2 Servers

Cloud-based servers are needed to host the web application, backend services, and databases. Services like AWS, Google Cloud, or Azure provide scalable and reliable infrastructure.

3.3.3 Database Servers

Cloud Firestore will be utilized as the database server. Firestore, being a fully managed NoSQL database, eliminates the need for physical database servers.

3.3.4 Networking Equipment

A high-speed and reliable internet connection is crucial for both development and deployment environments. This ensures smooth communication with cloud servers, efficient data transfer, and quick access to online development tools and resources.

3.3.5 Backup and Storage Solutions

Cloud storage solutions, such as Google Cloud Storage or AWS S3, are necessary for storing backups of the application data, user information, and other critical assets.

3.3.6 Security Hardware

To protect the platform from cyber threats, hardware firewalls and security appliances are needed. Ensuring robust security hardware is crucial for maintaining the integrity and confidentiality of user data.

3.3.7 User Access Devices

For the canteen staff, user access devices like kiosks or tablets are required to manage orders and interact with the platform.

3.3.8 Applications

1. Campus Dining: The primary application of this platform is within college campuses, providing students, faculty, and staff with a seamless dining experience. It allows users to browse menus, place orders, and make payments without using cash.
2. Event Management: During college events or gatherings, the platform can be used to manage food orders efficiently, avoiding long queues and ensuring timely service.
3. Cafeterias in Corporates and Institutions: Beyond educational institutions, this platform can be adapted for use in corporate cafeterias or other institutional dining facilities, offering the same benefits of cashless transactions and order management.
4. Food Courts: The platform can be implemented in food courts within malls or public spaces, where multiple vendors can manage their orders and payments digitally.
5. Mobile Ordering Services: The application can be extended to support mobile ordering services for various dining establishments, providing a convenient ordering system for on-the-go customers.

3.3.9 Advantages

1. Convenience: Users can easily browse menus, place orders, and make payments from their mobile devices or computers, eliminating the need to carry cash.
2. Efficiency: Streamlined order processing and virtual queue management reduce wait times and improve the overall dining experience.
3. Real-Time Updates: Users receive real-time notifications about their order status, ensuring they are informed when their food is ready.
4. Enhanced Security: Secure payment gateways like Razorpay ensure safe transactions, protecting user data and financial information.
5. Feedback and Improvement: The integrated feedback system allows users to share their dining experience, helping canteen management to continuously improve service quality.
6. Operational Optimization: For canteen management, the platform offers insights into order patterns, peak hours, and inventory needs, aiding in better resource management.
7. Sustainability: Reduces the need for paper receipts and menus, contributing to environmental sustainability.

3.3.10 Disadvantages

1. Dependency on Technology: The platform's effectiveness is highly dependent on the availability of technology. Any downtime in the system or issues with internet connectivity can disrupt service.
2. Initial Setup Cost: Implementing the platform requires an initial investment in technology infrastructure, training, and integration with existing systems.
3. User Adaptability: Some users, especially those not tech-savvy, may find it challenging to adapt to a digital system, potentially causing frustration or resistance.
4. Security Risks: Despite secure payment gateways, the platform is still susceptible to cyber threats. Continuous monitoring and updates are necessary to mitigate such risks.
5. Maintenance and Updates: Ongoing maintenance and regular updates are required to keep the platform running smoothly and securely, which can be resource-intensive.
6. Potential Over-Reliance on Digital Payments: In cases where digital payment systems fail or users do not have access to digital payment methods, the lack of a cash option could be problematic.

7. Privacy Concerns: Handling user data requires stringent compliance with privacy laws and regulations to protect user information and maintain trust.

3.4 General Requirements

3.4.1 User Authentication and Authorization

To ensure secure access, the platform will use Firebase Authentication, supporting various methods such as email/password, Google, and other social logins. This functionality will safeguard user data and provide personalized experiences by distinguishing between different user roles (e.g., students, faculty, and canteen staff).

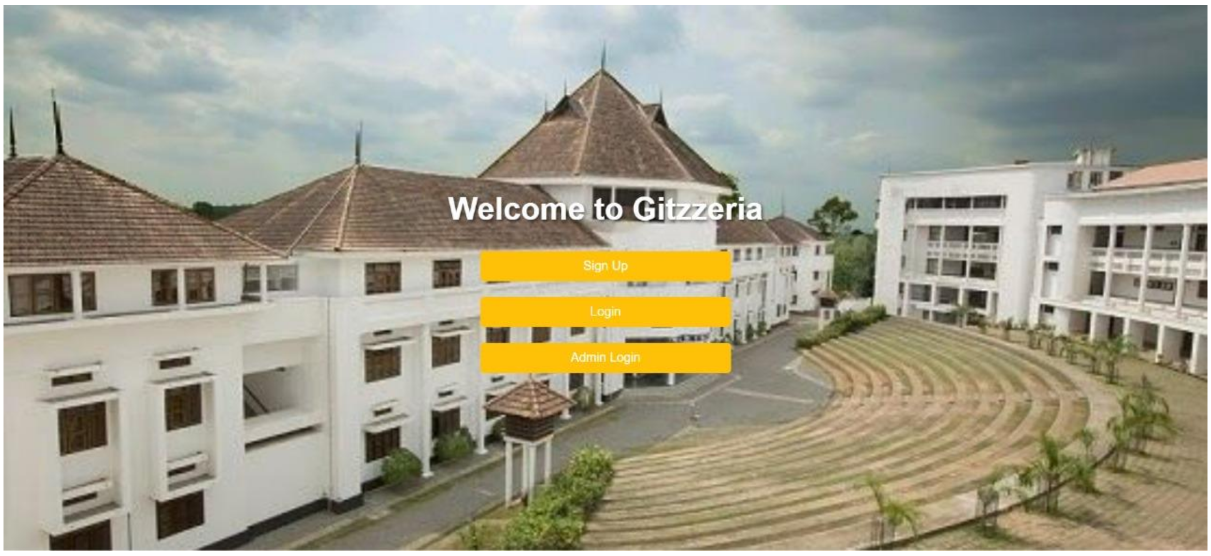


Fig 3.4.1 Landing page

3.4.2 Responsive User Interface

The platform must have a responsive design to provide a seamless user experience across various devices, including desktops, tablets, and smartphones. Using React and CSS frameworks like Bootstrap or Tailwind CSS, the platform will adapt to different screen sizes and orientations, ensuring accessibility and ease of use for all users.

3.4.3 Digital Menu and Ordering System

An intuitive digital menu interface will allow users to browse, select, and customize their orders efficiently by providing detailed information about menu items, including ingredients, prices, and availability.

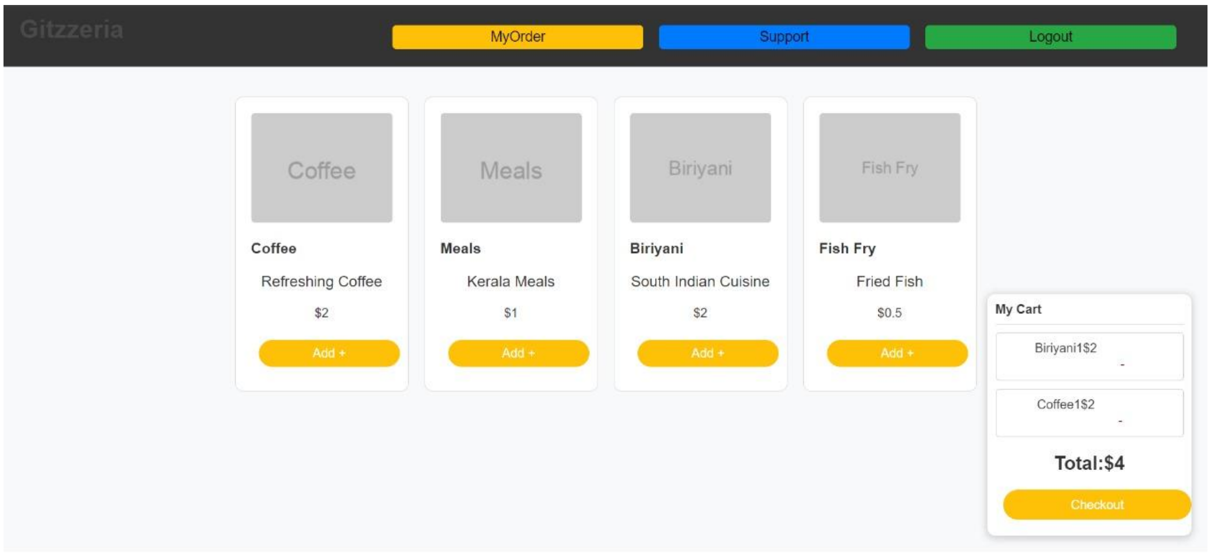


Fig 3.4.3 Menu page

3.4.4 Order Scheduling and Virtual Queue Management

Users will have the ability to schedule their orders in advance and join a virtual queue, which will be managed in real-time using Firebase Firestore.

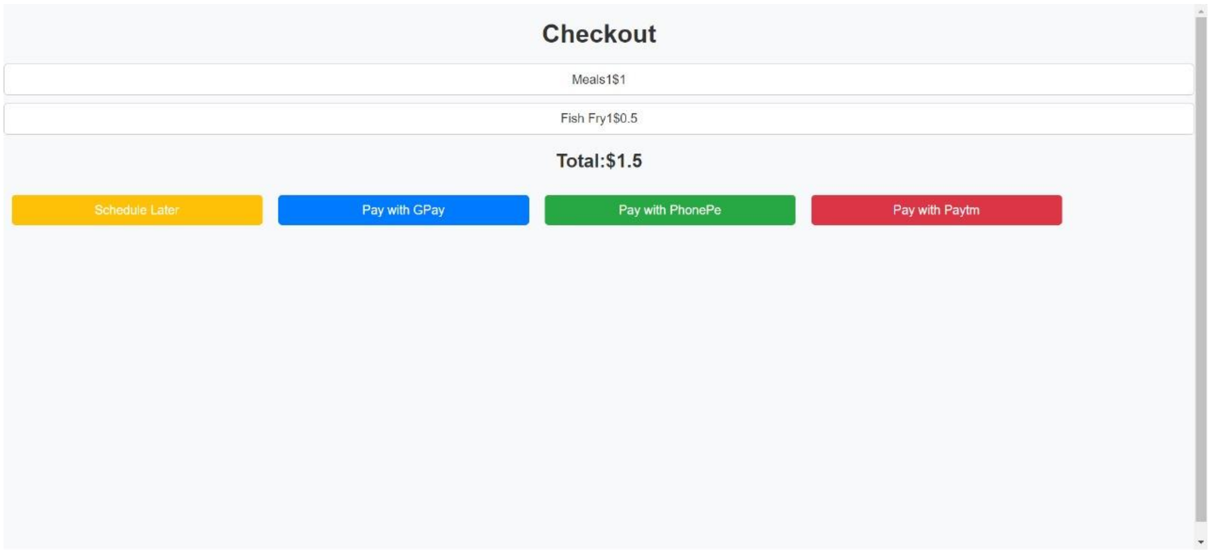


Fig 3.4.4 Order scheduling

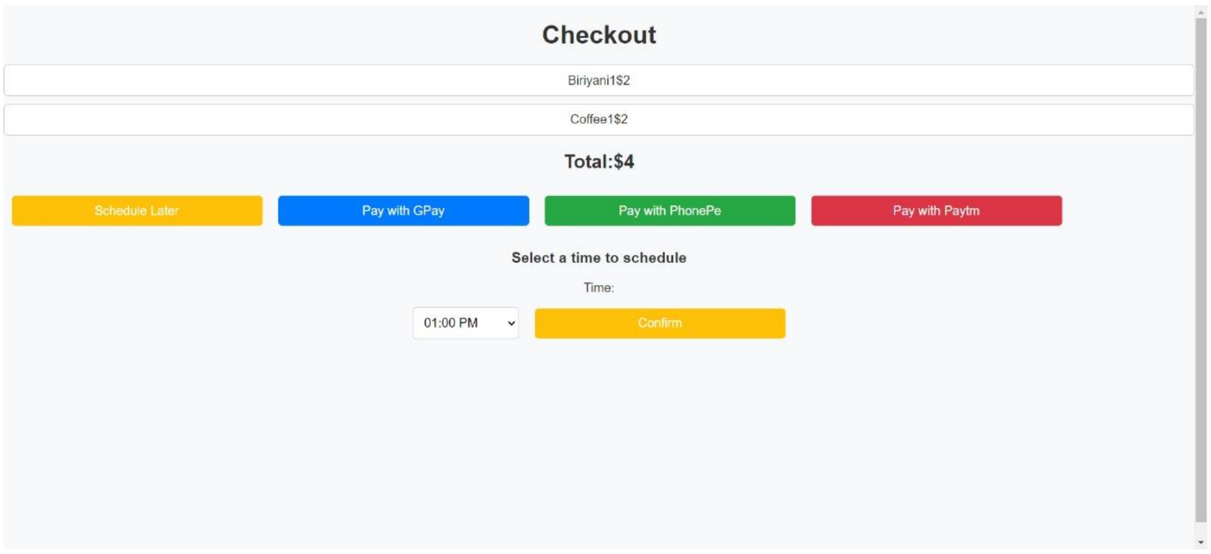


Fig 3.4.3 Scheduled order

3.4.5 Secure Payment Gateway Integration

The platform will integrate the Razorpay API to handle cashless transactions securely and supports multiple payment methods, ensuring flexibility and convenience for users.

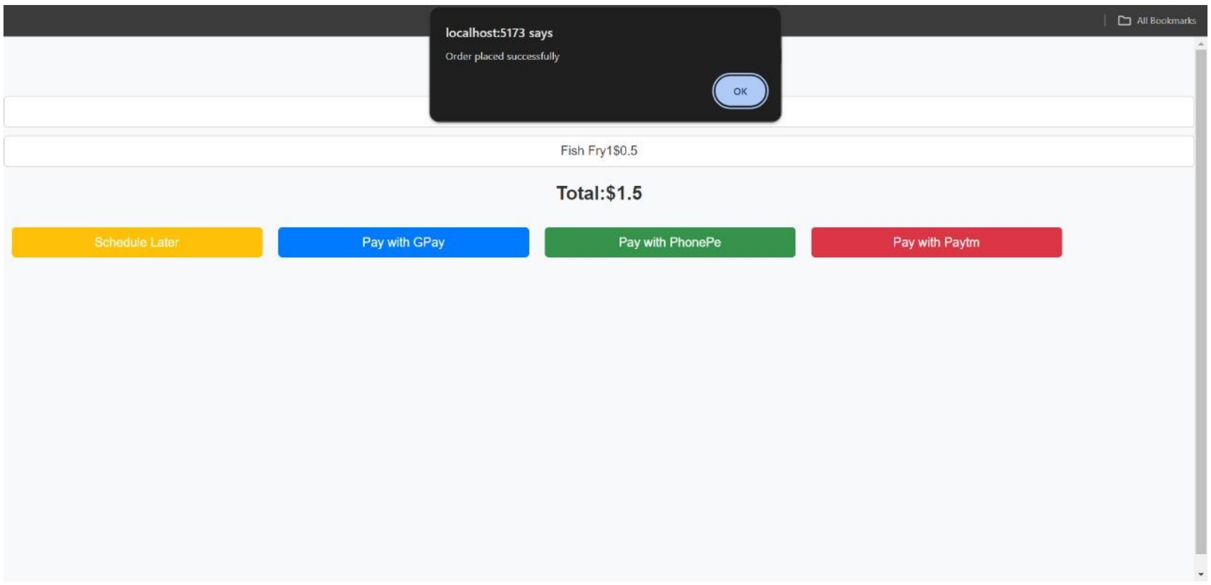
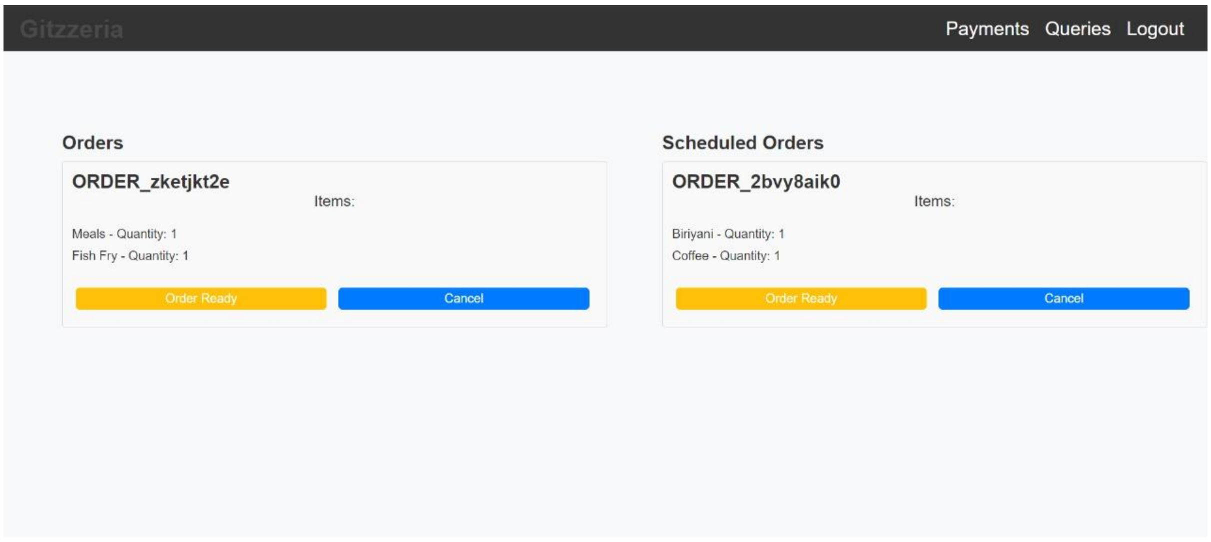


Fig 3.4.5 Payment

3.4.6 Admin Dashboard for Canteen Management

A comprehensive admin dashboard will be developed to allow canteen staff to manage menu items, view orders, update order statuses, and generate reports also enabling staff to efficiently handle daily tasks and make data-driven decisions.



Orders

ORDER_zketjkt2e

Items:

Meals - Quantity: 1

Fish Fry - Quantity: 1

Scheduled Orders

ORDER_2bvy8aik0

Items:

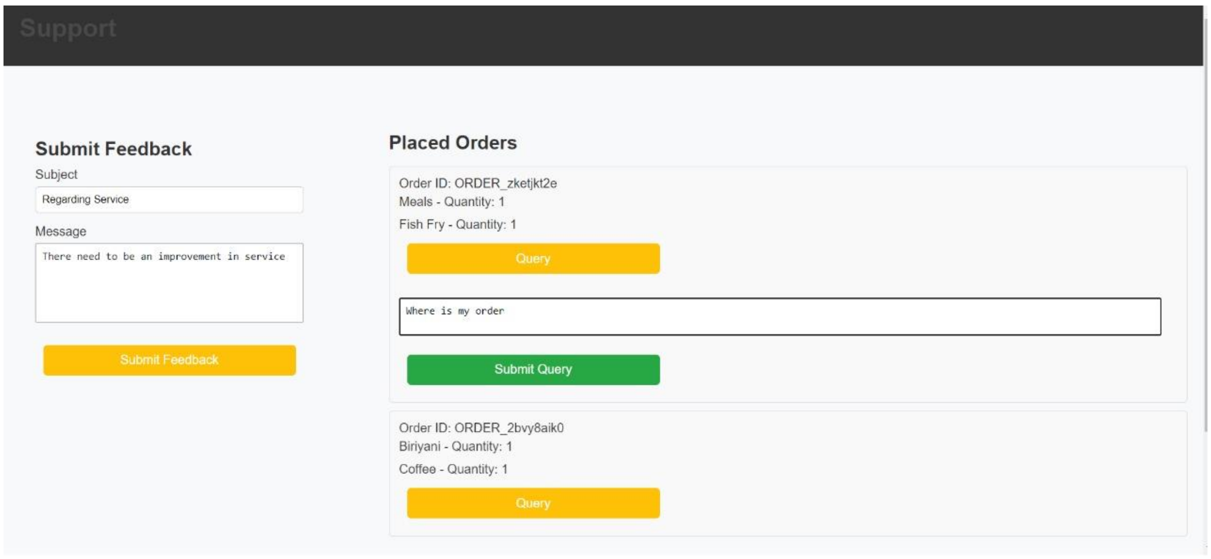
Biriyani - Quantity: 1

Coffee - Quantity: 1

Fig 3.4.6 Admin dashboard

3.4.7 Customer Feedback and Support

The platform will include a feature for customers to provide feedback and submit queries. This feedback loop will help improve service quality and customer satisfaction.



Submit Feedback

Subject: Regarding Service

Message: There need to be an improvement in service

Placed Orders

Order ID: ORDER_zketjkt2e
Meals - Quantity: 1
Fish Fry - Quantity: 1

Where is my order

Order ID: ORDER_2bvy8aik0
Biriyani - Quantity: 1
Coffee - Quantity: 1

Fig 3.4.7 Feedback and Support

3.4.8 Scalability and Performance

The system will be designed to handle high traffic and a large number of simultaneous users. Utilizing Firebase’s scalable infrastructure and Vite’s efficient build process, the platform will ensure fast load times and a smooth user experience even during peak usage.

CHAPTER 4

DESIGN

In the modern campus environment, efficiency and convenience are paramount, especially in high-traffic areas such as dining facilities. The proposed cashless counter system aims to revolutionize the dining experience for college students, faculty, and staff by providing a seamless, user-friendly platform for managing orders and payments. The system is designed to address common challenges faced in traditional dining settings, such as long queues, cash handling, and lack of real-time order tracking.

4.1 Use Case Diagram

This use case diagram outlines the interactions between different user roles—students, faculty, staff, and administrators and the system's functionalities. By leveraging digital technologies, the system ensures a streamlined operation, from order placement to payment processing and feedback collection.

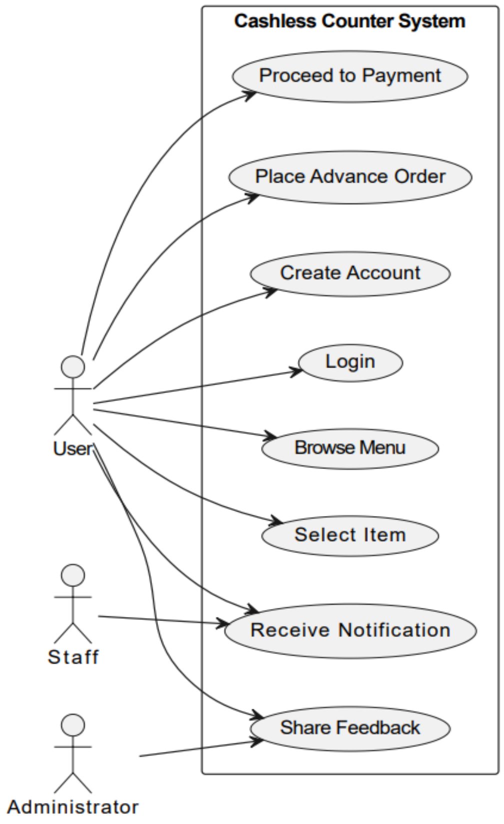


Fig.4.1. Gitzzeria use case diagram

1. Create Account:

Users and staff create their accounts to access the system. This involves entering personal information such as name, email, and password.

2. Login:

Users and staff log in to the system using their credentials to access the services provided by the cashless counter system.

3. Browse Menu:

Users can browse through the available menu items. This includes viewing different categories, item details, and prices.

4. Select Item:

Users select the items they wish to order from the menu. They can add these items to their cart.

5. Proceed to Payment:

Users proceed to pay for the items they have selected. This involves entering payment details and confirming the transaction.

6. Place Advance Order:

Users can place orders in advance, specifying a preferred time for order preparation and pickup. This helps manage queues and reduces wait times.

7. Receive Notification:

Users receive notifications about their order status, such as confirmation, preparation, and readiness for pickup. This keeps users informed and enhances their experience.

8. Share Feedback:

Users can share their feedback about the services, menu items, and overall experience. This feedback is valuable for continuous improvement.

9. Manage Users:

Administrators manage user accounts, including registration approvals, account updates, and handling any issues related to user accounts.

10. Update Menu:

Administrators and staff update the menu by adding new items, modifying existing items, and removing outdated items. This ensures that the menu is always current and accurate.

11. Process Orders:

Staff members process the orders received, including preparation and ensuring timely delivery or pickup. They also manage the order queue.

12. View Feedback:

Administrators and staff can view the feedback provided by users to make necessary improvements in the services and menu items.

4.2 Class Diagram

The class diagram for the cashless counter system provides a detailed blueprint of the system's architecture, defining the various classes, their attributes, methods, and the relationships between them. This diagram serves as a foundational component in the system design, illustrating how different parts of the system interact with each other to deliver the required functionalities.

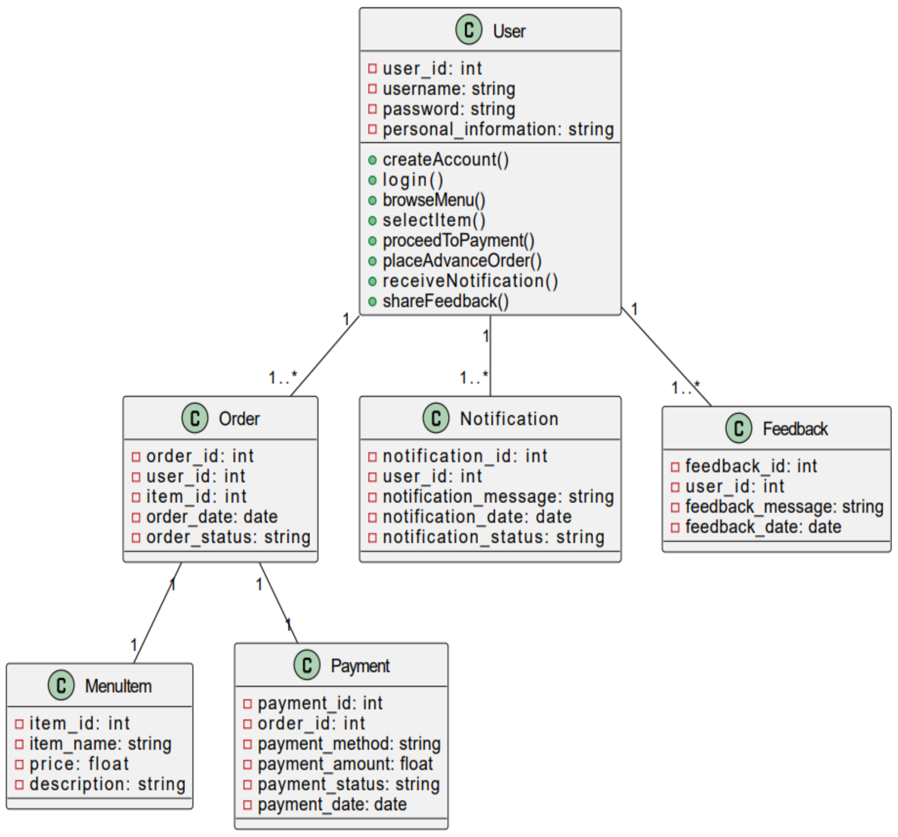


Fig.4.2. Gitzzeria Class diagram

Key Components of the Class Diagram:

1. User:

- Attributes:

  - user_id: Unique identifier for each user.

  - username: The user's login name.

- password: The user's login password.

- Methods:
  - createAccount(): Allows users to create an account.
  - login(): Authenticates users for system access.
  - browseMenu(): Enables users to view available menu items.
  - selectItem(): Allows users to choose items from the menu.
  - proceedToPayment(): Facilitates the payment process.
  - placeOrder(): Enables users to place orders in advance.
  - receiveNotification(): Notifies users about order status.
  - provideFeedback(): Allows users to provide feedback on their experience.

2. MenuItem:

- Attributes:
  - item_id: Unique identifier for each menu item.
  - item_name: Name of the menu item.
  - price: Price of the menu item.
  - description: Detailed description of the menu item.

3. Order:

- Attributes:
  - order_id: Unique identifier for each order.
  - user_id: Identifier linking the order to the user.
  - item_id: Identifier linking the order to the menu item.
  - order_date: Date when the order was placed.
  - order_status: Current status of the order (e.g., pending, completed).

4. Payment:

- Attributes:
  - payment_id: Unique identifier for each payment transaction.
  - order_id: Identifier linking the payment to the order.

- payment_method: Method of payment (e.g., credit card, PayPal).
- payment_amount: Amount paid for the order.
- payment_status: Status of the payment (e.g., successful, failed).
- payment_date: Date when the payment was made.

5. Notification:

- Attributes:
  - notification_id: Unique identifier for each notification.
  - user_id: Identifier linking the notification to the user.
  - notification_message: Content of the notification message.
  - notification_date: Date when the notification was sent.
  - notification_status: Status of the notification (e.g., read, unread).

6. Feedback:

- Attributes:
  - feedback_id: Unique identifier for each feedback entry.
  - user_id: Identifier linking the feedback to the user.
  - feedback_message: Content of the feedback message.
  - feedback_date: Date when the feedback was submitted.

4.3 Sequence Diagram

The sequence diagram for the cashless counter system provides a dynamic representation of the interactions between users and the system over time. This sequence diagram focuses on critical user actions, such as account creation, logging in, browsing the menu, placing orders, making payments, receiving notifications, and providing feedback. By visualizing these processes, the diagram helps in understanding the system's behaviour and identifying any potential issues in the workflow.

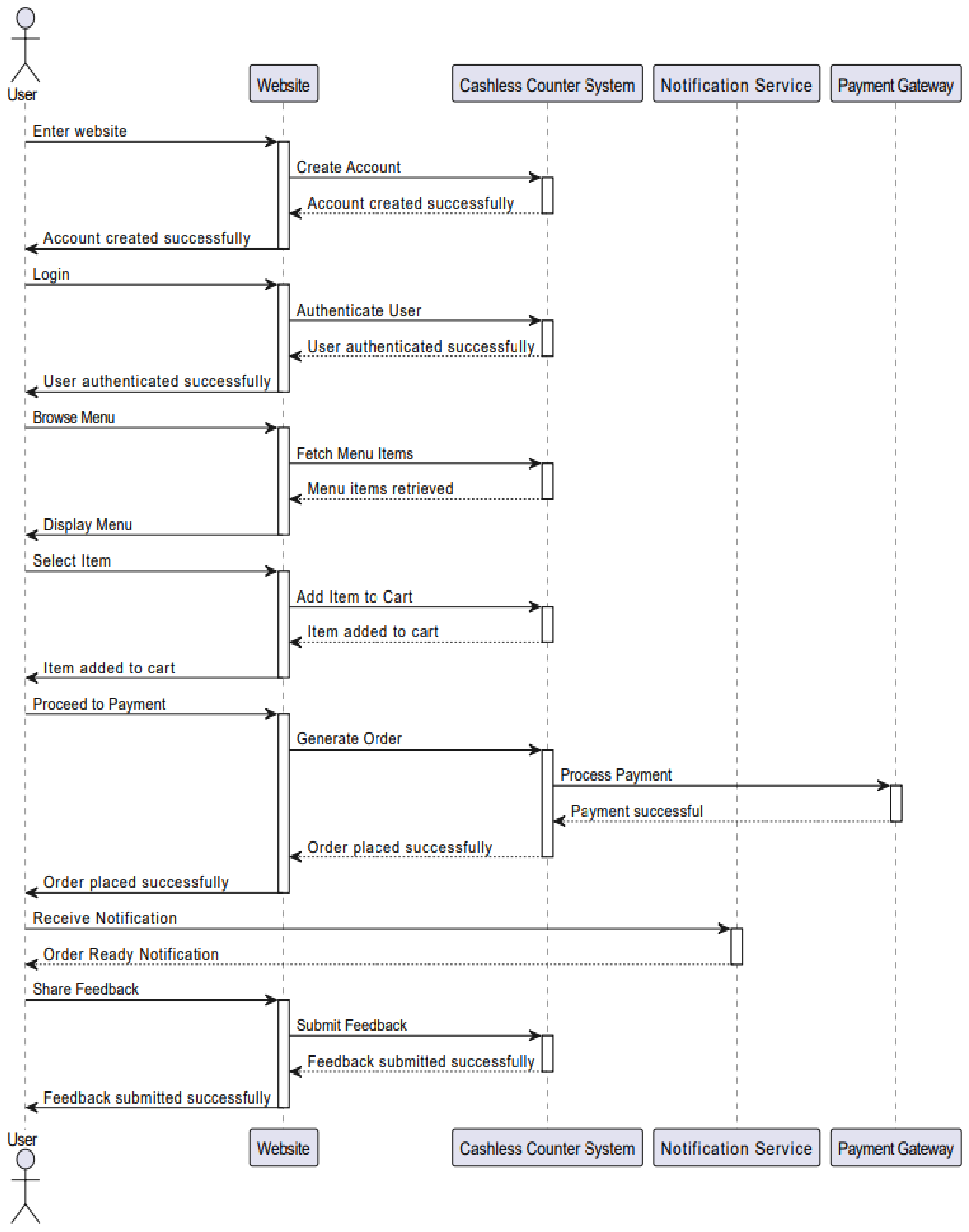


Fig.4.3. Gitzzeria Sequence Diagram

Sequence of Interactions:

1. **Enter Website**:

   - User Action: The user navigates to the website.

2. **Create Account**:

- User Action: The user initiates account creation by providing personal details.
- Website Action: The website sends account creation details to the cashless counter system.
- System Response: The system creates the account and confirms the creation to the user.

3. **Login**:
   - User Action: The user logs in using their credentials.
   - Website Action: The website forwards the login credentials to the cashless counter system.
   - System Response: The system authenticates the user and sends a success message back to the website.
4. **Browse Menu:**
   - User Action: The user browses the menu to view available items.
   - Website Action: The website requests menu items from the cashless counter system.
   - System Response: The system retrieves and sends the menu items to the website, which then displays them to the user.
5. **Select Item**:
   - User Action: The user selects an item to add to their cart.
   - Website Action: The website adds the item to the user's cart.
6. **Proceed to Payment**:
   - User Action: The user proceeds to make a payment for the selected items.
   - Website Action: The website generates an order and sends payment details to the payment gateway.
   - Payment Gateway Response: The payment gateway processes the payment and confirms success to the cashless counter system.
   - System Response: The cashless counter system confirms the order placement and sends a success message back to the website.
7. **Receive Notification**:
   - System Action: The cashless counter system sends an order ready notification to the user via the notification service.
8. **Share Feedback**:
   - User Action: The user submits feedback on their experience.
   - Website Action: The website forwards the feedback to the cashless counter system.

- System Response: The system processes the feedback and sends a success message back to the user.

4.4 Structural Architecture

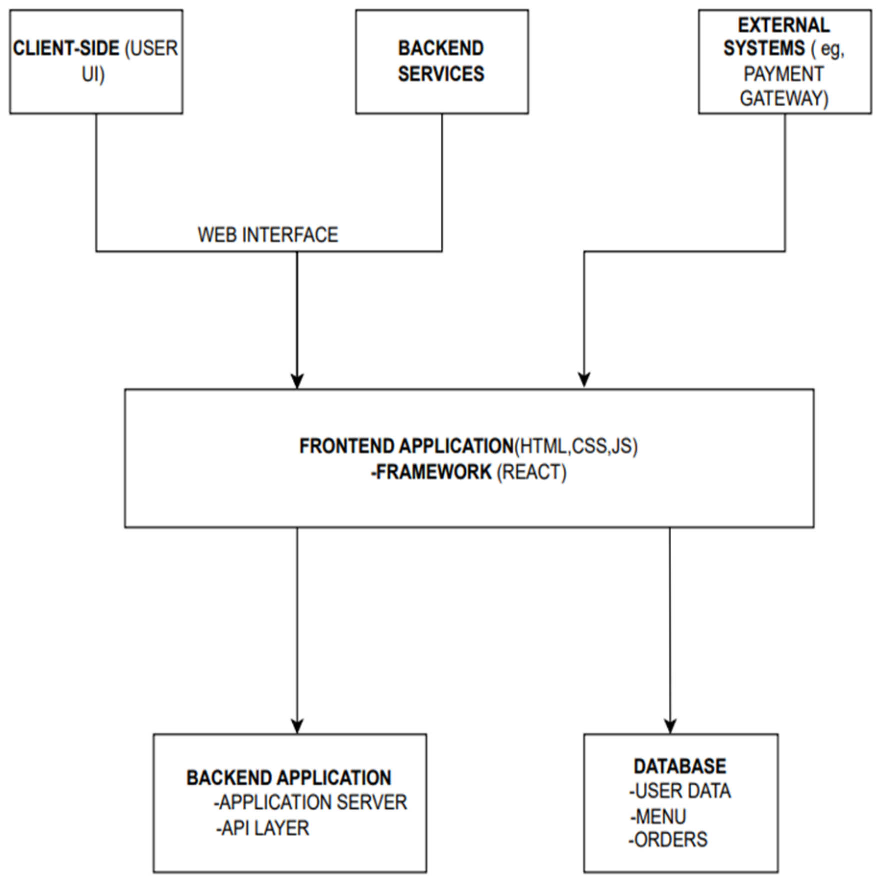


Fig.4.4. Structural Architecture

1. **Frontend Web Application**:
   - This represents the user-facing interface of the cashless counter system.
   - Entities within this package include:
     - User Registration/Login Interface: Allows users to create accounts and log in.
     - Digital Menu Interface: Displays the menu for users to browse and select items.
     - Order Placement Interface: Enables users to place orders.

- Virtual Queue Management: Manages virtual queues for order pickup.
- Feedback Submission Interface: Allows users to submit feedback about their experience.

2. **Backend Server**:
   - This represents the server-side logic and functionality of the cashless counter system.
   - Entities within this package include:
     - User Authentication and Authorization: Handles user authentication and access control.
     - Menu Management System: Manages menu items, categories, and availability.
     - Order Management System: Processes and manages orders.
     - Queue Management System: Handles virtual queue management for order pickup.
     - Feedback Management System: Collects and manages user feedback.
     - Payment Gateway Integration: Integrates with external payment gateways for transaction processing.

3. **Database**:
   - This represents the data storage component of the cashless counter system.
   - Entities within this package include:
     - User Database: Stores user account information.
     - Menu Database: Stores menu items, categories, and availability.
     - Order Database: Stores order details, including items, quantities, and status.
     - Feedback Database: Stores user feedback, ratings, and comments.

4. **Backend application**:
   - These backend applications collectively form the backbone of the cashless counter system, enabling seamless communication, efficient processing, and secure management of user interactions, orders, payments, and feedback.
   - Backend application includes application server and API layer.

5. **External Systems**;
   - An example for external systems is payment gateway.

- The Payment Gateway Integration ensures compliance with relevant security standards to protect sensitive payment information and prevent fraudulent activity.

4.5 Wireframe

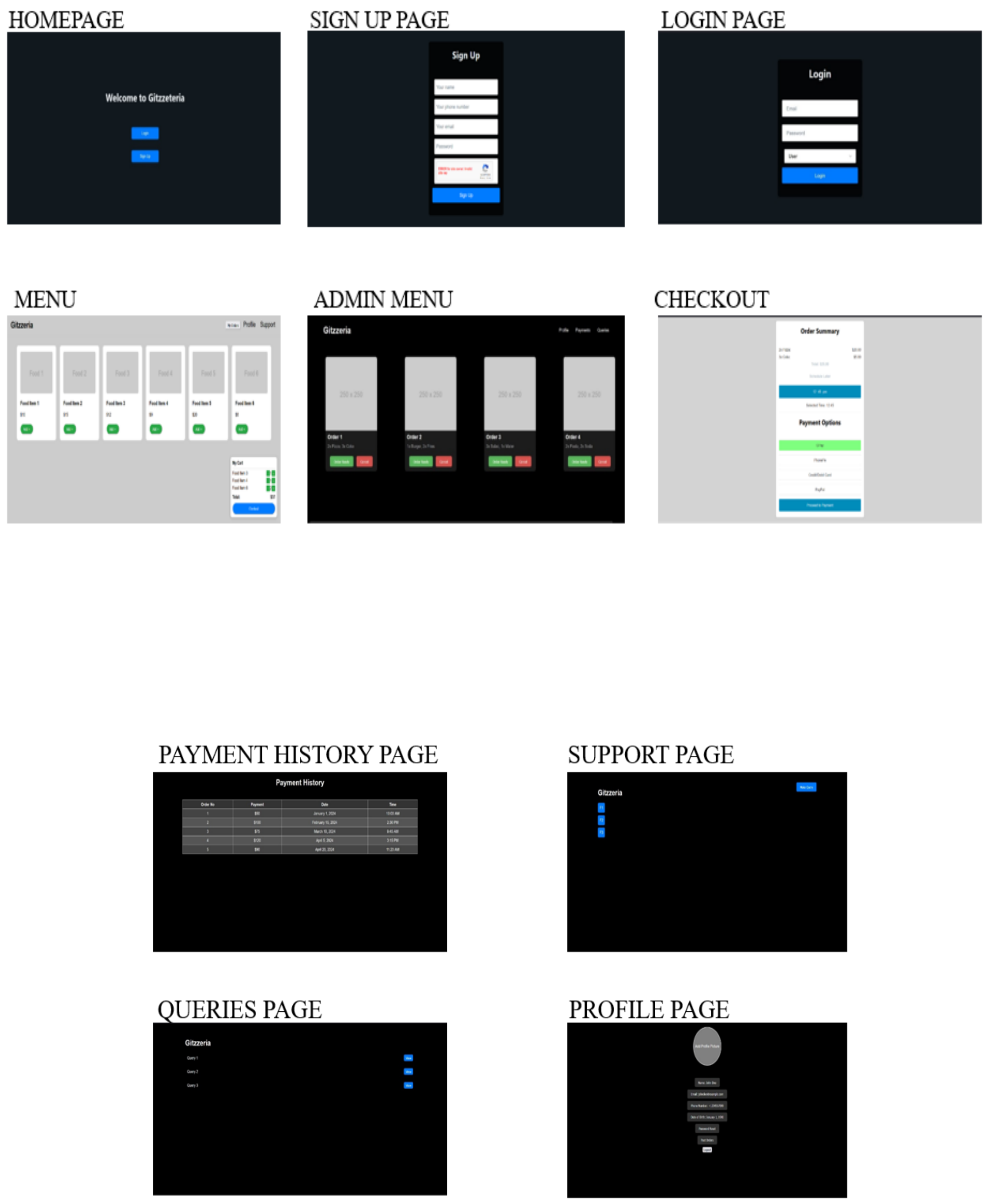
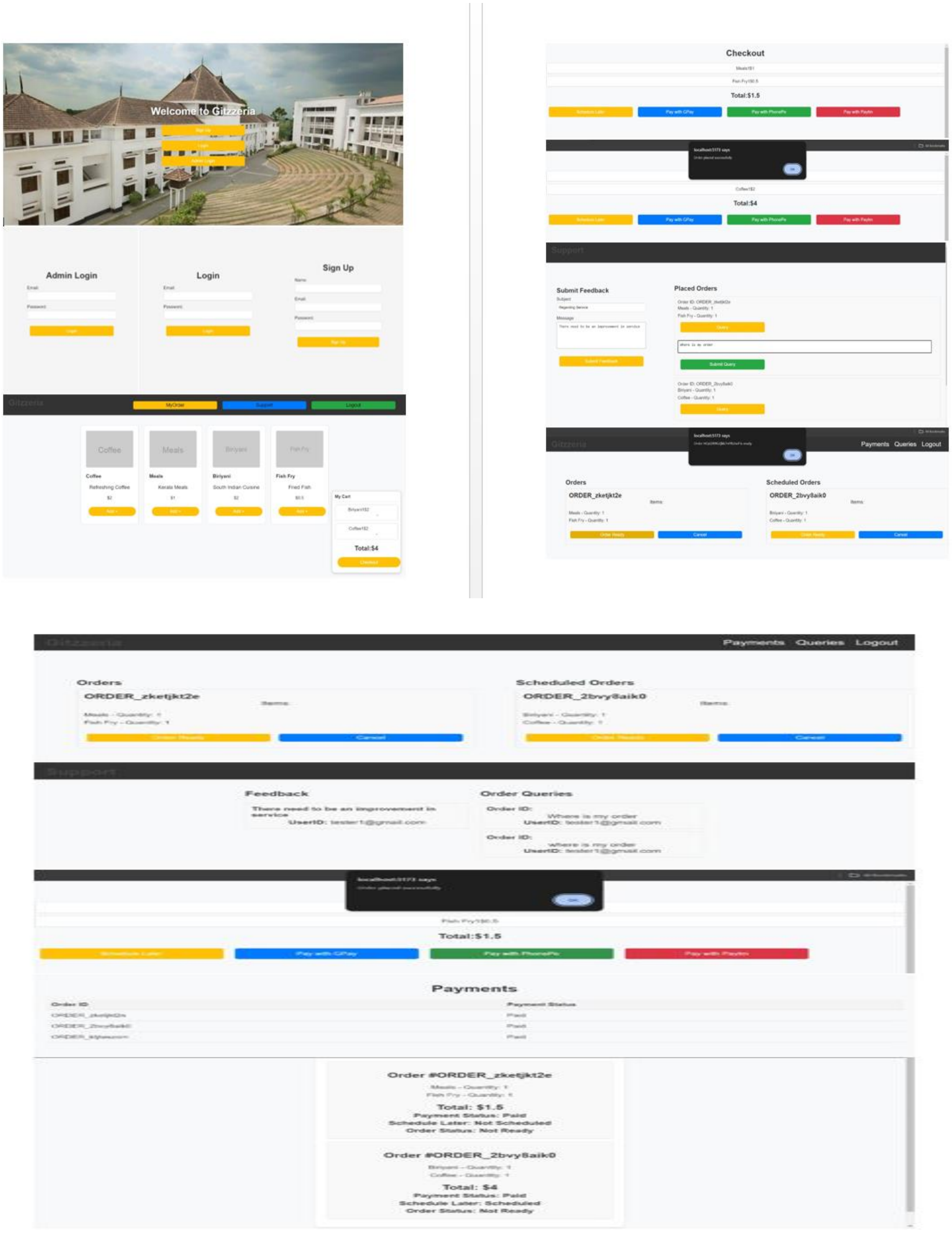


Fig 4.5 Wireframe

4.6 UI/UX



Orders

ORDER_zketjkt2e

Meals - Quantity: 1
Fish Fry - Quantity: 1

Scheduled Orders

ORDER_2bvy8aik0

Biryani - Quantity: 1
Coffee - Quantity: 1

Feedback

There need to be an improvement in service
UserID: tester1@gmail.com

Order Queries

Order ID:
Where is my order
UserID: tester1@gmail.com

Order ID:
where is my order
UserID: tester1@gmail.com

Total:$1.5

Payments

Order ID	Payment Status
ORDER_zketjkt2e	Paid
ORDER_2bvy8aik0	Paid
ORDER_[illegible]	Paid

Order #ORDER_zketjkt2e
Meals - Quantity: 1
Fish Fry - Quantity: 1
Total: $1.5
Payment Status: Paid
Schedule Later: Not Scheduled
Order Status: Not Ready

Order #ORDER_2bvy8aik0
Biryani - Quantity: 1
Coffee - Quantity: 1
Total: $4
Payment Status: Paid
Schedule Later: Scheduled
Order Status: Not Ready

Fig 4.6 UI/UX

CHAPTER 5
DEVELOPMENT

The development phase is a crucial stage in the creation of the proposed cashless counter website. This phase involves transforming the conceptual ideas and requirements into a functional and user-friendly site that can effectively revolutionize the dining experience for college students and faculty. During this phase, the foundational elements of the platform are built, integrating innovative features designed to streamline operations and enhance convenience. Steps involved are:

5.1. Requirement Analysis

- **User Needs**: Identify the needs of students, faculty, and canteen staff. This includes understanding their pain points with the current system, desired features, and user experience expectations.
- **Functional Requirements**: Define what the system should do, such as user authentication, order placement, payment processing, order tracking, and feedback collection.
- **Non-Functional Requirements**: Determine system performance, security, usability, and reliability criteria. This includes response times, data protection measures, and system availability.
- **Stakeholder Interviews**: Conduct interviews with college administration, canteen management, and potential users to gather insights and refine requirements.
- **Documentation**: Compile a comprehensive requirements document that outlines all functional and non-functional requirements, user stories, and acceptance criteria.

5.2. System Design

- **UI/UX Design**: Create wireframes and prototypes using tools like Figma. Focus on user-friendly design, intuitive navigation, and accessibility.
- **Database Design**: Design the data model and database schema using Firestore. Define collections for users, orders, menu items, feedback, and other necessary data entities.
- **API Design:** Define endpoints for user authentication, order management, and payment processing.

- **Security Design**: Plan for security features such as OAuth 2.0 for authentication, HTTPS for secure data transmission, and data encryption.

5.3. Development

- **Frontend Development:** Develop the frontend using React with Vite. Create reusable components, manage state with Context API and implement routing with React Router.
- **Backend Development**: Implement backend logic using Firebase Functions. Set up Firestore for real-time data storage and synchronization. Ensure server less functions handle business logic such as order processing and queue management.
- **Payment Integration**: Integrate Razorpay's JavaScript SDK to handle secure payment processing. Implement payment workflows, handle success and failure callbacks, and update order statuses accordingly.
- **Testing**: Write unit tests for individual components, integration tests for API endpoints, and end-to-end tests for the entire user flow. Use testing libraries like Jest, Mocha, or Cypress.

5.4. Integration and Testing

- **Continuous Integration (CI):** Set up CI pipelines using tools like GitHub Actions or Travis CI. Automate the building, testing, and deployment processes.
- **Automated Testing**: Implement automated tests to cover various aspects of the platform, including functionality, performance, and security.
- **User Acceptance Testing (UAT)**: Conduct UAT with actual users to validate that the platform meets their needs and expectations. Gather feedback and make necessary adjustments.
- **Performance Testing**: Test the platform's performance under different loads to ensure it can handle high traffic and large volumes of data. Use tools like Jmeter or LoadRunner.
- **Security Testing**: Perform security audits and penetration testing to identify and fix vulnerabilities. Ensure compliance with data protection regulations.

5.5. Deployment

- **Hosting**: Deploy the frontend application to Firebase Hosting. Configure it for fast and secure delivery.
- **Backend Deployment**: Deploy Firebase Functions for server less backend logic. Ensure they are configured for scalability and reliability.
- **Database Setup**: Set up Firestore with proper security rules and indexing to ensure efficient and secure data operations.
- **Domain Configuration**: Configure custom domains for the platform and set up SSL certificates for HTTPS.
- **Monitoring and Logging:** Implement monitoring using Firebase Analytics and Crashlytics to track user interactions and system performance. Set up logging for debugging and maintenance.

5.6. Maintenance and Updates

- **Bug Fixes**: Monitor the platform for bugs and issues reported by users. Prioritize and address these promptly.
- **Feature Enhancements**: Continuously gather user feedback and identify opportunities for feature improvements. Implement enhancements based on user needs and technological advancements.
- **Security Updates**: Regularly update dependencies and implement security patches to protect against vulnerabilities.
- **Performance Optimization**: Continuously monitor and optimize the platform's performance to ensure a smooth user experience.
- **Customer Suppor**t: Provide ongoing support to users through help desks or feedbacks. Address user queries and issues efficiently.

CHAPTER 6

TESTING & MAINTENANCE

The testing process for the proposed cashless counter website is a crucial step in ensuring that the platform delivers a seamless, secure, and user-friendly experience for college students and faculty. This process aims to rigorously evaluate the website's functionalities, performance, security, and overall usability to meet the high standards expected by its target audience.

Our primary focus is to validate that all features—ranging from user registration and digital menu navigation to secure payment processing and virtual queue management—operate flawlessly under various conditions. Through comprehensive testing, we aim to identify and address any issues that could impact the user experience or compromise security.

6.1. Testing Process

6.1.1 Functional Testing

- **User Registration and Login:**

  Verify that users can create accounts using valid personal information. Ensure that login functionality works with correct credentials and handles incorrect credentials gracefully.

- **Digital Menu Interface**:

  Test the digital menu for correct item display, category browsing, and search functionality. Ensure that item selection, quantity adjustment, and order placement features are working properly.

- **Order Placement and Payment**:

  Validate the entire order placement workflow, including item selection, cart review, and payment gateway integration. Check the secure processing of payments and handling of different payment methods.

- **Virtual Queue Management**:

  Ensure that users can place orders in advance and receive notifications when their orders are ready. Test the system's ability to manage and update the virtual queue in real-time.

- **Feedback System**:

  Verify that users can submit feedback easily and that the feedback is stored and accessible for review by management.

6.1.2. Usability Testing

- Conduct tests with real users from the target audience (college students and faculty) to gather feedback on the website's usability and interface design.
- Evaluate the intuitiveness of navigation, the clarity of information, and the overall user experience.

6.1.3. Performance Testing

- **Load Testing**: Simulate high traffic scenarios to ensure the website can handle multiple users simultaneously without performance degradation.
- **Stress Testing**: Test the website's robustness under extreme conditions to identify any breaking points or critical failures.
- **Response Time Testing**: Measure the website's response times for various actions (e.g., loading the menu, processing payments) to ensure they are within acceptable limits.

6.1.4. Security Testing

- **Vulnerability Assessment**: Conduct regular scans for common vulnerabilities (e.g., SQL injection, XSS) to ensure the website is secure.
- **Penetration Testing**: Perform manual and automated penetration tests to identify and fix security weaknesses.
- **Payment Gateway Security**: Verify that the payment gateways comply with industry security standards (e.g., PCI DSS) to protect user data.

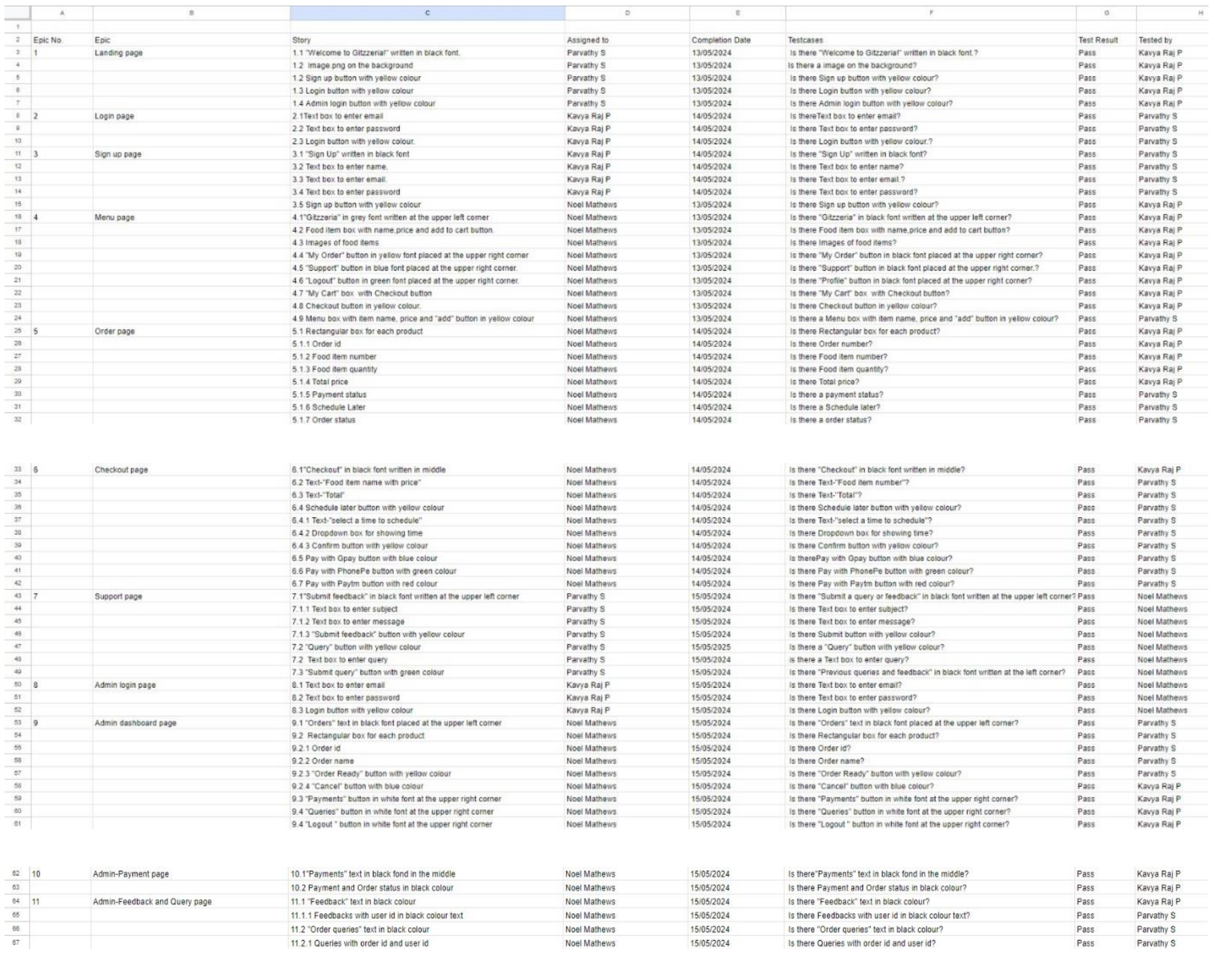
6.1.5. Compatibility Testing

- Test the website across different browsers (Chrome, Firefox, Safari, Edge) and devices (desktops, tablets, smartphones) to ensure consistent functionality and appearance.

6.1.6. Regression Testing

- After updates or bug fixes, conduct regression testing to ensure that existing functionalities are not affected.

6.1.7. Test case Report



Epic No	Epic	Story	Assigned to	Completion Date	Testcases	Test Result	Tested by
1	Landing page	1.1 "Welcome to Gitzzeria!" written in black font.	Parvathy S	13/05/2024	Is there "Welcome to Gitzzeria!" written in black font.?	Pass	Kavya Raj P
		1.2 image.png on the background	Parvathy S	13/05/2024	Is there a image on the background?	Pass	Kavya Raj P
		1.2 Sign up button with yellow colour	Parvathy S	13/05/2024	Is there Sign up button with yellow colour?	Pass	Kavya Raj P
		1.3 Login button with yellow colour	Parvathy S	13/05/2024	Is there Login button with yellow colour?	Pass	Kavya Raj P
		1.4 Admin login button with yellow colour	Parvathy S	13/05/2024	Is there Admin login button with yellow colour?	Pass	Kavya Raj P
2	Login page	2.1Text box to enter email	Kavya Raj P	14/05/2024	Is thereText box to enter email?	Pass	Parvathy S
		2.2 Text box to enter password	Kavya Raj P	14/05/2024	Is there Text box to enter password?	Pass	Parvathy S
		2.3 Login button with yellow colour.	Kavya Raj P	14/05/2024	Is there Login button with yellow colour.?	Pass	Parvathy S
3	Sign up page	3.1 "Sign Up" written in black font	Kavya Raj P	14/05/2024	Is there "Sign Up" written in black font?	Pass	Parvathy S
		3.2 Text box to enter name.	Kavya Raj P	14/05/2024	Is there Text box to enter name?	Pass	Parvathy S
		3.3 Text box to enter email.	Kavya Raj P	14/05/2024	Is there Text box to enter email.?	Pass	Parvathy S
		3.4 Text box to enter password	Kavya Raj P	14/05/2024	Is there Text box to enter password?	Pass	Parvathy S
		3.5 Sign up button with yellow colour	Noel Mathews	13/05/2024	Is there Sign up button with yellow colour?	Pass	Kavya Raj P
4	Menu page	4.1"Gitzzeria" in grey font written at the upper left corner	Noel Mathews	13/05/2024	Is there "Gitzzeria" in black font written at the upper left corner?	Pass	Kavya Raj P
		4.2 Food item box with name,price and add to cart button.	Noel Mathews	13/05/2024	Is there Food item box with name,price and add to cart button?	Pass	Kavya Raj P
		4.3 Images of food items	Noel Mathews	13/05/2024	Is there Images of food items?	Pass	Kavya Raj P
		4.4 "My Order" button in yellow font placed at the upper right corner	Noel Mathews	13/05/2024	Is there "My Order" button in black font placed at the upper right corner?	Pass	Kavya Raj P
		4.5 "Support" button in blue font placed at the upper right corner.	Noel Mathews	13/05/2024	Is there "Support" button in black font placed at the upper right corner.?	Pass	Kavya Raj P
		4.6 "Logout" button in green font placed at the upper right corner.	Noel Mathews	13/05/2024	Is there "Profile" button in black font placed at the upper right corner?	Pass	Kavya Raj P
		4.7 "My Cart" box with Checkout button	Noel Mathews	13/05/2024	Is there "My Cart" box with Checkout button?	Pass	Kavya Raj P
		4.8 Checkout button in yellow colour.	Noel Mathews	13/05/2024	Is there Checkout button in yellow colour?	Pass	Kavya Raj P
		4.9 Menu box with item name, price and "add" button in yellow colour	Noel Mathews	13/05/2024	Is there a Menu box with item name, price and "add" button in yellow colour?	Pass	Parvathy S
5	Order page	5.1 Rectangular box for each product	Noel Mathews	14/05/2024	Is there Rectangular box for each product?	Pass	Kavya Raj P
		5.1.1 Order id	Noel Mathews	14/05/2024	Is there Order number?	Pass	Kavya Raj P
		5.1.2 Food item number	Noel Mathews	14/05/2024	Is there Food item number?	Pass	Kavya Raj P
		5.1.3 Food item quantity	Noel Mathews	14/05/2024	Is there Food item quantity?	Pass	Kavya Raj P
		5.1.4 Total price	Noel Mathews	14/05/2024	Is there Total price?	Pass	Kavya Raj P
		5.1.5 Payment status	Noel Mathews	14/05/2024	Is there a payment status?	Pass	Parvathy S
		5.1.6 Schedule Later	Noel Mathews	14/05/2024	Is there a Schedule later?	Pass	Parvathy S
		5.1.7 Order status	Noel Mathews	14/05/2024	Is there a order status?	Pass	Parvathy S
6	Checkout page	6.1"Checkout" in black font written in middle	Noel Mathews	14/05/2024	Is there "Checkout" in black font written in middle?	Pass	Kavya Raj P
		6.2 Text-"Food item name with price"	Noel Mathews	14/05/2024	Is there Text-"Food item number"?	Pass	Parvathy S
		6.3 Text-"Total"	Noel Mathews	14/05/2024	Is there Text-"Total"?	Pass	Parvathy S
		6.4 Schedule later button with yellow colour	Noel Mathews	14/05/2024	Is there Schedule later button with yellow colour?	Pass	Parvathy S
		6.4.1 Text-"select a time to schedule"	Noel Mathews	14/05/2024	Is there Text-"select a time to schedule"?	Pass	Parvathy S
		6.4.2 Dropdown box for showing time	Noel Mathews	14/05/2024	Is there Dropdown box for showing time?	Pass	Parvathy S
		6.4.3 Confirm button with yellow colour	Noel Mathews	14/05/2024	Is there Confirm button with yellow colour?	Pass	Parvathy S
		6.5 Pay with Gpay button with blue colour	Noel Mathews	14/05/2024	Is therePay with Gpay button with blue colour?	Pass	Parvathy S
		6.6 Pay with PhonePe button with green colour	Noel Mathews	14/05/2024	Is there Pay with PhonePe button with green colour?	Pass	Parvathy S
		6.7 Pay with Paytm button with red colour	Noel Mathews	14/05/2024	Is there Pay with Paytm button with red colour?	Pass	Parvathy S
7	Support page	7.1"Submit feedback" in black font written at the upper left corner	Parvathy S	15/05/2024	Is there "Submit a query or feedback" in black font written at the upper left corner?	Pass	Noel Mathews
		7.1.1 Text box to enter subject	Parvathy S	15/05/2024	Is there Text box to enter subject?	Pass	Noel Mathews
		7.1.2 Text box to enter message	Parvathy S	15/05/2024	Is there Text box to enter message?	Pass	Noel Mathews
		7.1.3 "Submit feedback" button with yellow colour	Parvathy S	15/05/2024	Is there Submit button with yellow colour?	Pass	Noel Mathews
		7.2 "Query" button with yellow colour	Parvathy S	15/05/2025	Is there a "Query" button with yellow colour?	Pass	Noel Mathews
		7.2 Text box to enter query	Parvathy S	15/05/2024	is there a Text box to enter query?	Pass	Noel Mathews
		7.3 "Submit query" button with green colour	Parvathy S	15/05/2024	Is there "Previous queries and feedback" in black font written at the left corner?	Pass	Noel Mathews
8	Admin login page	8.1 Text box to enter email	Kavya Raj P	15/05/2024	Is there Text box to enter email?	Pass	Noel Mathews
		8.2 Text box to enter password	Kavya Raj P	15/05/2024	Is there Text box to enter password?	Pass	Noel Mathews
		8.3 Login button with yellow colour	Kavya Raj P	15/05/2024	Is there Login button with yellow colour?	Pass	Noel Mathews
9	Admin dashboard page	9.1 "Orders" text in black font placed at the upper left corner	Noel Mathews	15/05/2024	Is there "Orders" text in black font placed at the upper left corner?	Pass	Parvathy S
		9.2 Rectangular box for each product	Noel Mathews	15/05/2024	Is there Rectangular box for each product?	Pass	Parvathy S
		9.2.1 Order id	Noel Mathews	15/05/2024	Is there Order id?	Pass	Parvathy S
		9.2.2 Order name	Noel Mathews	15/05/2024	Is there Order name?	Pass	Parvathy S
		9.2.3 "Order Ready" button with yellow colour	Noel Mathews	15/05/2024	Is there "Order Ready" button with yellow colour?	Pass	Parvathy S
		9.2.4 "Cancel" button with blue colour	Noel Mathews	15/05/2024	Is there "Cancel" button with blue colour?	Pass	Kavya Raj P
		9.3 "Payments" button in white font at the upper right corner	Noel Mathews	15/05/2024	Is there "Payments" button in white font at the upper right corner?	Pass	Kavya Raj P
		9.4 "Queries" button in white font at the upper right corner	Noel Mathews	15/05/2024	Is there "Queries" button in white font at the upper right corner?	Pass	Kavya Raj P
		9.4 "Logout " button in white font at the upper right corner	Noel Mathews	15/05/2024	Is there "Logout " button in white font at the upper right corner?	Pass	Kavya Raj P
10	Admin-Payment page	10.1"Payments" text in black fond in the middle	Noel Mathews	15/05/2024	Is there"Payments" text in black fond in the middle?	Pass	Kavya Raj P
		10.2 Payment and Order status in black colour	Noel Mathews	15/05/2024	Is there Payment and Order status in black colour?	Pass	Kavya Raj P
11	Admin-Feedback and Query page	11.1 "Feedback" text in black colour	Noel Mathews	15/05/2024	Is there "Feedback" text in black colour?	Pass	Kavya Raj P
		11.1.1 Feedbacks with user id in black colour text	Noel Mathews	15/05/2024	Is there Feedbacks with user id in black colour text?	Pass	Parvathy S
		11.2 "Order queries" text in black colour	Noel Mathews	15/05/2024	Is there "Order queries" text in black colour?	Pass	Parvathy S
		11.2.1 Queries with order id and user id	Noel Mathews	15/05/2024	Is there Queries with order id and user id?	Pass	Parvathy S

Fig 6.1.7 Epic Diagram

6.2 Maintenance Process

6.2.1. Regular Updates

- **Feature Enhancements**: Periodically add new features or improve existing ones based on user feedback and technological advancements.
- **Security Patches**: Apply security patches promptly to address newly discovered vulnerabilities.

6.2.2. Performance Monitoring

- Continuously monitor website performance using tools and analytics to identify and address performance bottlenecks.

6.2.3. User Feedback Integration

- Regularly review user feedback and make necessary improvements to enhance the user experience.
- Implement a feedback loop to inform users about changes made based on their suggestions.

6.2.4. Data Backup

- Perform regular backups of user data, order history, and other critical information to prevent data loss.

6.2.5. Technical Support

- Provide ongoing technical support to users for any issues they encounter.
- Maintain a helpdesk or support ticket system to track and resolve user issues efficiently.

6.2.6. Documentation and Training

- Maintain comprehensive documentation for all features and functionalities of the website.
- Provide training sessions or resources for canteen staff to effectively use the system.

CHAPTER 7
CONCLUSION

The development of the cashless canteen platform represents a significant step forward in modernizing and enhancing the dining experience for college students, faculty, and staff. By leveraging a robust technological stack including HTML, CSS, JavaScript, React with Vite, Firebase, and Razorpay, this platform integrates seamlessly into the daily routines of its users, offering a comprehensive suite of features designed to streamline operations and enhance convenience. The platform's functionalities, from order scheduling and secure payment gateways to virtual queue management and interactive feedback channels, collectively address common pain points encountered in traditional dining settings.

The meticulous approach to requirement analysis, system design, development, integration, and testing ensures that the platform not only meets but exceeds user expectations. Its user-friendly interface, real-time capabilities, and secure transactions provide a reliable and efficient solution that enhances user satisfaction while optimizing operational efficiency for canteen management. The integration of Firebase for backend services and real-time data synchronization, along with Razorpay for secure and versatile payment processing, guarantees that the platform is both technologically advanced and user-centric.

Moreover, the platform's scalability and adaptability position it for future growth and continuous improvement. By implementing continuous monitoring, performance optimization, and regular updates, the platform remains responsive to evolving user needs and technological advancements. This proactive approach ensures long-term sustainability and relevance, making the cashless canteen platform a valuable asset for educational institutions aiming to modernize their dining services.

In conclusion, the cashless canteen platform is a transformative solution that aligns with the needs of modern campus life. Its comprehensive design and innovative features not only enhance the dining experience for users but also streamline canteen operations, paving the way for a more efficient, secure, and enjoyable dining environment. This project underscores the importance of integrating advanced technology in everyday campus services, ultimately contributing to a more connected and convenient campus community.

FUTURE SCOPE

1. **Advanced Personalization**
   Future enhancements could include leveraging AI and machine learning to provide personalized recommendations based on users' past orders, preferences, and dietary restrictions. Customized user profiles storing detailed preferences and health integration for nutritional insights can offer a tailored dining experience. Predictive ordering and dynamic menu adjustments will further streamline the ordering process and enhance user satisfaction.
2. **Integration with Other Campus Services**
   Integrating the platform with campus SSO systems can simplify the login process, allowing students and faculty to use existing credentials. Payments through campus ID cards can provide a seamless experience. Extending features to include library bookings, event management, and transportation services can create a unified campus services app, enhancing convenience and usability.
3. **Enhanced Order Management**
   Utilizing AI for demand forecasting can help manage inventory and staffing more efficiently, reducing wait times. Dynamic pricing models based on demand and special events can optimize revenue. Features for bulk ordering, order customization, and automated inventory management can further streamline operations, enhancing overall efficiency and customer satisfaction.
4. **Robust Analytics and Reporting**
   Developing a comprehensive analytics dashboard can provide in-depth insights into sales trends, peak times, and customer preferences, aiding decision-making. Real-time and predictive analytics can monitor trends and forecast future demands. Analyzing customer feedback and operational efficiency reports can identify areas for improvement, ensuring continuous optimization of services.
5. **Mobile Application Development**
   Creating native mobile apps for iOS and Android can provide a seamless user experience. Offline functionality will ensure users can browse menus and place orders without internet connectivity. Push notifications can keep users informed about order status and promotions. Geo-fencing can offer location-based services, and integrating mobile wallets can enhance payment convenience.

6. **Sustainability Features**

   Encouraging eco-friendly practices through digital receipts and incentives for using reusable containers can promote sustainability. Using data analytics to track food waste and optimize inventory management can reduce excess. Providing information on the carbon footprint of orders can encourage environmentally conscious choices. Partnering with environmental organizations for green initiatives can further enhance the platform's sustainability efforts.

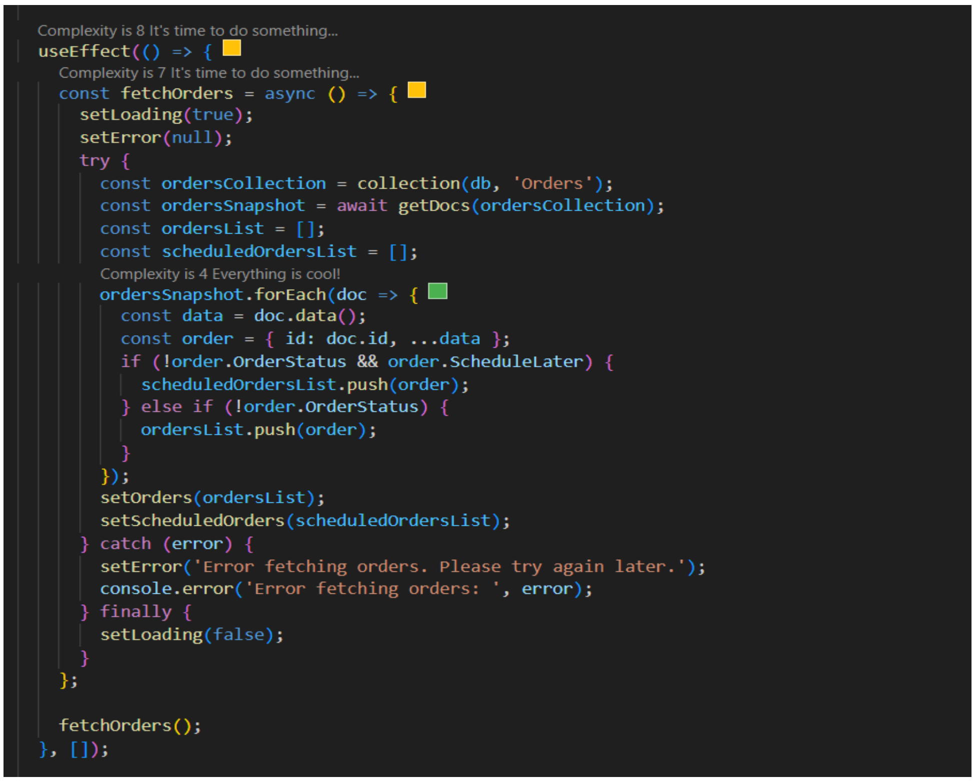
REFERENCES

1. Malla, R., & Shah, M. (2021). A Review on Campus Canteen Management system using Mobile Computing. International Research Journal of modernization in Engineering, Technology and Science, 3(3), 81-85.
2. Yang, H., & Ting, W. (2022). Canteen Food Ordering and managing system based on Android Application. Information, 51(29).
3. Joseph George, Sreenath P, Siddharth K G, Jeswill Paulose, "Food Ordering Application for Canteen" International Journal of Engineering Research & Technology (IJERT), Vol. 9 Issue 06, June-2020, ISSN: 2278-0181.
4. Monik Shah, Shalin Shah, Mohd Danish Shaikh, Kaustubh, "Canteen Automation System". International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 01 | Jan-2018, ISSN: 2395-0056.
5. A Gowthami, T Banupriya, E Vadivukkarasi, "Mobile Application For Canteen Automation System Using Android". International Journal of Advanced Research in Computer Engineering & Technology Volume 9, Issue 3, March 2020, ISSN: 2278 – 1323.
6. Kowshik reddy, Sumanth, Ashik Teja, Gopi Krishna, "Online Canteen System". 2018 JETIR October 2018, Volume 5, Issue 10, ISSN-2349-5162.
7. M. Ambika, Saravana Kumar R, Sandhya S Nair, Ranjith Kumar S,"Cashless Canteen Management System". International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-7, May 2020.
8. Shourya Pradhan, Shubham Jain, Shubhanshu Pratap Singh, Yash Gupta, "Canteen Management App". International Journal of Information Sciences and Application (IJISA). ISSN 0974-2255, Vol.11, No.1, 2019.
9. Shaina Carl, Tanmay Parulekar, Aishwarya Sedamkar, Kenneth Fonseca, "Cross Platform Application for Canteen Food Ordering System" Pramana Research Journal Volume 9, Issue 2, 2019 ISSN NO: 2249-2976.
10. Chanchal Antony, Adon Biju, Amal Kuriakose, Amal Mathew, Jismon M U (2022). Canteen Food Ordering and Managing System. International Journal of Current Science Research and Review, 5(6), 2194-2199.

APPENDIX

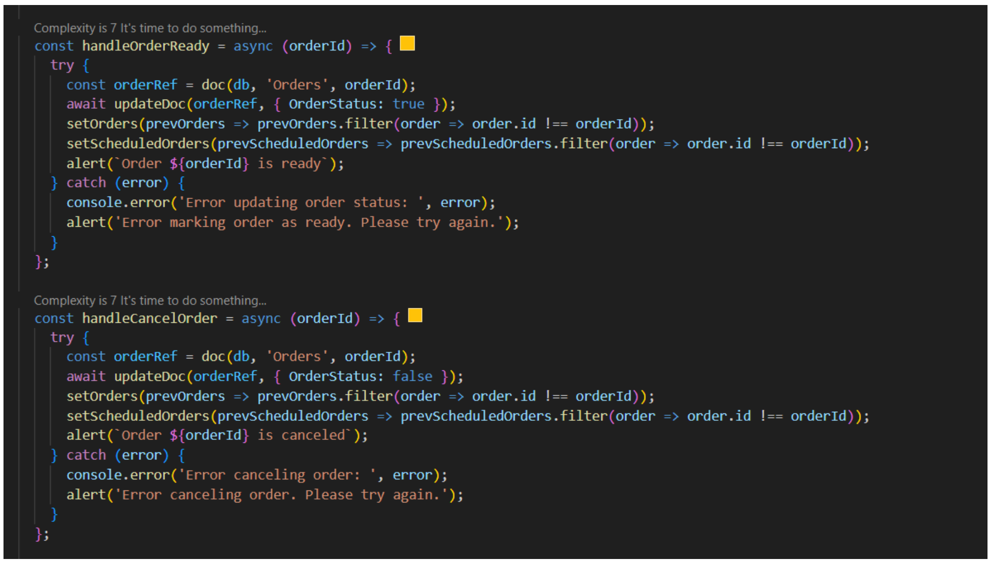
Appendix A

1. Admin Fetch



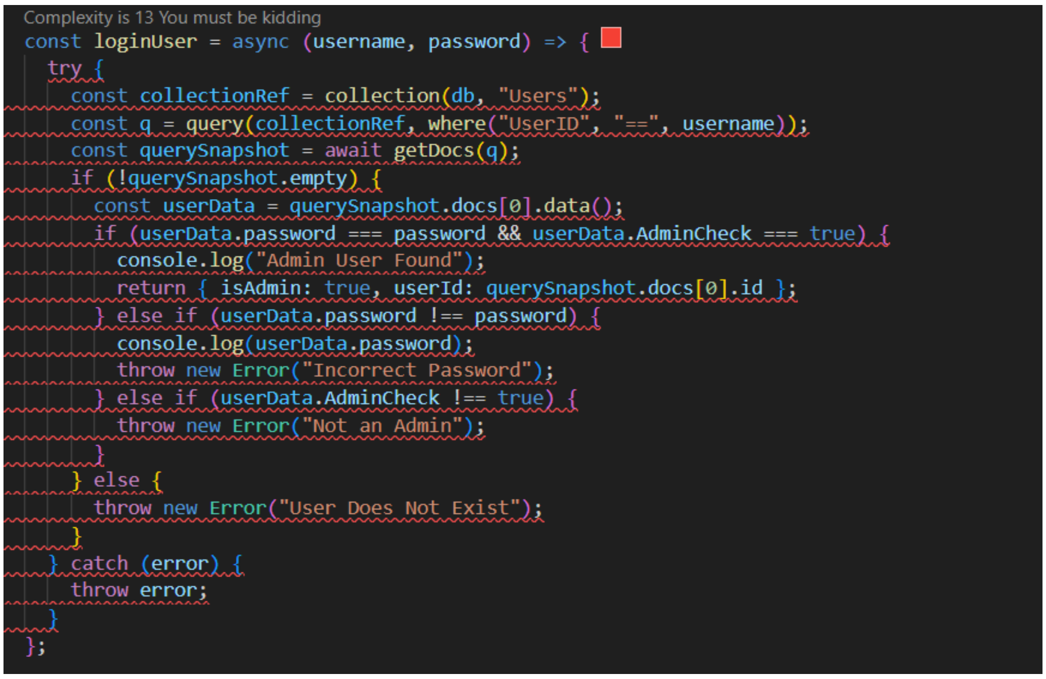
```
Complexity is 8 It's time to do something...
useEffect(() => {
  Complexity is 7 It's time to do something...
  const fetchOrders = async () => {
    setLoading(true);
    setError(null);
    try {
      const ordersCollection = collection(db, 'Orders');
      const ordersSnapshot = await getDocs(ordersCollection);
      const ordersList = [];
      const scheduledOrdersList = [];
      Complexity is 4 Everything is cool!
      ordersSnapshot.forEach(doc => {
        const data = doc.data();
        const order = { id: doc.id, ...data };
        if (!order.OrderStatus && order.ScheduleLater) {
          scheduledOrdersList.push(order);
        } else if (!order.OrderStatus) {
          ordersList.push(order);
        }
      });
      setOrders(ordersList);
      setScheduledOrders(scheduledOrdersList);
    } catch (error) {
      setError('Error fetching orders. Please try again later.');
      console.error('Error fetching orders: ', error);
    } finally {
      setLoading(false);
    }
  };

  fetchOrders();
}, []);
```

2. Admin ready cancel



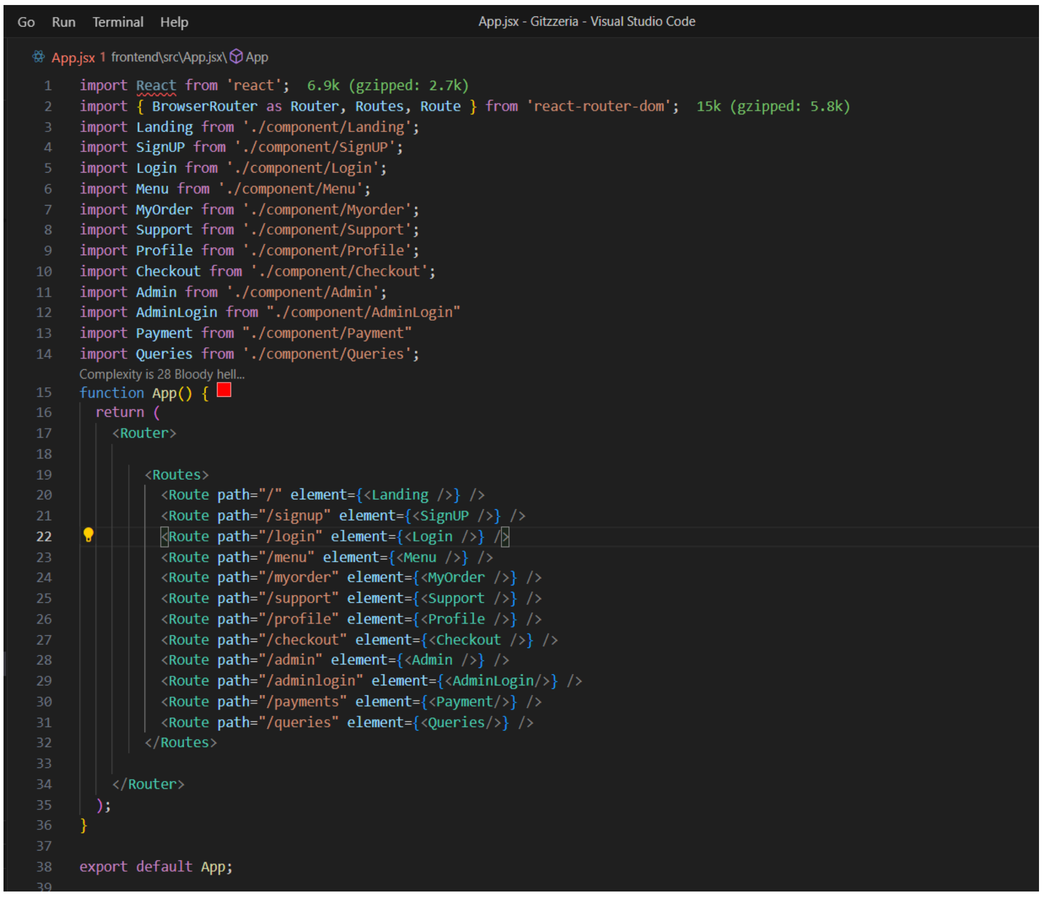
```
Complexity is 7 It's time to do something...
const handleOrderReady = async (orderId) => {
  try {
    const orderRef = doc(db, 'Orders', orderId);
    await updateDoc(orderRef, { OrderStatus: true });
    setOrders(prevOrders => prevOrders.filter(order => order.id !== orderId));
    setScheduledOrders(prevScheduledOrders => prevScheduledOrders.filter(order => order.id !== orderId));
    alert(`Order ${orderId} is ready`);
  } catch (error) {
    console.error('Error updating order status: ', error);
    alert('Error marking order as ready. Please try again.');
  }
};

Complexity is 7 It's time to do something...
const handleCancelOrder = async (orderId) => {
  try {
    const orderRef = doc(db, 'Orders', orderId);
    await updateDoc(orderRef, { OrderStatus: false });
    setOrders(prevOrders => prevOrders.filter(order => order.id !== orderId));
    setScheduledOrders(prevScheduledOrders => prevScheduledOrders.filter(order => order.id !== orderId));
    alert(`Order ${orderId} is canceled`);
  } catch (error) {
    console.error('Error canceling order: ', error);
    alert('Error canceling order. Please try again.');
  }
};
```

3. Admin login



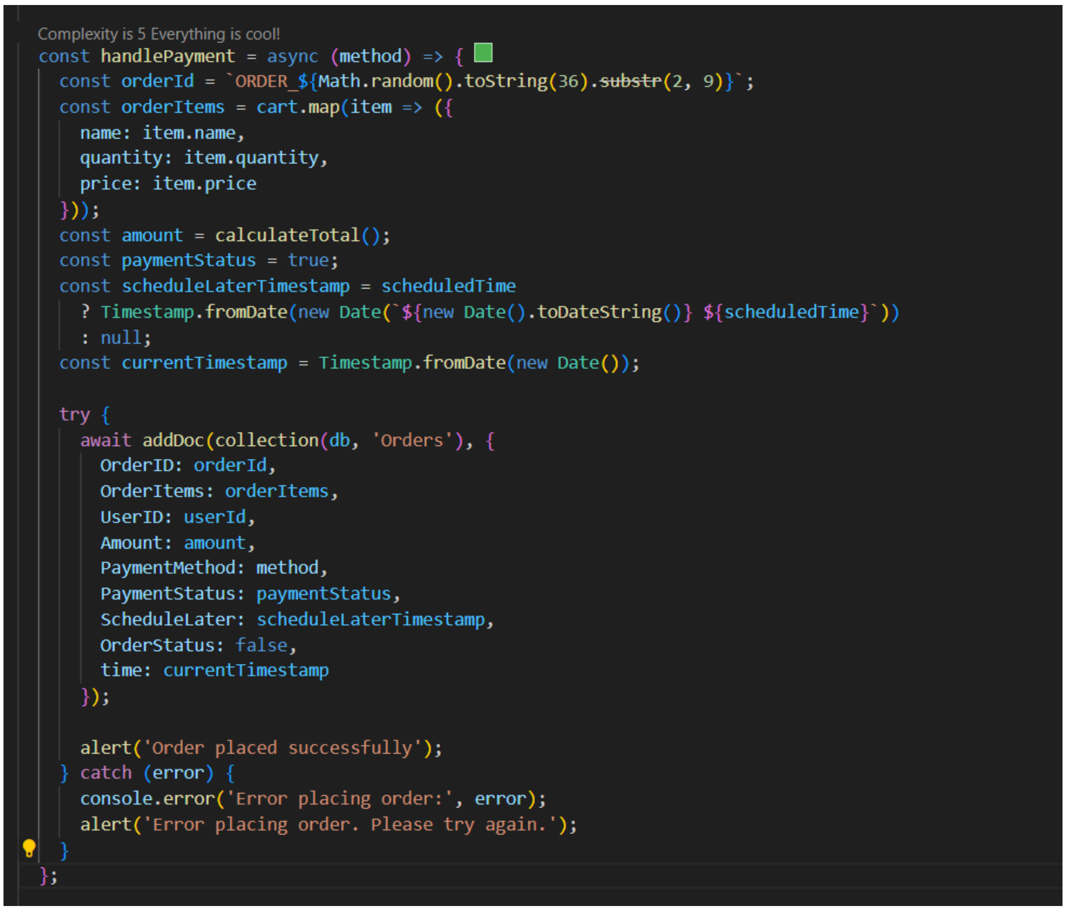
```
Complexity is 13 You must be kidding
const loginUser = async (username, password) => {
  try {
    const collectionRef = collection(db, "Users");
    const q = query(collectionRef, where("UserID", "==", username));
    const querySnapshot = await getDocs(q);
    if (!querySnapshot.empty) {
      const userData = querySnapshot.docs[0].data();
      if (userData.password === password && userData.AdminCheck === true) {
        console.log("Admin User Found");
        return { isAdmin: true, userId: querySnapshot.docs[0].id };
      } else if (userData.password !== password) {
        console.log(userData.password);
        throw new Error("Incorrect Password");
      } else if (userData.AdminCheck !== true) {
        throw new Error("Not an Admin");
      }
    } else {
      throw new Error("User Does Not Exist");
    }
  } catch (error) {
    throw error;
  }
};
```

4. App

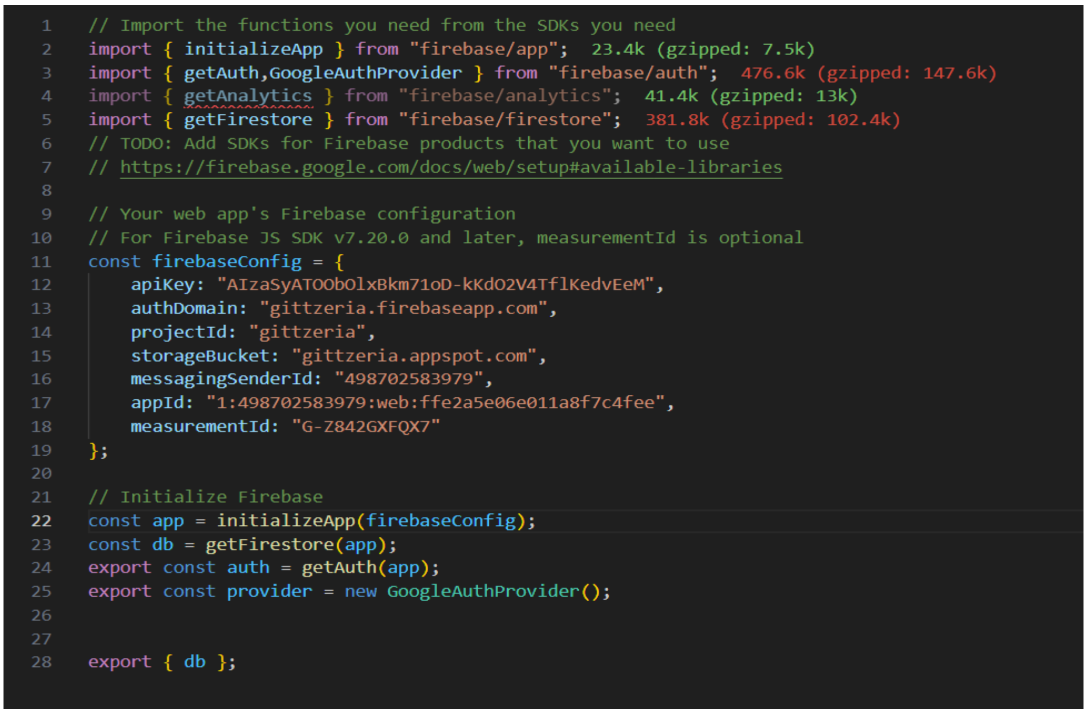

Go
Run
Terminal
Help
App.jsx - Gitzzeria - Visual Studio Code
App.jsx 1 frontend\src\App.jsx\ App

```
import React from 'react';  6.9k (gzipped: 2.7k)
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';  15k (gzipped: 5.8k)
import Landing from './component/Landing';
import SignUP from './component/SignUP';
import Login from './component/Login';
import Menu from './component/Menu';
import MyOrder from './component/Myorder';
import Support from './component/Support';
import Profile from './component/Profile';
import Checkout from './component/Checkout';
import Admin from './component/Admin';
import AdminLogin from "./component/AdminLogin"
import Payment from "./component/Payment"
import Queries from './component/Queries';
Complexity is 28 Bloody hell...
function App() {
  return (
    <Router>

      <Routes>
        <Route path="/" element={<Landing />} />
        <Route path="/signup" element={<SignUP />} />
        <Route path="/login" element={<Login />} />
        <Route path="/menu" element={<Menu />} />
        <Route path="/myorder" element={<MyOrder />} />
        <Route path="/support" element={<Support />} />
        <Route path="/profile" element={<Profile />} />
        <Route path="/checkout" element={<Checkout />} />
        <Route path="/admin" element={<Admin />} />
        <Route path="/adminlogin" element={<AdminLogin/>} />
        <Route path="/payments" element={<Payment/>} />
        <Route path="/queries" element={<Queries/>} />
      </Routes>

    </Router>
  );
}

export default App;
```

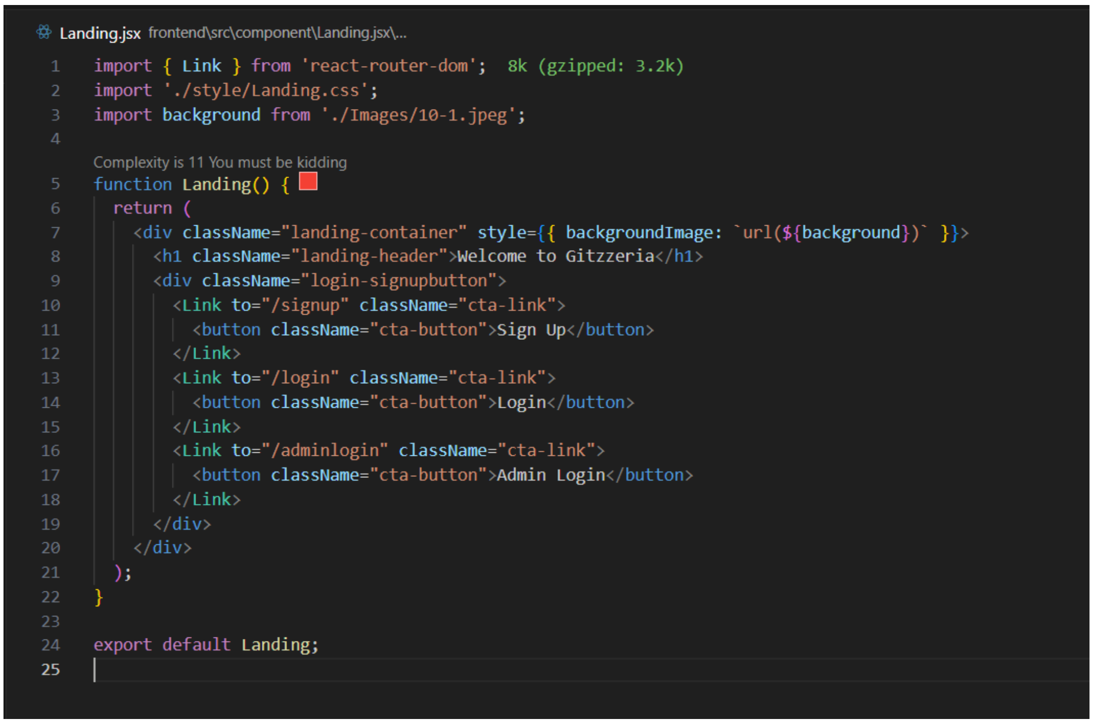
5. Checkout


```
Complexity is 5 Everything is cool!
const handlePayment = async (method) => {
  const orderId = `ORDER_${Math.random().toString(36).~~substr~~(2, 9)}`;
  const orderItems = cart.map(item => ({
    name: item.name,
    quantity: item.quantity,
    price: item.price
  }));
  const amount = calculateTotal();
  const paymentStatus = true;
  const scheduleLaterTimestamp = scheduledTime
    ? Timestamp.fromDate(new Date(`${new Date().toDateString()} ${scheduledTime}`))
    : null;
  const currentTimestamp = Timestamp.fromDate(new Date());

  try {
    await addDoc(collection(db, 'Orders'), {
      OrderID: orderId,
      OrderItems: orderItems,
      UserID: userId,
      Amount: amount,
      PaymentMethod: method,
      PaymentStatus: paymentStatus,
      ScheduleLater: scheduleLaterTimestamp,
      OrderStatus: false,
      time: currentTimestamp
    });

    alert('Order placed successfully');
  } catch (error) {
    console.error('Error placing order:', error);
    alert('Error placing order. Please try again.');
  }
};
```

6. Firebase

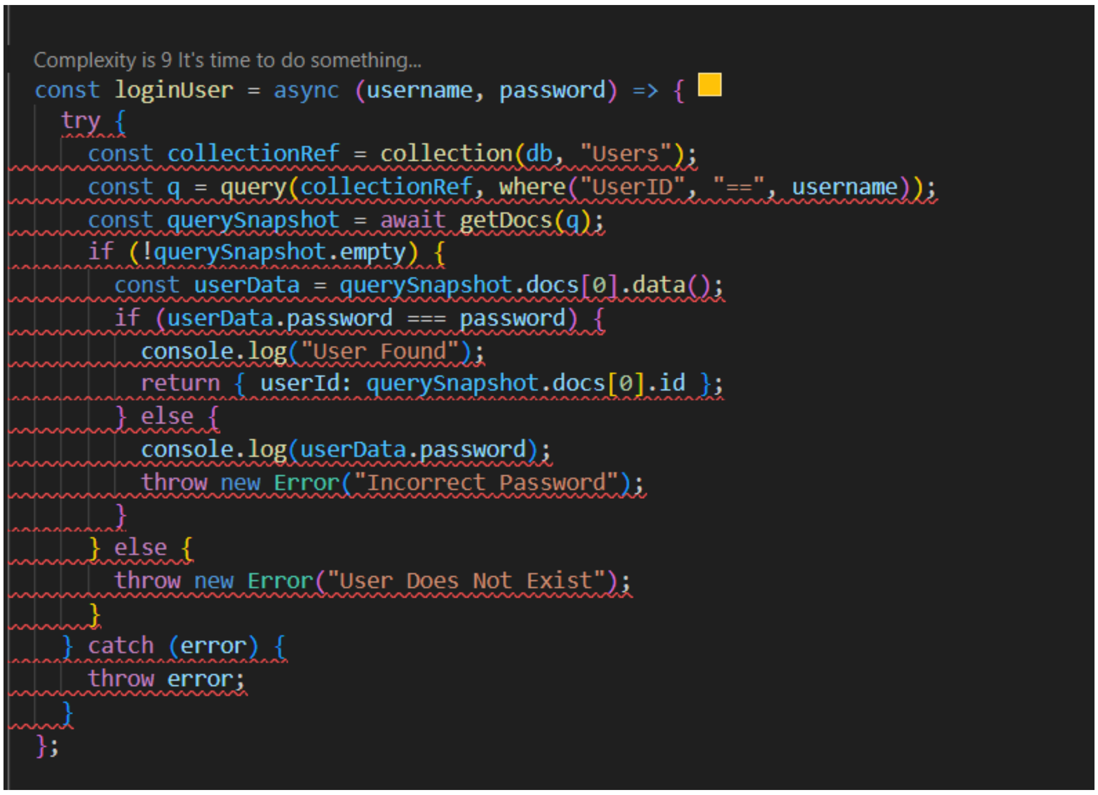

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";  23.4k (gzipped: 7.5k)
import { getAuth,GoogleAuthProvider } from "firebase/auth";  476.6k (gzipped: 147.6k)
import { getAnalytics } from "firebase/analytics";  41.4k (gzipped: 13k)
import { getFirestore } from "firebase/firestore";  381.8k (gzipped: 102.4k)
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
    apiKey: "AIzaSyATOObOlxBkm71oD-kKdO2V4TflKedvEeM",
    authDomain: "gittzeria.firebaseapp.com",
    projectId: "gittzeria",
    storageBucket: "gittzeria.appspot.com",
    messagingSenderId: "498702583979",
    appId: "1:498702583979:web:ffe2a5e06e011a8f7c4fee",
    measurementId: "G-Z842GXFQX7"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);
export const auth = getAuth(app);
export const provider = new GoogleAuthProvider();


export { db };
```

7. Landing



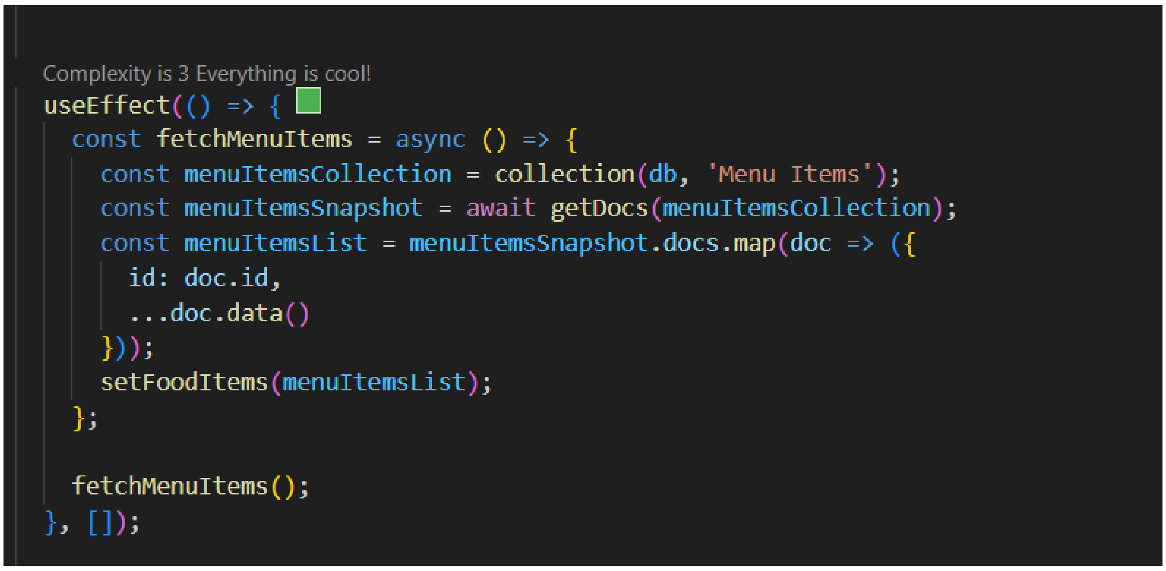
```
Landing.jsx frontend\src\component\Landing.jsx\...
import { Link } from 'react-router-dom';  8k (gzipped: 3.2k)
import './style/Landing.css';
import background from './Images/10-1.jpeg';

Complexity is 11 You must be kidding
function Landing() {
  return (
    <div className="landing-container" style={{ backgroundImage: `url(${background})` }}>
      <h1 className="landing-header">Welcome to Gitzzeria</h1>
      <div className="login-signupbutton">
        <Link to="/signup" className="cta-link">
          <button className="cta-button">Sign Up</button>
        </Link>
        <Link to="/login" className="cta-link">
          <button className="cta-button">Login</button>
        </Link>
        <Link to="/adminlogin" className="cta-link">
          <button className="cta-button">Admin Login</button>
        </Link>
      </div>
    </div>
  );
}

export default Landing;
```

8. Login



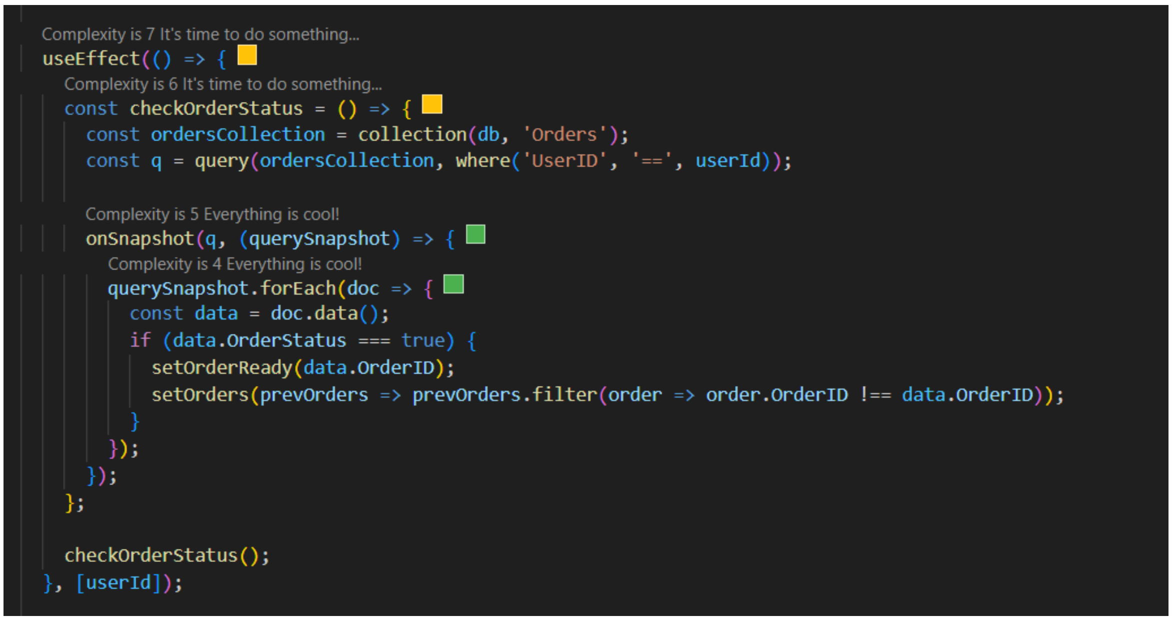
```
Complexity is 9 It's time to do something...
const loginUser = async (username, password) => {
  try {
    const collectionRef = collection(db, "Users");
    const q = query(collectionRef, where("UserID", "==", username));
    const querySnapshot = await getDocs(q);
    if (!querySnapshot.empty) {
      const userData = querySnapshot.docs[0].data();
      if (userData.password === password) {
        console.log("User Found");
        return { userId: querySnapshot.docs[0].id };
      } else {
        console.log(userData.password);
        throw new Error("Incorrect Password");
      }
    } else {
      throw new Error("User Does Not Exist");
    }
  } catch (error) {
    throw error;
  }
};
```

9.Menu



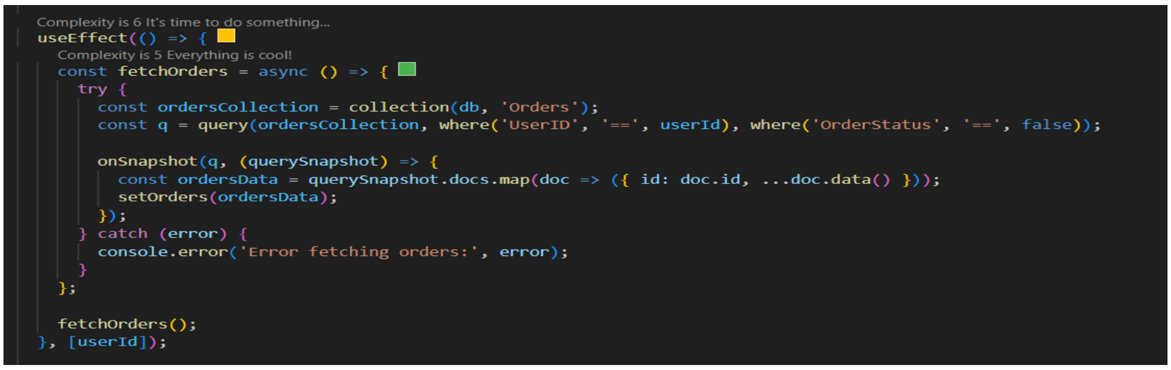
```
Complexity is 3 Everything is cool!
useEffect(() => {
  const fetchMenuItems = async () => {
    const menuItemsCollection = collection(db, 'Menu Items');
    const menuItemsSnapshot = await getDocs(menuItemsCollection);
    const menuItemsList = menuItemsSnapshot.docs.map(doc => ({
      id: doc.id,
      ...doc.data()
    }));
    setFoodItems(menuItemsList);
  };

  fetchMenuItems();
}, []);
```

10.Myorder check



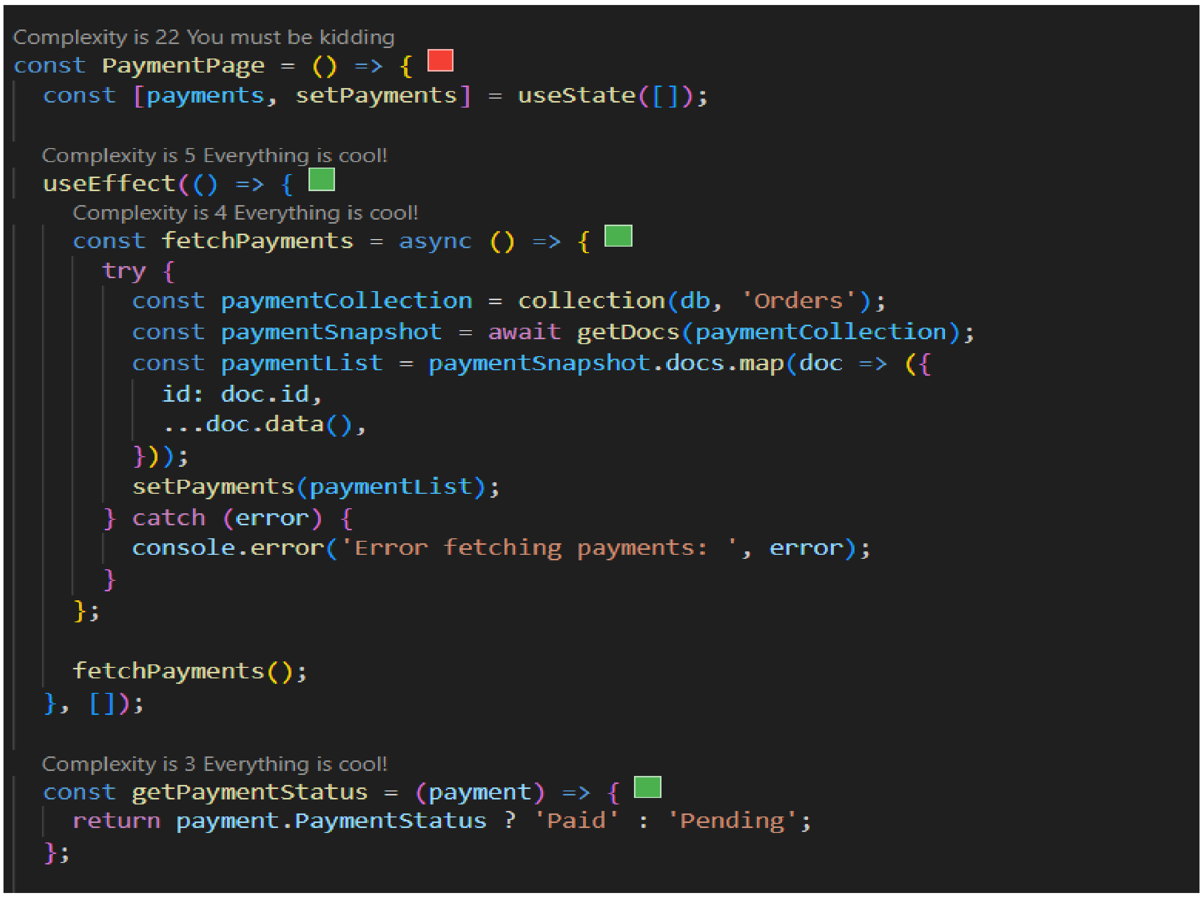
```
Complexity is 7 It's time to do something...
useEffect(() => {
  Complexity is 6 It's time to do something...
  const checkOrderStatus = () => {
    const ordersCollection = collection(db, 'Orders');
    const q = query(ordersCollection, where('UserID', '==', userId));

    Complexity is 5 Everything is cool!
    onSnapshot(q, (querySnapshot) => {
      Complexity is 4 Everything is cool!
      querySnapshot.forEach(doc => {
        const data = doc.data();
        if (data.OrderStatus === true) {
          setOrderReady(data.OrderID);
          setOrders(prevOrders => prevOrders.filter(order => order.OrderID !== data.OrderID));
        }
      });
    });
  };

  checkOrderStatus();
}, [userId]);
```

11. Myorder fetch



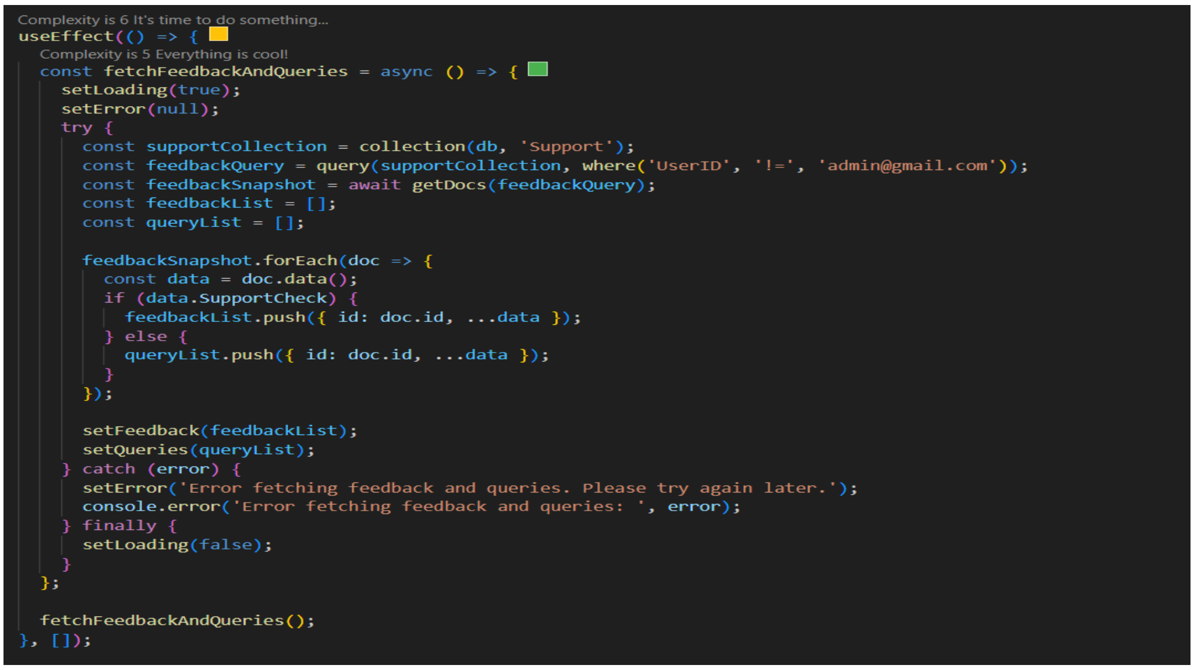
```
Complexity is 6 It's time to do something...
useEffect(() => {
  Complexity is 5 Everything is cool!
  const fetchOrders = async () => {
    try {
      const ordersCollection = collection(db, 'Orders');
      const q = query(ordersCollection, where('UserID', '==', userId), where('OrderStatus', '==', false));

      onSnapshot(q, (querySnapshot) => {
        const ordersData = querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
        setOrders(ordersData);
      });
    } catch (error) {
      console.error('Error fetching orders:', error);
    }
  };

  fetchOrders();
}, [userId]);
```

12. Payment



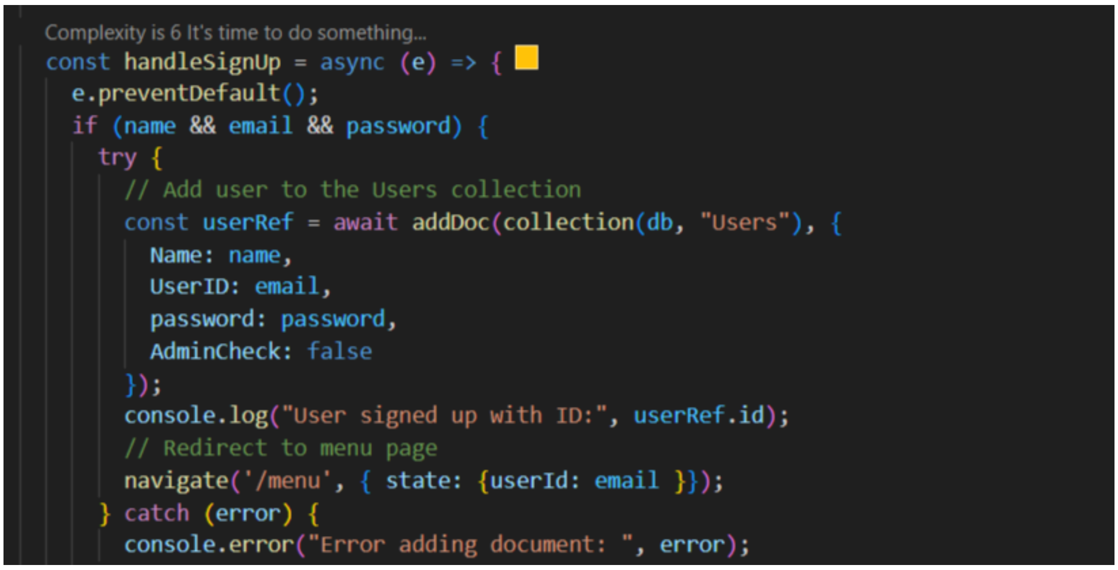
```
Complexity is 22 You must be kidding
const PaymentPage = () => { 🟥
  const [payments, setPayments] = useState([]);

  Complexity is 5 Everything is cool!
  useEffect(() => { 🟩
    Complexity is 4 Everything is cool!
    const fetchPayments = async () => { 🟩
      try {
        const paymentCollection = collection(db, 'Orders');
        const paymentSnapshot = await getDocs(paymentCollection);
        const paymentList = paymentSnapshot.docs.map(doc => ({
          id: doc.id,
          ...doc.data(),
        }));
        setPayments(paymentList);
      } catch (error) {
        console.error('Error fetching payments: ', error);
      }
    };

    fetchPayments();
  }, []);

  Complexity is 3 Everything is cool!
  const getPaymentStatus = (payment) => { 🟩
    return payment.PaymentStatus ? 'Paid' : 'Pending';
  };
```

13. Queries



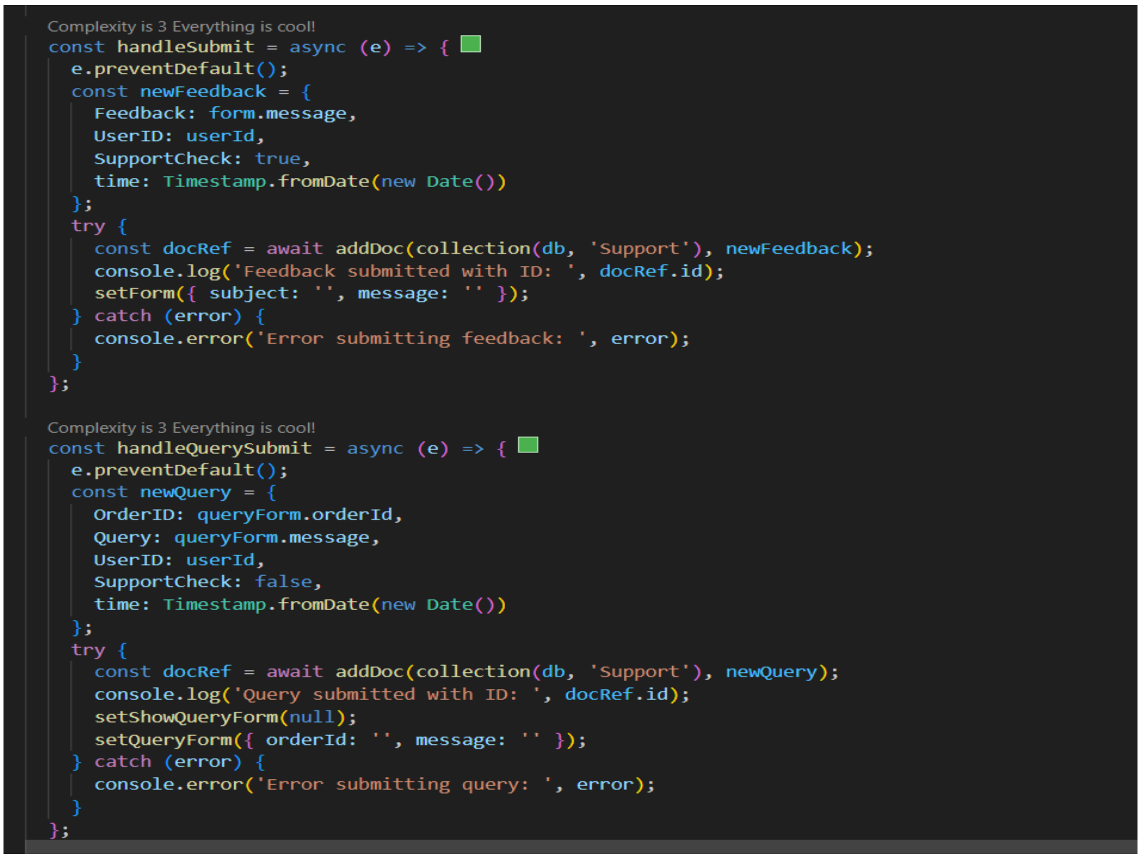
```
Complexity is 6 It's time to do something...
useEffect(() => { 🟨
  Complexity is 5 Everything is cool!
  const fetchFeedbackAndQueries = async () => { 🟩
    setLoading(true);
    setError(null);
    try {
      const supportCollection = collection(db, 'Support');
      const feedbackQuery = query(supportCollection, where('UserID', '!=', 'admin@gmail.com'));
      const feedbackSnapshot = await getDocs(feedbackQuery);
      const feedbackList = [];
      const queryList = [];

      feedbackSnapshot.forEach(doc => {
        const data = doc.data();
        if (data.SupportCheck) {
          feedbackList.push({ id: doc.id, ...data });
        } else {
          queryList.push({ id: doc.id, ...data });
        }
      });

      setFeedback(feedbackList);
      setQueries(queryList);
    } catch (error) {
      setError('Error fetching feedback and queries. Please try again later.');
      console.error('Error fetching feedback and queries: ', error);
    } finally {
      setLoading(false);
    }
  };

  fetchFeedbackAndQueries();
}, []);
```

14.Signup



```
Complexity is 6 It's time to do something...
const handleSignUp = async (e) => { 🟨
  e.preventDefault();
  if (name && email && password) {
    try {
      // Add user to the Users collection
      const userRef = await addDoc(collection(db, "Users"), {
        Name: name,
        UserID: email,
        password: password,
        AdminCheck: false
      });
      console.log("User signed up with ID:", userRef.id);
      // Redirect to menu page
      navigate('/menu', { state: {userId: email }});
    } catch (error) {
      console.error("Error adding document: ", error);
```

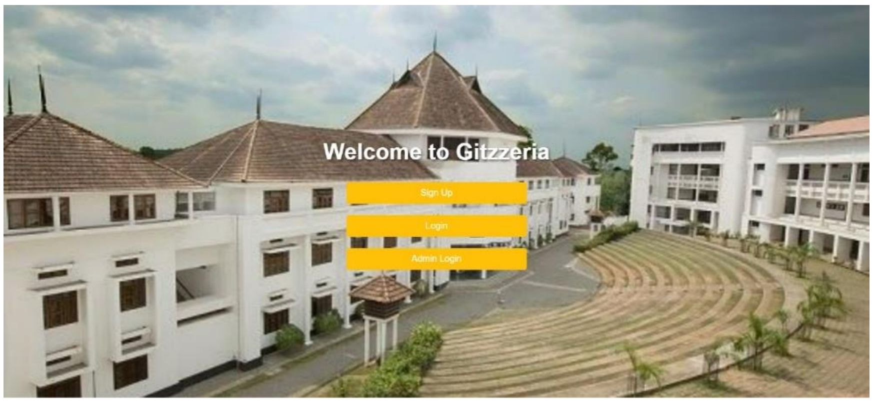
15.Support



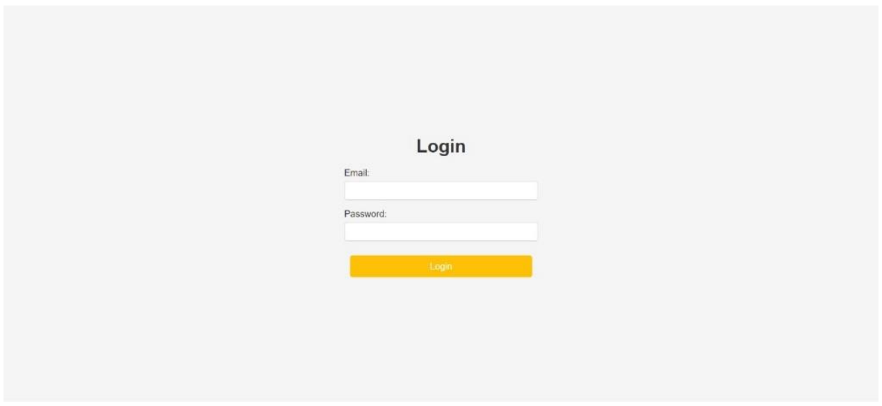
```
Complexity is 3 Everything is cool!
const handleSubmit = async (e) => { 🟩
  e.preventDefault();
  const newFeedback = {
    Feedback: form.message,
    UserID: userId,
    SupportCheck: true,
    time: Timestamp.fromDate(new Date())
  };
  try {
    const docRef = await addDoc(collection(db, 'Support'), newFeedback);
    console.log('Feedback submitted with ID: ', docRef.id);
    setForm({ subject: '', message: '' });
  } catch (error) {
    console.error('Error submitting feedback: ', error);
  }
};

Complexity is 3 Everything is cool!
const handleQuerySubmit = async (e) => { 🟩
  e.preventDefault();
  const newQuery = {
    OrderID: queryForm.orderId,
    Query: queryForm.message,
    UserID: userId,
    SupportCheck: false,
    time: Timestamp.fromDate(new Date())
  };
  try {
    const docRef = await addDoc(collection(db, 'Support'), newQuery);
    console.log('Query submitted with ID: ', docRef.id);
    setShowQueryForm(null);
    setQueryForm({ orderId: '', message: '' });
  } catch (error) {
    console.error('Error submitting query: ', error);
  }
};
```

Appendix B

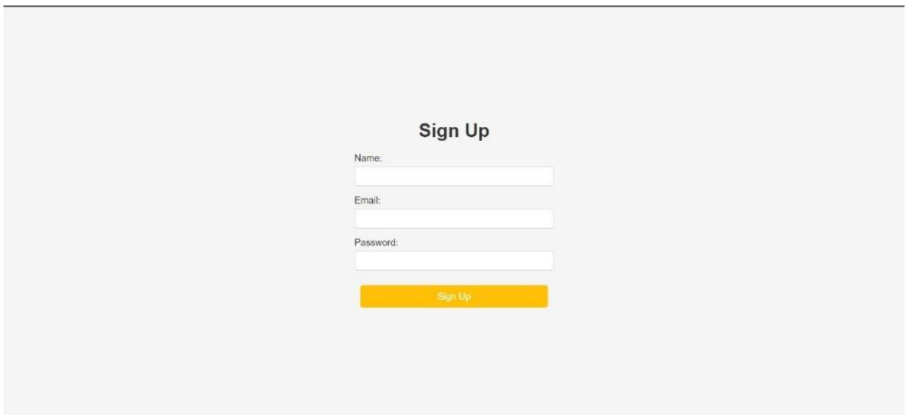
1. Landing page



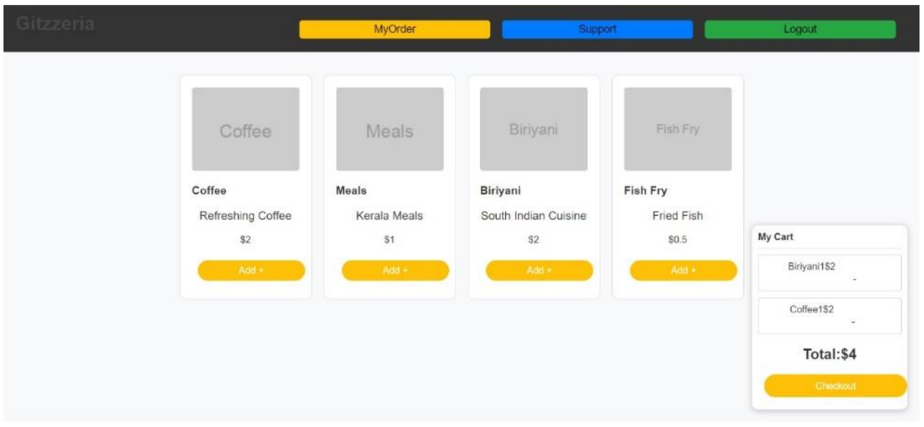
2. User login



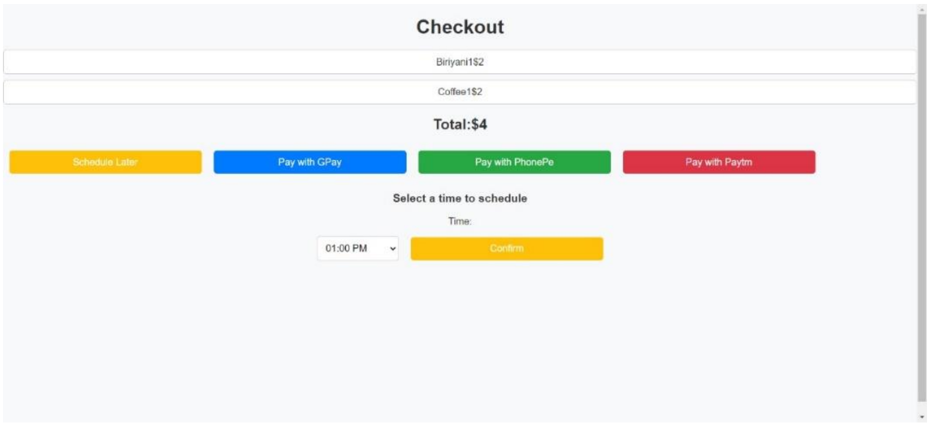
3. User sign up



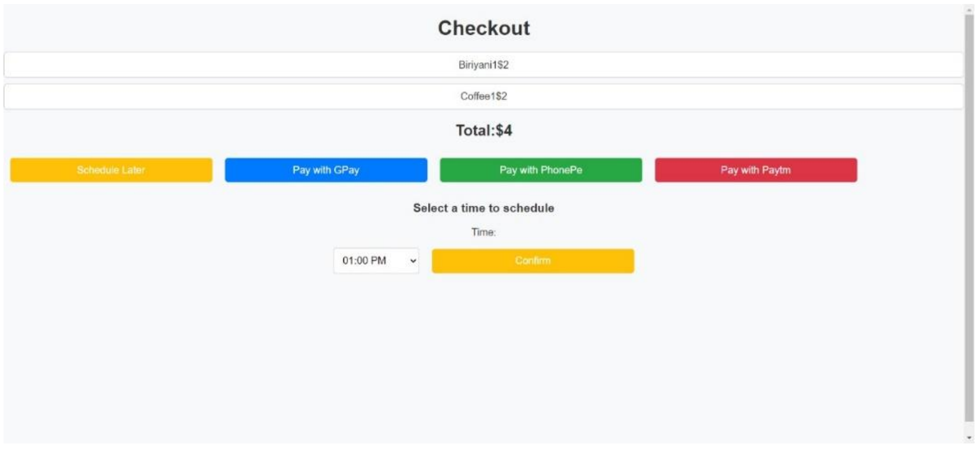
4. Menu page



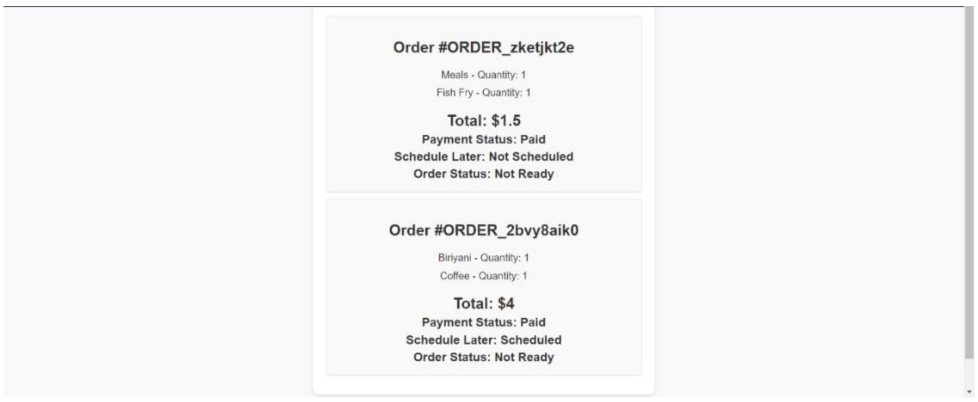
5. Checkout page



6. Schedule page



7. My order page


Order #ORDER_zketjkt2e

Meals - Quantity: 1
Fish Fry - Quantity: 1

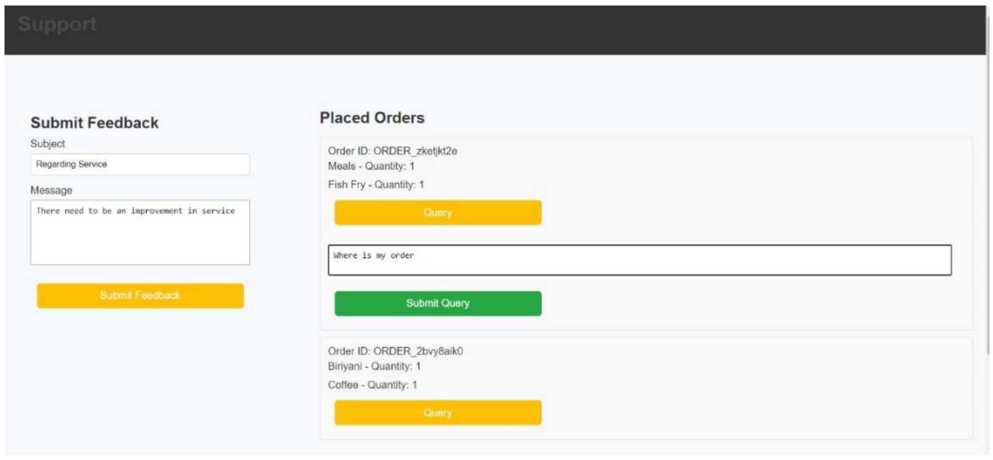
Total: $1.5
Payment Status: Paid
Schedule Later: Not Scheduled
Order Status: Not Ready

Order #ORDER_2bvy8aik0

Biriyani - Quantity: 1
Coffee - Quantity: 1

Total: $4
Payment Status: Paid
Schedule Later: Scheduled
Order Status: Not Ready

8. Feedback page



Submit Feedback

Subject
Regarding Service

Message
There need to be an improvement in service

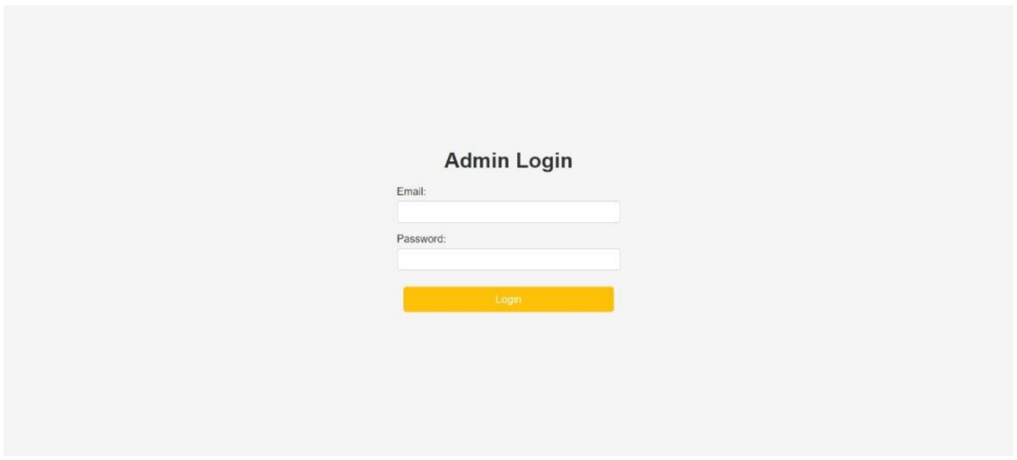
Placed Orders

Order ID: ORDER_zketjkt2e
Meals - Quantity: 1
Fish Fry - Quantity: 1

Where is my order

Order ID: ORDER_2bvy8aik0
Biriyani - Quantity: 1
Coffee - Quantity: 1

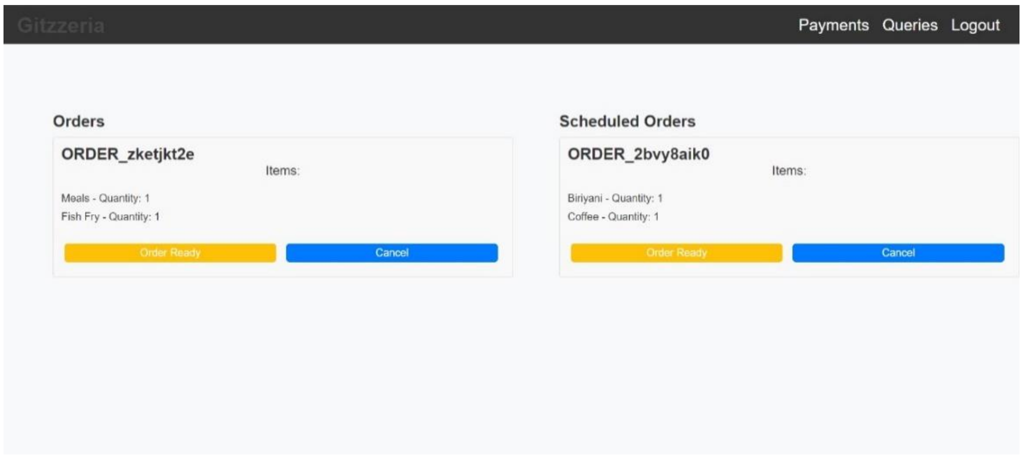
9. Admin login


Admin Login

Email:

Password:

10. Admin dashboard



Orders

ORDER_zketjkt2e

Items:

Meals - Quantity: 1
Fish Fry - Quantity: 1

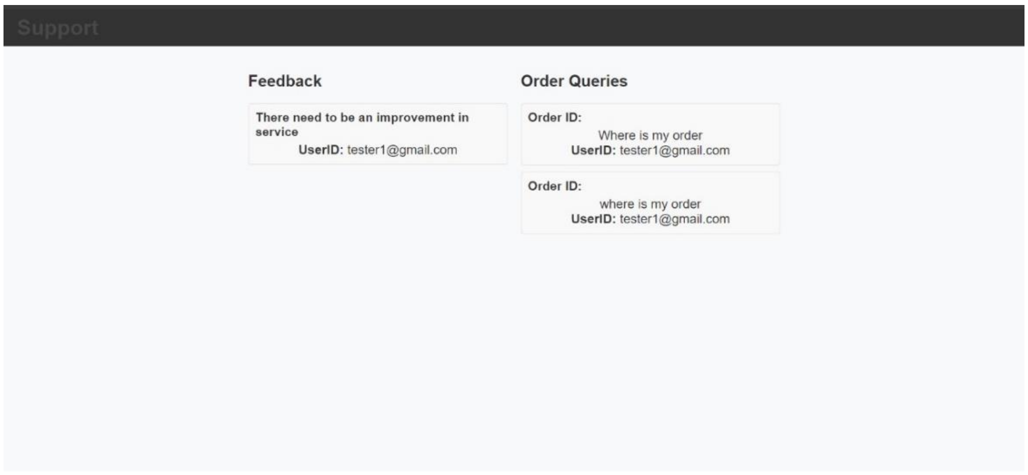
Scheduled Orders

ORDER_2bvy8aik0

Items:

Biriyani - Quantity: 1
Coffee - Quantity: 1

11. Admin query



Feedback

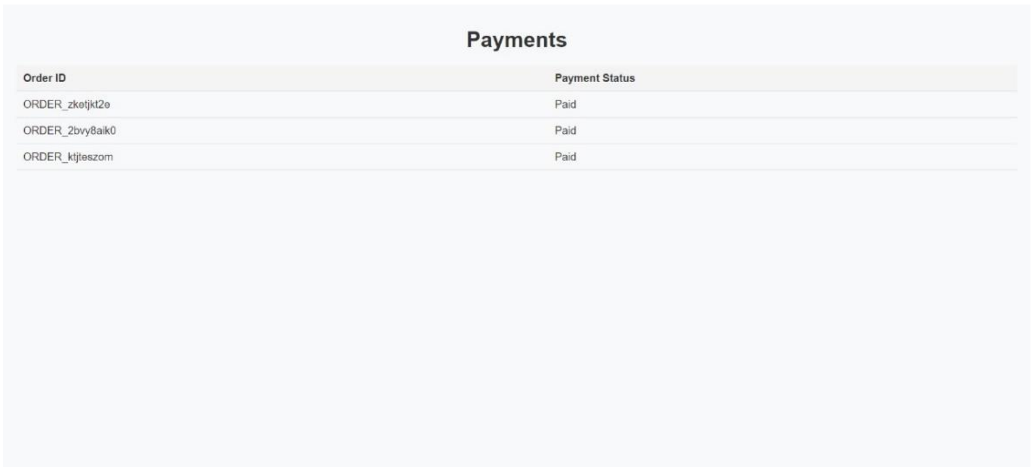
There need to be an improvement in service
UserID: tester1@gmail.com

Order Queries

Order ID:
Where is my order
UserID: tester1@gmail.com

Order ID:
where is my order
UserID: tester1@gmail.com

12. Admin payment



Payments

Order ID	Payment Status
ORDER_zketjkt2e	Paid
ORDER_2bvy8aik0	Paid
ORDER_ktjteszom	Paid