Tarun K S, 2025, 13:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Smart Canteen: Efficient Food Ordering for College Campuses

Tarun K S, Assistant Professor R.S.Nagasundaramd
VISTAS

Abstract- College canteen, also known as a college cafeteria or college food service, is a facility within a college or university campus that provides food and beverages to students, faculty, staff, and visitors. It serves as a convenient on-campus dining option, offering a variety of meals, snacks, and beverages to meet the nutritional needs of the college community. Current systems may rely on manual order processing, leading to delays and inaccuracies in fulfilling student and staff orders. The College Canteen Food Ordering System is a web-based platform designed to streamline and enhance the food ordering process for students and staff within a college campus. This system involves four main actors: Students, Staff, College Admin, and Canteen Admin, each with distinct roles and functionalities. Students can access the system to browse the canteen menu, add items to their cart, and securely place orders. Real-time notifications keep them informed about order status, and they have the option to provide feedback on food quality and service. The system also enables students to manage their profiles and view order history. College Admins are responsible for user management, ensuring the smooth functioning of student and staff accounts. They monitor overall system performance, access order data, and generate reports for strategic decision-making. Canteen Admins play a pivotal role in updating the menu, receiving orders, processing payments, managing inventory, and coordinating with kitchen staff. They handle payment confirmations, generate invoices, and respond to inquiries from students and staff. Feedback from users is actively addressed to enhance service quality. The system integrates secure authentication measures, a user-friendly interface, and real-time notifications to enhance the user experience. Regular reporting and feedback mechanisms contribute to continuous system improvement. The College Canteen Food Ordering System aims to create an efficient, transparent, and enjoyable food ordering experience within the college community.

Keywords- College Canteen, Food Ordering System, Online Food Ordering, Campus Dining System, Canteen Management System, Web-based Ordering, Student Food Services

I. INTRODUCTION

Canteen is the heart of every college or institution. It caters to the basic needs of everyone. The canteen is primarily responsible for serving nutritious and hygienic food to the students and staff. It also serves meals to the students residing in the campus. As many students come from distant places, it is essential for the students to have nutritious food and refreshments at affordable prices so as to participate in the daily academic activities actively. The college canteen plays an important role in this regard by catering the daily nutritional requirements of students and staff members.

The objective of the Canteen and meal service is to protect by reducing the risk of foodborne illness, with proper sanitary conditions, and preventing adulterated food. From the very beginning of the College, the College Canteen has been functioning efficiently. The canteen is located inside the College. The hostellers and day scholars are provided meals by assuring food safety and quality. The students who come from faraway places, start their journey to the College early in the morning. The mid-day meal is provided. Even many day-scholars use canteen facilities for their breakfast and evening tea with snacks. The canteen provides healthy, tasty eatables, fresh fruit juices, hot and cold drinks to the students at subsidized rates. The location of our college canteen is such that it is easily approachable from all the departments. Kitchen staff takes care to provide the students and staff a nutritious and hygienic food at our campus canteens. A variety of hygienic food and snack items includes South and North Indian Meals, Variety Rice's Chinese Foods Fresh Juices, etc.

1.2. Problem Statement

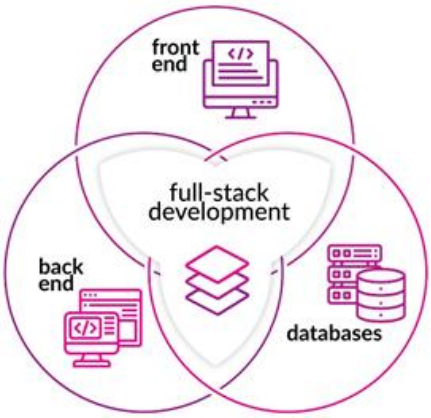
The traditional manual processes in college canteens often lead to a range of operational inefficiencies. With order processing handled through physical forms or word-of-mouth communication, there is a higher likelihood of delays, miscommunication, and errors, especially during peak hours when demand is at its highest. This not only frustrates students and staff but also burdens canteen staff with excessive administrative tasks, reducing the overall effectiveness of the operation. Moreover, without a centralized system, it becomes difficult to track inventory accurately, manage menu changes, or ensure timely payments, often resulting in a lack of transparency for users. Students and staff may also face inconvenience due to the limited availability of information on menus, order statuses, or special promotions, which affects their overall dining experience. To address these challenges, the Smart Canteen project proposes a modern, web-based solution that automates and streamlines canteen operations. By providing real-time menu browsing, order management, payment processing, and inventory tracking, the platform ensures faster, more accurate service. It also introduces transparent communication between users and staff, enabling seamless order placement and updates. The feedback system further enhances user satisfaction, giving students and staff a voice in improving food quality and service. Ultimately, the Smart Canteen project aims to improve operational efficiency, reduce administrative workload, and elevate the overall dining experience within the college campus.

1.3. Full Stack Development

Full stack development is the process of designing, creating, testing, and deploying a complete web application from start to finish. Full stack developers possess the skills to work on all layers of an application, from the user interface to the server and database. This comprehensive approach allows them to create end-to-end solutions by integrating different technologies and frameworks

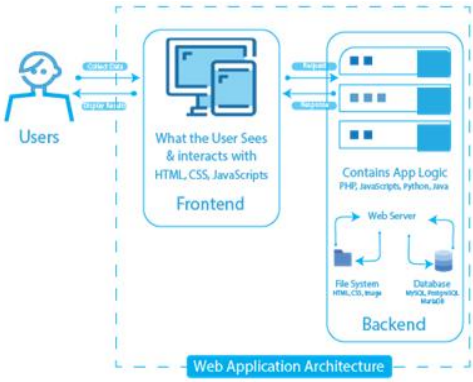
.

It involves working with various technologies and tools, including front-end web development, back-end web development, and database development. And full stack development is a term used to describe a software engineer or developer who works with both the front and back end of a website or application. A full-stack developer is comfortable working with front-end and back-end technologies that power a website or application.



Frontend: A website's or web app's frontend is the user interface visitors use to take action. Frontend development deals with everything that users see or do on a website or web app, from design to the customer journey, like how users navigate the website. This also involves improving the website's usability, user-friendliness, and aesthetic appeal while making it easier for the users to navigate

.



Frontend development employs programming languages and frameworks such as CSS, HTML, React, Angular, TezJS, JavaScript, TypeScript, NextJS, and Vue. Here's the deal, you can hire expert full-stack software engineers and web development team from Radixweb. Our professional developers are well-versed in every full-stack programming language and framework, so you can be stress-free during development and deployment.

Backend: Backend development creates software that enables interaction between the interface and the database. The backend of a website must be kept up-to-date, and backend developers must manage servers, software, and databases. They should know all the popular backend frameworks. Making it straightforward to access and update involves working on the server, website structure, and database. The ultimate goals of backend development are to control what clients can't see and ensure they have a positive experience on your website or web app.

Database Management: Full-stack developers use database technologies such as MySQL, PostgreSQL, and MongoDB to store and manage data within the web application. They must also be familiar with APIs (Application Programming Interfaces) which enable software applications to communicate with each other.

1.4. AIM AND OBJECTIVE

Aim

The aim of the project is to develop and implement a project that enhances the efficiency, transparency, and user experience of food ordering processes within the college campus.

Objective

• To streamline food ordering processes for students and staff.

• To improve menu visibility and accessibility in real-time.

• To facilitate efficient communication among canteen staff.

• To optimize inventory management and stock control.

• To ensure robust security measures and compliance standards.

• To empower users with control over their profiles and orders.

• To enhance the overall user experience of campus dining.

1.5. SCOPE OF THE PROJECT

The scope of the project focusing on the following key aspects:

• System Development: Designing and building a web-based platform that facilitates food ordering processes for students and staff within the college campus.

• User Interfaces: Creating intuitive and user-friendly interfaces for students, staff, college admins, and canteen admins to interact with the system effectively.

• Menu Management: Implementing features for updating, managing, and displaying the canteen menu, including item descriptions, prices, and availability.

- Order Processing: Developing functionalities for placing, processing, and tracking food orders in real-time, ensuring timely and accurate order fulfillment.
- Communication Channels: Integrating communication tools to facilitate seamless interaction among canteen staff, including kitchen and delivery personnel, for efficient order management.
- Inventory Control: Establishing a centralized inventory management system to monitor stock levels, track usage, and facilitate timely restocking of food items.
- Security Measures: Implementing robust security measures, such as encryption and user authentication, to safeguard user data and ensure transaction security.
- Feedback Mechanisms: Incorporating features for users to provide feedback on food quality, service, and overall experience, enabling continuous improvement.
- User Management: Providing functionalities for users to create and manage their profiles, view order history, and access personalized settings.
- Reporting and Analytics: Developing tools for generating reports and analyzing data related to order trends, sales performance, and inventory management for informed decision-making.
- Integration and Deployment: Integrating various components of the system and deploying it on reliable hosting platforms to ensure accessibility and scalability.
- Training and Support: Providing training materials and support resources to users and administrators to facilitate smooth adoption and operation of the system.

The scope also includes ongoing maintenance and updates to ensure the system's continued functionality, security, and relevance to the needs of the college community.

II. SYSTEM ANLYSIS

2.1. EXISTING SYSTEM

The existing system for food ordering in college canteens typically involves manual processes and limited technological support. Here's an overview of the typical characteristics of the existing system:

Manual Order Placement: Students and staff physically visit the canteen counter to place their food orders. This process often involves queuing, which can result in long waiting times during peak hours.

•Physical Visit to Canteen Counter

In the existing system, students and staff who wish to order food from the college canteen must physically visit the canteen counter. They approach the counter and join the queue of other customers who are also waiting to place their orders.

•Order Taking by Canteen Staff

When a student or staff member reaches the front of the queue, they interact with the canteen staff who are stationed behind the counter. The canteen staff member takes the customer's order by verbally asking for their food preferences and choices. The customer communicates their order, specifying the items they want to purchase, along with any customizations or special requests.

•Manual Writing of Orders

As the customer provides their order, the canteen staff manually write down the items and any special instructions on a paper order form or notepad. The written order may include details such as the name of the item, quantity, and specific preferences (e.g., extra sauce, no onions).

•Communication to Kitchen Staff

After the order is written down, the canteen staff member communicates the order to the kitchen staff who are responsible for food preparation. This communication may occur verbally or through physical delivery of the written order slip to the kitchen area.

•Waiting Time for Order Preparation

Once the order is communicated to the kitchen staff, the customer must wait for their food to be prepared. The waiting time can vary depending on factors such as the complexity of the order, the number of orders being processed, and the efficiency of the kitchen staff.

•Order Pickup and Payment

When the food is ready, the customer is notified by the canteen staff, either verbally or through a call-out of their name. The customer then proceeds to the pickup counter to collect their order. Upon receiving the food, the customer makes the

payment, either in cash or through other accepted payment methods.

•Completion of Order Transaction

After the customer has collected their order and made the payment, the transaction is considered complete. The customer can then take their food and proceed to enjoy their meal.

Overall, the manual order placement process in college canteens involves several steps, including physical visits to the counter, verbal communication of orders, manual writing of orders, and waiting times for order preparation. While this traditional approach has been widely used, it can be inefficient and time-consuming, especially during peak hours when queues are long and waiting times are high.

2.1.1. DISADVANTAGES

•Manual order placement leads to long waiting times and queues at the canteen counter.

•Limited visibility into menu options and availability contributes to uninformed choices.

•Verbal communication of orders increases the risk of errors and misunderstandings.

•Lack of centralized order tracking makes it difficult to monitor order status and processing times.

•Manual inventory management results in inconsistencies and stock shortages.

•Limited feedback channels hinder opportunities for improving food quality and service.

•Cash-based transactions pose security risks and inconvenience for users.

•Lack of user control over orders and profiles limits customization and personalization options.

2.2. PROPOSED SYSTEM

The proposed system for the project aims to revolutionize the food ordering experience within the college campus by leveraging modern technology and automation. Here's an overview of the key features and functionalities of the proposed system:

•Online Ordering Platform

Introduce a user-friendly web-based platform accessible via desktop and mobile devices, allowing students and staff to place orders remotely.

•Dynamic Menu Presentation

Implement a dynamic menu interface that displays real-time updates on available items, prices, and descriptions, enabling informed decision-making.

•Order Management System

Develop a centralized system for managing orders from placement to fulfillment, providing real-time status updates and notifications to users.

•Automated Communication Channels

Establish automated communication channels between canteen staff and kitchen personnel to streamline order processing and minimize delays.

•Inventory Tracking Mechanism

Integrate an inventory management system to monitor stock levels, track usage, and automate restocking processes for efficient supply chain management.

•Secure Online Transactions

Implement secure online payment gateways to facilitate cashless transactions, ensuring data privacy and financial security for users.

•Feedback Mechanism

Incorporate a feedback mechanism allowing users to provide ratings and reviews, enabling continuous improvement of food quality and service.

•User Profiles and Preferences

Enable users to create personalized profiles, save order preferences, and access order history for a seamless and customized ordering experience.

•Administrative Dashboard

Provide administrative dashboards for canteen admins and college admins to manage menus, track orders, and generate reports for data-driven decision-making.

2.2.1. Advantages

•Enhanced convenience through remote ordering from any location within the campus.

•Real-time menu updates for informed decision-making and improved user experience.

•Streamlined order processing, reducing waiting times and increasing efficiency.

•Automated communication channels for seamless coordination among canteen staff.

•Improved inventory management, minimizing stock shortages and wastage.

•Secure online transactions for cashless payments, ensuring financial security for users.

•Continuous feedback mechanism for quality improvement and customer satisfaction.

•Personalized user profiles and order history for a customized dining experience.

•Comprehensive administrative dashboards for data-driven decision-making and operations management.
•Scalability and flexibility to adapt to future growth and changes in canteen operations.
•Ongoing training and support for users and administrators to maximize system utilization and effectiveness.
Existing studies emphasize the prospect of mobile-based library systems. [1] Which shows how Android applications minimize manual labour through automated book transactions. [2], [3], [4], [5], [6], [7] up to [12] all the research emphasizes Android Studio's advantages, including compatibility with Material Design guidelines and easy integration with cloud databases such as Firebase. Features that are common in previous studies are book searching, user login, and fundamental transaction tracking, commonly done with Java and SQLite. Gaps exist, such as poor consideration of offline operation, sophisticated user experience (UX) design, and support for cutting-edge technologies such as IoT. This study improves on the foundations by the inclusion of Kotlin, cloud synchronisation, and richer UX, fixing flaws established in previous studies.

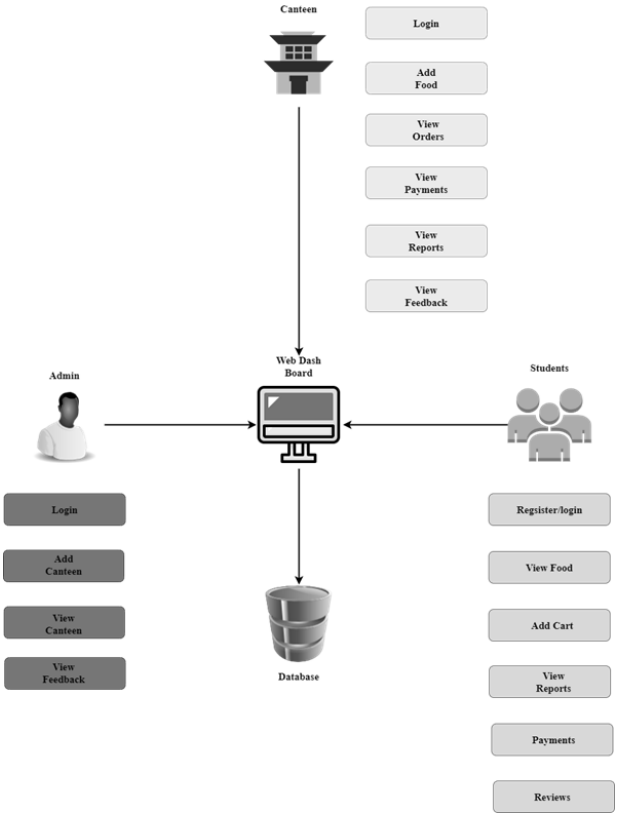
III. SYSTEM SPECIFICATION

3.1. HARDWARE REQUIREMENTS
•Processor : Intel Core i5 or higher
•RAM : 8GB or more
•Storage : Minimum 256GB SSD
•Display : 14" Full HD (1920x1080) resolution or higher
•Network Interface: Ethernet or Wi-Fi connectivity
3.2. SOFTWARE REQUIREMENTS
•Operating System: Windows 10
•Programming : Python
•Framework : Flask framework for building the web application
•Web Design : HTML, CSS, JavaScript, and Bootstrap
•Database : MySQL

IV. SYSTEM DESIGN

4.1. INTRODUCTION

The system design of the project utilizes a client-server architecture to streamline food ordering and management processes. The design ensures a user-friendly and efficient experience for students, staff, and administrators. The front-end leverages HTML, CSS, JavaScript, and Bootstrap to create a responsive interface, while the back-end uses Python and Flask for seamless server-side processing. MySQL is employed for reliable data storage, handling user profiles, orders, and menu items. The app integrates secure payment gateways to support both online and cash-on-delivery options.
4.2. ARCHITECTURE DESIGN



4.3. INPUT DESIGN

The input design of the project is focused on creating an intuitive and user-friendly interface to facilitate easy data entry and ensure smooth interactions for all users. The system design includes various input forms and validation mechanisms for different functionalities:
1. User Login/Authentication

The login interface allows users to input their username and password to access their accounts. Validation checks ensure that the username exists in the system and the password matches the stored hash. In case of incorrect login attempts, clear error messages are displayed, guiding users to correct their inputs.

2. Menu Browsing

The menu browsing section offers a search bar for users to find food items quickly. Users can also filter items by categories like vegetarian, non-vegetarian, drinks, or based on availability. Input fields for the name, description, and price of menu items are available for administrators to manage the offerings.

3. Order Placement

In the order placement section, users can select menu items, specify the quantity, and provide optional special instructions. The system ensures that the item is in stock, and quantities are valid, while the cart dynamically updates to show added items, their quantities, and the total price.

4. Payment

For payment, users can choose their payment method, enter card details (card number, expiration date, CVV) for online payments, or opt for cash on delivery. Secure input fields and real-time validation ensure that sensitive data is entered correctly. A final confirmation screen displays the total amount, including applicable taxes and discounts, before completing the transaction.

5. Feedback and Ratings

The feedback section allows users to rate food items with a scale from 1 to 5 stars and provide textual comments. Validation ensures ratings fall within the accepted range and that the comment box isn't empty. Users are prompted to complete all fields before submitting their feedback.

6. Admin Panel

For administrators, the menu management input fields allow them to add, edit, or remove items. Administrators can enter details such as item name, price, availability, and description. Additional options for setting time-based promotions or availability periods are also available.

7. Order Tracking

The order tracking interface requires the order ID for administrators to manage and update the status of orders (e.g., in progress, completed). The status can be changed as needed to reflect the real-time progress of an order.

8. Notification Preferences

The notification preferences section enables users to select how they would like to be notified about updates, including email, SMS, or in-app alerts. Users can save these preferences for future updates and notifications.

This input design ensures clarity, reduces the chances of error, and provides users with a seamless experience while interacting with the Campus Dining Web App.

4.4. OUTPUT DESIGN

The output design of the project is focused on providing users and administrators with clear, concise, and actionable information, ensuring a seamless experience and effective management of operations.

1. User Dashboard

The user dashboard displays a personalized view of the dining experience. It includes information such as menu options, active orders, order status, and any notifications related to promotions or new items. Users can quickly view their order history, track the status of their current order, and access payment receipts. The dashboard also shows recommended items based on their previous orders and ratings.

2. Menu Display

The menu page presents the available food items, organized into categories (e.g., starters, mains, drinks, etc.). For each item, the user can see the name, price, description, availability status, and any special promotions or discounts. Images of the items are included for better visualization. The system dynamically updates to reflect real-time availability of items and promotions.

3. Order Confirmation Screen

After users finalize their order, an order confirmation screen appears. It summarizes the items selected, including their quantity and total price, with an option to edit the order. The screen also displays an estimated delivery time and offers a final confirmation button to submit the order.

4. Payment Confirmation

Once payment is processed, a payment confirmation page displays the transaction details, including the amount paid, payment method, and a transaction ID for the user's reference. If the payment is successful,

users receive a receipt with an option to download or email it for record-keeping.

5. Order Tracking

The order tracking page allows users to monitor the progress of their order in real-time. It shows the current status (e.g., processing, out for delivery, delivered) with a timestamp and a progress bar. Users can also view the estimated delivery time and get notifications if the status changes.

6. Feedback and Ratings Confirmation

Once a user submits feedback or a rating for an item, a confirmation message appears, thanking them for their input. Users can also view their previous ratings and comments on items they have rated in the past.

7. Admin Panel

For administrators, the admin panel offers an overview of all active orders, inventory levels, and menu items. Administrators can view real-time order statuses, update menu items, and access detailed reports such as sales data, user feedback, and system performance. They can also manage user accounts, including viewing user activity, updating order statuses, and sending notifications to users.

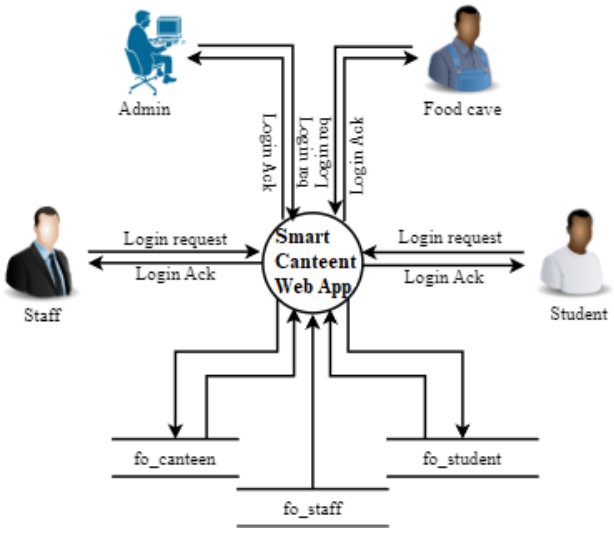
8. Notifications

Notifications are delivered in real-time through in-app alerts, SMS, or email, depending on the user's preference. These notifications include order updates, promotions, payment confirmations, and other relevant messages such as special offers or menu changes.

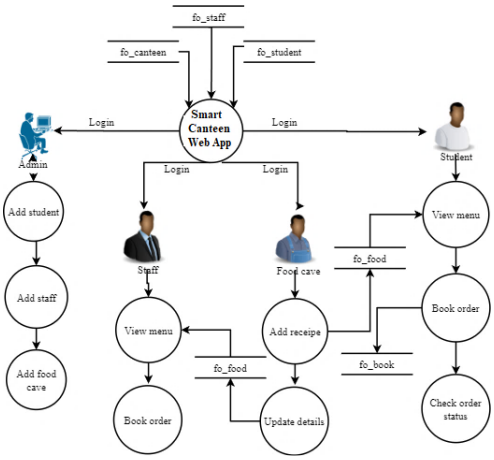
The output design is focused on providing an efficient, user-friendly experience for both users and administrators. By presenting information clearly and in an easily accessible manner, it ensures smooth interactions and helps maintain transparency throughout the entire process.

4.5. DATA FLOW DIAGRAM

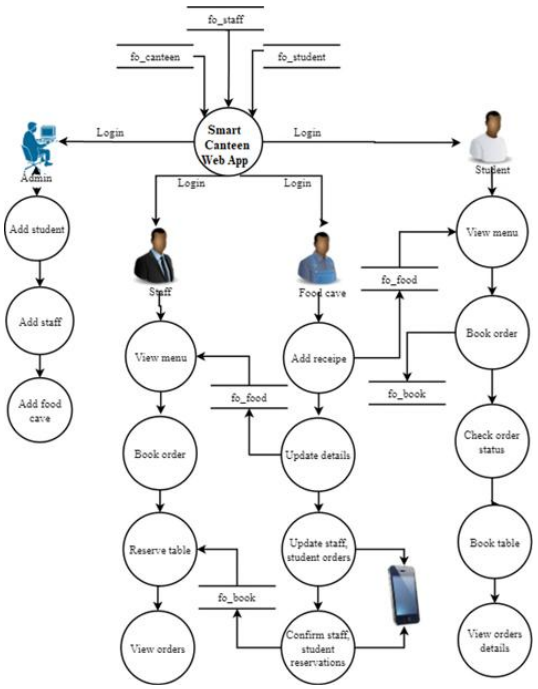
LEVEL 0



LEVEL 1



LEVEL 2



4.6. DATABASE DESIGN

1. ADMIN TABLE

Description: Stores administrator login credentials

S.No	Field	Data Type	Field Size	Constraint	Description
1	Username	Varchar	20	**Primary Key**	Admin username
2	Password	Varchar	20	Not Null	Admin password

2. STAFF TABLE

Description: Stores staff details, including login credentials

S.No	Field	Data Type	Field Size	Constraint	Descri
1	Staff_Id	Int	11	**Primary Key**	Unique
2	Name	Varchar	50	Not Null	Staff Na
3	Mobile	Varchar	15	Not Null	Staff No
4	Email	Varchar	50	Unique	Staff Em
5	Department	Varchar	50	**Foreign Key** (References Department Table)	Staff Departn
6	Username	Varchar	20	Unique	Login Usernan
7	Password	Varchar	20	Not Null	Login Passwor
8	Join_Date	DATE	-	Not Null	Date of

STUDENT TABLE

Description**:** Stores student information and login credentials

S.No	Field	Data Type	Field Size	Constraint	Description
1	Student_Id	Int	11	**Primary Key**	Unique Student ID
2	Name	Varchar	50	Not Null	Student Name
3	Mobile	Varchar	15	Not Null	Student Mobile No
4	Email	Varchar	50	Unique	Student Email
5	Department	Varchar	50	**Foreign Key** (References Department Table)	Student Department
6	Username	Varchar	20	Unique	Login Username
7	Password	Varchar	20	Not Null	Login Password
8	Join_Date	DATE	-	Not Null	Date of Joining

CANTEEN TABLE

Description: Stores canteen vendor details.

S.No	Field	Data Type	Field Size	Constraint	Description
1	Canteen_Id	Int	11	**Primary Key**	Unique Canteen Vendor ID
2	Name	Varchar	50	Not Null	Vendor Name
3	Mobile	Varchar	15	Not Null	Vendor Mobile No
4	Email	Varchar	50	Unique	Vendor Email
5	Profile	Varchar	50	-	Vendor Profile Details
6	Address	Varchar	100	Not Null	Vendor Address
7	Username	Varchar	20	Unique	Login Username
8	Password	Varchar	20	Not Null	Login Password
9	Join_Date	DATE	-	Not Null	Date of Joining

5. FOOD TABLE

Description: Stores details of food items available in the canteen.

S.No	Field	Data Type	Field Size	Constraint	Description
1	Recipe_Id	Int	11	**Primary Key**	Unique Recipe ID
2	Recipe_Name	Varchar	50	Unique	Name of the Recipe
3	Price	Int	50	Not Null	Price of the Recipe
4	Category	Varchar	50	-	Recipe Category
5	Type	Varchar	50	-	Recipe Type (Veg/Non-Veg)
6	Image	Varchar	100	-	Recipe Image URL
7	Post_Date	DATE	-	Not Null	Date Recipe was Added

6. FOOD ORDER TABLE

Description: Stores customer food orders.

S.No	Field	Data Type	Field Size	Constraint	Description
1	Order_Id	Int	11	**Primary Key**	Unique Order ID
2	Name	Varchar	50	Not Null	Customer Name
3	Mobile	Varchar	15	Not Null	Customer Mobile No
4	Department	Varchar	50	-	Customer Department
5	User_Type	Varchar	20	-	Type of Orderer
6	Recipe_Id	Int	11	**Foreign Key** (References Food Table)	Ordered Recipe
7	Quantity	Int	11	Not Null	Quantity Ordered
8	Price	Int	20	Not Null	Price of Recipe
9	Order_Date	DATE	-	Not Null	Date of Order
10	Username	Varchar	20	**Foreign Key** (References Student Table)	Customer Username
11	Order_Status	Varchar	20	-	Status of Order

7. TABLE BOOKING

Description: Stores restaurant table booking details.

S.No	Field	Data Type	Field Size	Constraint	Description
1	Book_Id	Int	11	**Primary Key**	Unique Booking ID
2	Name	Varchar	50	Not Null	Name of the Booker
3	Mobile	Varchar	15	Not Null	Contact No
4	Department	Varchar	50	-	Department
5	User_Type	Varchar	20	-	Type of User
6	Recipe_Id	Int	11	**Foreign Key** (References Food Table)	Ordered Recipe
7	Price	Int	11	Not Null	Price of Recipe
8	Book_Date	DATE	-	Not Null	Booking Date
9	Book_Time	TIME	-	Not Null	Booking Time
10	Guest_Count	Int	5	Not Null	Number of Guests
11	Username	Varchar	20	**Foreign Key** (References Student Table)	Booking Username
12	Status	Varchar	20	-	Booking Status

V. SOFTWARE DESCRIPTION

5.1. PYTHON 3.8

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard library which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing

Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the

fundamental high-level building block for doing practical, real world data analysis in Python.

Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda's library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

5.2. MYSQL 5

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

5.3. WAMPSERVER

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription.

5.4. BOOTSTRAP

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Oper)

5.5. FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python. Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

VI. SYSTEM IMPLEMENTATION

6.1. MODULES

1. Campus Dining Web App
2. End User
3. Authentication Module
4. Menu Management

5. Order Management
6. Payment
7. Notification
8. Feedbacks and Ratings

6.2. MODULES DESCRIPTION

1. Campus Dining Web App

The design and development of the Campus Dining Web App involve creating both frontend and backend components, along with setting up a MySQL database.

Frontend Development

•Use HTML, CSS, and JavaScript to create the user interface for the web app.

•Utilize Bootstrap framework for responsive design and layout.

•Implement dynamic elements and interactivity using JavaScript.

•Integrate Bootstrap components for consistent styling and user experience.

Backend Development with Flask

•Install Flask, a lightweight Python web framework, using pip.

•Define routes and views to handle HTTP requests and serve web pages.

•Implement authentication and authorization mechanisms for user login and access control.

•Develop controllers and business logic to process orders, manage menus, and handle other app functionalities.

•Use Flask extensions such as Flask-WTF for form handling and Flask-MySQLdb for MySQL database integration.

Database Design and Development with MySQL

•Install and configure MySQL database server using Wampserver.

•Design the database schema to store user data, menu items, orders, and other relevant information.

•Create tables, define relationships, and establish constraints to ensure data integrity.

•Use SQL queries to insert, update, retrieve, and delete data from the database.

•Connect Flask backend to MySQL database using SQLAlchemy or Flask-MySQLdb for database operations.

Deployment

•Deploy the Campus Dining Web App to a Wampserver environment for hosting.

•Configure server settings and permissions to ensure proper functioning of the app.

•Monitor server performance and troubleshoot any deployment-related issues.

•Continuously update and maintain the app to address user feedback and improve functionality.

2. End User

2.1. Student

Authentication: Students can log in using their student credentials. Additionally, a password recovery/reset functionality is available to assist with any login issues.

Menu Interaction: Students can browse the canteen menu, viewing item names, descriptions, and prices. They have the ability to add items to their cart, view and modify the cart contents, and specify any special instructions or preferences for their orders.

Order Placement: Students can securely confirm and place their orders, choosing between online payment or cash-on-delivery options for added convenience.

Order Status: Students receive real-time notifications confirming their order placement and can check the status of current orders, as well as view the history of past orders for reference.

Feedback System: Students can provide feedback on food quality and service, rating and commenting on specific items to help improve the overall dining experience.

User Profile: Students have the option to edit their personal information (optional) and view their order history to track their dining habits and preferences over time.

Notifications: Students receive alerts for promotions, discounts, or new menu items to stay informed about any updates or special offers.

2.2. Staff

Authentication: Staff members log in using their credentials, with password recovery/reset functionality available if needed.

Order Processing: Staff receive real-time notifications for new orders and can view and manage incoming orders, updating order status as needed and marking orders as processed.

Menu Management: Staff have the ability to add, edit, or remove items from the menu, update item availability, adjust prices and descriptions, ensuring the menu is up-to-date and accurate.

Inventory Management: Staff monitor and manage food inventory, receiving alerts for low stock and updating stock levels after processing orders to ensure items are available for order.

Communication: Staff communicate with kitchen staff for order preparation and can send notifications to students regarding order status.

Reporting: Staff generate reports on daily sales, popular items, and inventory levels, analyzing trends and optimizing stock based on demand.

Feedback Handling: Staff view and respond to feedback from students, addressing any issues related to food quality or service to enhance the overall dining experience.

2.3. College Admin

Authentication: College admins log in using their admin credentials, with password recovery/reset functionality available as a safeguard.

User Management: College admins manage student and staff accounts, reset passwords, and handle any account-related issues that may arise.

Order Monitoring: College admins access and monitor overall order data, view order history, and generate reports on sales and popular items for analysis.

System Management: College admins oversee general system settings, ensuring security and compliance with relevant regulations.

2.4. Canteen Admin

Authentication: Canteen admins log in using their credentials, with password recovery/reset functionality in place for security purposes.

Menu Management: Canteen admins update the menu by adding, editing, or removing items, indicating items as temporarily out of stock as needed.

Order Handling: Canteen admins receive real-time notifications for new orders, access a centralized dashboard displaying incoming orders, and process orders, updating order status and communicating with kitchen staff for order preparation.

Supply Orders: Canteen admins ensure that kitchen staff are aware of order details for food preparation, monitor and manage inventory to fulfill orders, and coordinate with delivery staff if applicable.

Payment Handling: Canteen admins receive payment, confirm payment status for each order, handle cash payments upon order delivery, and generate invoices for each order.

Inventory Management: Canteen admins monitor inventory levels, receive alerts for low stock, update stock levels after processing orders, and ensure accurate stock levels for each menu item.

Communication: Canteen admins coordinate with kitchen staff regarding order details and respond to inquiries from students or staff regarding menu items, orders, or payment.

Reporting: Canteen admins generate reports on daily sales, popular items, and inventory levels, analyzing trends and making data-driven decisions to optimize operations.

Feedback Handling: Canteen admins access and respond to feedback provided by students and staff, using feedback to improve the quality of service and menu offerings.

3. Authentication Module

The Authentication Module serves as the gateway for user access to the Campus Dining Web App, ensuring secure authentication and authorization processes. It allows users to log in using their respective credentials, such as usernames and passwords, verifying their identity before granting access to the system. Additionally, the module incorporates functionalities for password recovery and reset, providing users with a seamless experience in case they forget their login credentials. Through robust encryption and authentication mechanisms, the Authentication Module safeguards sensitive user information and ensures the integrity of user accounts, thereby maintaining a secure environment for users to interact with the web application.

4. Menu Management

The Menu Management Module is pivotal in maintaining an up-to-date and dynamic canteen menu within the project. This module empowers administrators to curate and manage the menu offerings efficiently. Administrators can seamlessly add, edit, or remove items from the menu, ensuring that it reflects the latest culinary offerings available to users. Additionally, administrators can update item availability status, providing real-time information to users about which items are currently in stock. The module allows for adjustments to item descriptions, prices, and special promotions,

ensuring accurate and enticing presentation to users. Through intuitive interfaces and streamlined processes, the Menu Management Module facilitates the smooth operation of the canteen menu, enhancing user experience and satisfaction within the Campus.

5. Order Management

The Order Management Module is the backbone of the project, orchestrating the entire lifecycle of food orders placed by users. This module seamlessly facilitates the journey of orders from placement to fulfillment, ensuring a smooth and efficient process for both users and staff. Users can browse the menu, add items to their carts, and securely place orders, while staff members receive real-time notifications for new orders. The module tracks order status, allowing users to stay informed about the progress of their orders, from processing to delivery. Furthermore, it maintains a comprehensive order history, enabling users to review past orders for reference or reordering. Through robust features and seamless integration with other modules, the Order Management Module optimizes the ordering process, enhancing user satisfaction and operational efficiency within the Campus Dining Web App.

6. Payment

The Payment Module facilitating secure and convenient transactions for users. This module enables users to choose between various payment methods, including online payment and cash-on-delivery, providing flexibility and convenience. For online payments, the module integrates with secure payment gateways to ensure the confidentiality and integrity of financial transactions. Users can securely enter their payment information and complete transactions with peace of mind. Additionally, the Payment Module includes features for handling cash payments upon order delivery, ensuring seamless payment processing for users who prefer this option. Through robust encryption and adherence to industry-standard security protocols, the Payment Module ensures the security of sensitive financial data and enhances the overall user experience within the Campus Dining Web App.

7. Notification

The Notification Module keeping users and staff informed about important updates and events in real-time. This module enables the system to send notifications to users and staff members via various channels, such as email, SMS, or in-app alerts. Users receive notifications regarding order confirmations, status updates, promotions, discounts, or new menu items, ensuring they stay informed and engaged throughout their dining experience. Similarly, staff members receive notifications for new orders, order status changes, critical system updates, or issues requiring attention. By delivering timely and relevant notifications, the Notification Module enhances user satisfaction, improves operational efficiency, and fosters better communication between users and staff within the Campus Dining Web App.

8. Feedbacks and Ratings

The Feedback and Rating Module gathering user insights and enhancing the overall dining experience within the Campus Dining Web App. This module empowers users to provide feedback on food quality, service, and overall experience through ratings and comments. Users can rate specific items on the menu and provide detailed comments, allowing administrators to understand their preferences and areas for improvement. Additionally, the module enables users to share their suggestions and recommendations, fostering a collaborative environment for continuous enhancement. Administrators can access and analyze the feedback provided by users, leveraging it to make informed decisions and implement necessary improvements. By facilitating open communication and soliciting user feedback, the Feedback and Rating Module contributes to the refinement of menu offerings, service quality, and overall user satisfaction within the Campus Dining Web App.

VII. SYSTEM TESTING

7.1. SOFTWARE TESTING

System testing in the software development life cycle aimed at verifying that the implemented system meets the specified requirements and functions correctly. Here's an overview of the different types of testing that can be conducted for the Campus Dining Web App:

1.Unit Testing

•Test individual components or modules of the system in isolation.

•Verify the functionality of functions, methods, or classes.
•Use tools like unittest in Python to automate unit tests.
2.Integration Testing
•Test the interaction between different modules or components of the system.
•Verify that components work together as expected.
•Ensure proper communication and data exchange between frontend and backend components.
•Conduct tests for API endpoints and database interactions.
3.Functional Testing
•Test the system's functionality against the specified requirements.
•Verify that users can perform actions such as logging in, browsing the menu, placing orders, etc., as intended.
•Conduct tests for edge cases, error handling, and boundary conditions.
4.User Interface (UI) Testing
•Test the user interface for usability, responsiveness, and consistency across different devices and browsers.
•Verify that UI elements are properly aligned, styled, and functional.
•Conduct tests for user interactions such as clicking buttons, entering text, etc.
5.Performance Testing
•Test the system's performance under various load conditions.
•Measure response times, throughput, and resource utilization to identify bottlenecks and optimize performance.
•Conduct stress testing to determine the system's capacity and scalability.
6.Security Testing
•Test the system for vulnerabilities and weaknesses that could be exploited by attackers.
•Conduct tests for authentication, authorization, data encryption, and protection against common security threats such as SQL injection, cross-site scripting (XSS), etc.
7.Compatibility Testing
•Test the system's compatibility with different operating systems, browsers, and devices.
•Verify that the web app functions correctly and displays properly across a variety of platforms.
8.User Acceptance Testing (UAT)
•Involve end users (students, staff, admins) to validate the system against their requirements and expectations.
•Gather feedback and address any usability issues or concerns raised by users.
9.Regression Testing
•Repeatedly test the system after each change or update to ensure that existing functionality has not been affected.
•Automate regression tests to streamline the testing process and detect regressions early.
By conducting thorough testing across these different types, the Campus Dining Web App can be validated for functionality, performance, security, and user satisfaction, ensuring a high-quality and reliable system for users.

7.2. TEST CASES

1. Test Case ID: TC001
Input: Valid student login credentials
Expected Result: User is authenticated and directed to student dashboard
Actual Result: User successfully logged in and directed to dashboard
Status: Passed

2. Test Case ID: TC002
Input: Invalid student login credentials
Expected Result: Error message displayed indicating invalid credentials
Actual Result: Error message displayed as expected
Status: Passed

3. Test Case ID: TC003
Input: Browse menu and add items to cart
Expected Result: Items are added to the cart
Actual Result: Items added to the cart as expected
Status: Passed

4. Test Case ID: TC004
Input: Modify cart (remove item)
Expected Result: Item is removed from the cart
Actual Result: Item successfully removed from the cart
Status: Passed

5. Test Case ID: TC005
Input: Place order with online payment option
Expected Result: Order is placed successfully and payment is processed
Actual Result: Order placed and payment processed successfully

Status: Passed
6. Test Case ID: TC006
Input: Place order with cash-on-delivery
Expected Result: Order is placed successfully with cash-on-delivery
Actual Result: Order placed with cash-on-delivery as expected
Status: Passed
7. Test Case ID: TC007
Input: Provide feedback on food quality
Expected Result: Feedback is submitted successfully
Actual Result: Feedback submitted successfully
Status: Passed
8. Test Case ID: TC008
Input: Check order status
Expected Result: Order status is displayed correctly
Actual Result: Order status displayed correctly
Status: Passed
9. Test Case ID: TC009
Input: View order history
Expected Result: User's order history is displayed
Actual Result: User's order history displayed correctly
Status: Passed
10. Test Case ID: TC010
Input: Update menu (add new item)
Expected Result: New item is added to the menu
Actual Result: New item added to the menu as expected
Status: Passed
11. Test Case ID: TC011
Input: Process incoming order
Expected Result: Order is processed and status is updated
Actual Result: Order processed and status updated as expected
Status: Passed
12. Test Case ID: TC012
Input: Generate sales report
Expected Result: Sales report is generated and displayed
Actual Result: Sales report generated and displayed as expected
Status: Passed
13. Test Case ID: TC013
Input: Test UI responsiveness
Expected Result: UI elements adjust properly to different screen sizes
Actual Result: UI elements adjust properly as expected
Status: Passed
14. Test Case ID: TC014
Input: Test compatibility with browsers
Expected Result: Web app functions correctly across different browsers
Actual Result: Web app functions correctly as expected
Status: Passed
15. Test Case ID: TC015
Input: Test security vulnerabilities
Expected Result: No security vulnerabilities found
Actual Result: No security vulnerabilities found
Status: Passed

7.3. TEST REPORT

1. Introduction: The project is a system designed to facilitate food ordering for students and staff within a college campus. This test report outlines the testing activities conducted to validate the functionality, usability, performance, and security of the application.

2. Test Objective: The objective of the testing is to ensure that the project meets the specified requirements and functions correctly across different scenarios. This includes verifying user authentication, menu browsing, order placement, payment handling, feedback submission, and system management functionalities.

3. Test Scope: The scope of the testing covers all major features and functionalities of the Campus Dining Web App, including frontend and backend components. It includes testing across different user roles (students, staff, admins) and scenarios such as normal usage, edge cases, and error handling.

4. Test Environment

•Operating System: Windows 10
•Web Browsers: Chrome, Firefox, Edge
•Programming Languages: Python, HTML, CSS
•Frameworks: Flask, Bootstrap
•Database: MySQL
•Tools: Selenium for automated testing, Postman for API testing

5. Test Result: The testing activities were conducted systematically, covering various aspects of the application. The test cases were executed, and the results were recorded as follows:

•Passed: All expected results matched the actual results, indicating that the system functions correctly.
•Failed: Deviations or discrepancies were found between the expected and actual results, indicating potential issues or defects in the system.
•Pending: Some test cases may require further investigation or validation before a conclusive result can be determined.
6. Test Conclusion: Overall, the project performed well during testing, with the majority of test cases passing successfully. Any identified issues or defects were documented and will be addressed by the development team. The test results indicate that the system is ready for deployment, with confidence in its functionality, usability, and reliability

1.Performance

• Transaction Time: Borrowing a book took - 3 seconds whereas manually it takes more than 20+ seconds.
• Scalability: Handling up to 500+ books and 50+ users, validated via testing.

2.Usability

Goal is to design a natural, effective, and user-friendly interface that benefits both librarians and users. The app gives the advantage of searching for books, checking for availability, reserving titles, and looking at due dates right from their smartphones. The UI is minimal and clean, with easy navigation and well-annotated icons or menus to provide ease of use even to non-technical users.
The Library Management System based on Android Studio efficiently computerizes library operations with mobility, efficiency, and scalability. It minimizes manual effort by 70% (based on transaction time comparison) and maximizes user engagement through real-time functionality. The adoption of Kotlin and Firebase demonstrates best practices in sync with today's technologies, ensuring the system is future ready.

VIII. APPENDIX

8.1. SOURCE CODE

Packages

```
from flask import Flask, render_template, redirect, request, session, url_for
import datetime
import os
from werkzeug.utils import secure_filename
from flask import send_from_directory, abort
import mysql.connector
import uuid
Register
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
email=request.form['email']
username=request.form['username']
password=request.form['password']
now = datetime.datetime.now()
date_join=now.strftime("%d-%m-%Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT count(*) FROM fo_staff where username=%s",(username, ))
cnt = mycursor.fetchone()[0]
if cnt==0:
mycursor.execute("SELECT max(id)+1 FROM fo_staff")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_staff(id, name, mobile, email, dept, username, password, date_join) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, mobile, email, dept, username, password, date_join)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
msg="fail"
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
email=request.form['email']
username=request.form['username']
password=request.form['password']
now = datetime.datetime.now()
date_join=now.strftime("%d-%m-%Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT count(*) FROM fo_stu where username=%s",(username, ))
```

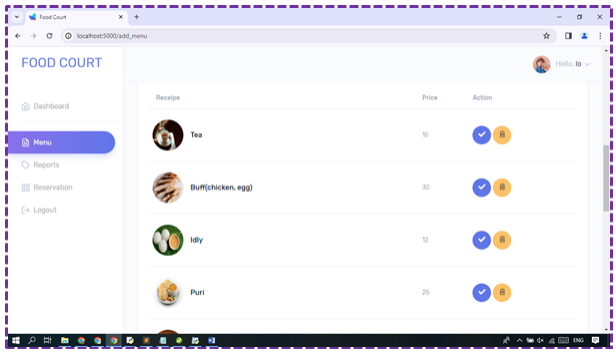
```
cnt = mycursor.fetchone()[0]
if cnt==0:
mycursor.execute("SELECT max(id)+1 FROM fo_stu")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_stu(id, name, mobile, email, dept, username, password, date_join) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, mobile, email, dept, username, password, date_join)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
msg="fail"
msg=""
if request.method=='POST':
name=request.form['name']
address=request.form['address']
mobile=request.form['mobile']
email=request.form['email']
username=request.form['username']
password=request.form['password']
now = datetime.datetime.now()
date_join=now.strftime("%d-%m-%Y")
if 'profile' in request.files:
profile = request.files['profile']
if profile and allowed_file(profile.filename):
filename = secure_filename(profile.filename)
profile_path = 'D:/kalirajan/Food_court/static/license/' + filename
profile.save(profile_path)
mycursor = mydb.cursor()
mycursor.execute("SELECT count(*) FROM fo_canteen where username=%s",(username, ))
cnt = mycursor.fetchone()[0]
if cnt==0:
mycursor.execute("SELECT max(id)+1 FROM fo_canteen")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_canteen(id, name, address, mobile, email, profile, username, password, date_join) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, address, mobile, email, filename, username, password, date_join)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
msg="fail"
Add foods
username=session.get('username')
cursor=mydb.cursor()
cursor.execute("SELECT * FROM fo_food")
data = cursor.fetchall()
cursor.close()
msg=""
if request.method=='POST':
rec_name=request.form['rec_name']
price=request.form['price']
rec_type=request.form['rec_type']
rec_category=request.form['rec_category']
now = datetime.datetime.now()
date_join=now.strftime("%Y-%m-%d")
if 'profile' in request.files:
profile = request.files['profile']
if profile and allowed_file(profile.filename):
filename = secure_filename(profile.filename)
profile_path = 'D:/kalirajan/Food_court/static/food/' + filename
profile.save(profile_path)
mycursor=mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fo_food")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_food(id, rec_name, price, rec_type, rec_category, profile, date_join) VALUES (%s, %s, %s, %s, %s, %s, %s)"
val = (maxid, rec_name, price, rec_type, rec_category, filename, date_join)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
msg="fail"
View menu
mag=""
username=session.get('username')
cursor=mydb.cursor()
```

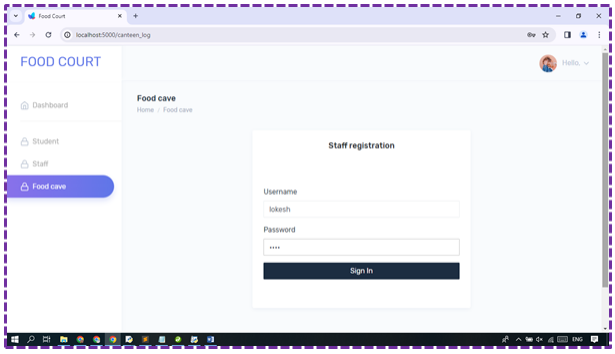
```
cursor.execute("SELECT * FROM fo_staff where username=%s", (username,))
st = cursor.fetchone()
cursor.close()
data1=""
cursor=mydb.cursor()
cursor.execute("SELECT * FROM fo_category")
data = cursor.fetchall()
cursor.close()
type1 = request.args.get('type')
if type1:
print("Received 'type1' value:", type1)  # Debugging: Print the received value
cursor = mydb.cursor()
cursor.execute("SELECT * FROM fo_food WHERE rec_type=%s", (type1,))
data1 = cursor.fetchall()
cursor.close()
else:
print("No 'type1' value received.")  # Debugging: Indicate if no value is received
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
type1=request.form['type']
rec_name=request.form['rec_name']
price=request.form['price']
quantity=request.form['quantity']
now = datetime.datetime.now()
date=now.strftime("%B %d, %Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fo_book")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_book(id, name, dept, mobile, type, rec_name, price, quantity, date, username) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, dept, mobile, type1, rec_name, price, quantity, date, username)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
msg="fail"
Orders
msg=""
username=session.get('username')
cursor=mydb.cursor()
cursor.execute("SELECT * FROM fo_stu where username=%s", (username,))
user = cursor.fetchone()
cursor.close()
dataa=""
cursor=mydb.cursor()
cursor.execute("SELECT * FROM fo_category")
data1 = cursor.fetchall()
cursor.close()
food = request.args.get('item')
if food:
print("Received 'type1' value:", food)  # Debugging: Print the received value
cursor = mydb.cursor()
cursor.execute("SELECT * FROM fo_food WHERE rec_type=%s", (food,))
dataa = cursor.fetchall()
cursor.close()
else:
print("No 'type1' value received.")
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
type1=request.form['type']
rec_name=request.form['rec_name']
price=request.form['price']
quantity=request.form['quantity']
now = datetime.datetime.now()
date=now.strftime("%B %d, %Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fo_book")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_book(id, name, dept, mobile, type, rec_name, price, quantity, date, username) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, dept, mobile, type1, rec_name, price, quantity, date, username)
mycursor.execute(sql, val)
mydb.commit()
msg="success"
else:
```

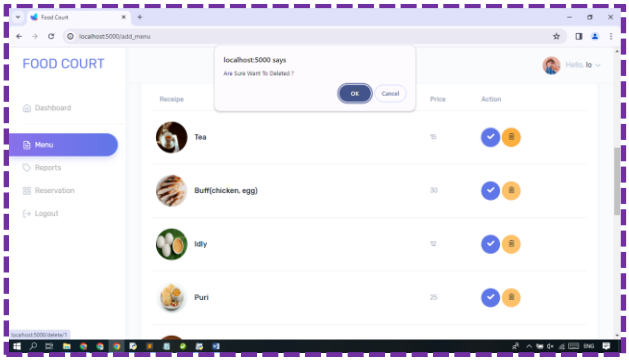
```
msg="fail"
Reports
username=session.get('username')
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
type1=request.form['type']
rec_name=request.form['rec_name']
price=request.form['price']
quantity=request.form['quantity']
date=request.form['date']
time=request.form['time']
now = datetime.datetime.now()
book_date=now.strftime("%B %d, %Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fo_table")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_table(id, name, dept, mobile, type, rec_name, price, quantity, date, time, book_date, username) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, dept, mobile, type1, rec_name, price, quantity, date, time, book_date, username)
mycursor.execute(sql, val)
mydb.commit()
username=session.get('username')
if request.method=='POST':
name=request.form['name']
dept=request.form['dept']
mobile=request.form['mobile']
type1=request.form['type']
rec_name=request.form['rec_name']
price=request.form['price']
quantity=request.form['quantity']
date=request.form['date']
time=request.form['time']
now = datetime.datetime.now()
book_date=now.strftime("%B %d, %Y")
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fo_table")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fo_table(id, name, dept, mobile, type, rec_name, price, quantity, date, time, book_date, username) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid, name, dept, mobile, type1, rec_name, price, quantity, date, time, book_date, username)
mycursor.execute(sql, val)
mydb.commit()
no=""
user_mobile=""
user_name=""
mess=""
tab1 = request.args.get('type')
cursor=mydb.cursor()
cursor.execute("SELECT * FROM fo_table WHERE type=%s", (tab1,))
data = cursor.fetchall()
cursor.close()
act=request.args.get("act")
if act=="ok":
aid=request.args.get("aid")
cursor = mydb.cursor()
cursor.execute("update fo_table set action=1 where id=%s",(aid,))
mydb.commit()
print("successfully accepted")
no="1"
aid=request.args.get("aid")
cursor = mydb.cursor()
cursor.execute("SELECT * FROM fo_table where id = %s", (aid,))
data1 = cursor.fetchone()
Mysql connection
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
charset="utf8",
use_pure=True,
database="food_court"
```

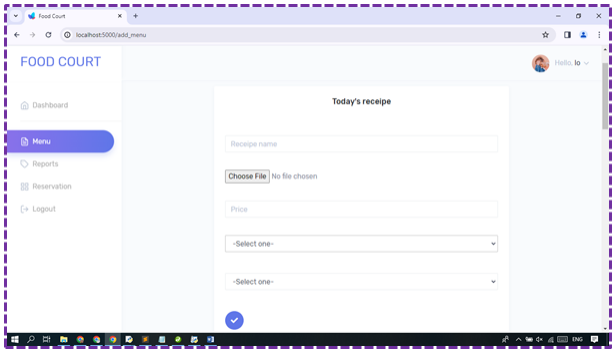
8.2. SCREENSHOTS

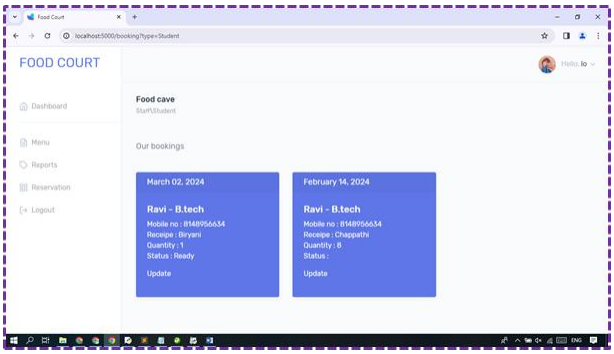

Food court
Login
Welcome To Food Court
when looking at its layout. The point of using Lorem Ipsum

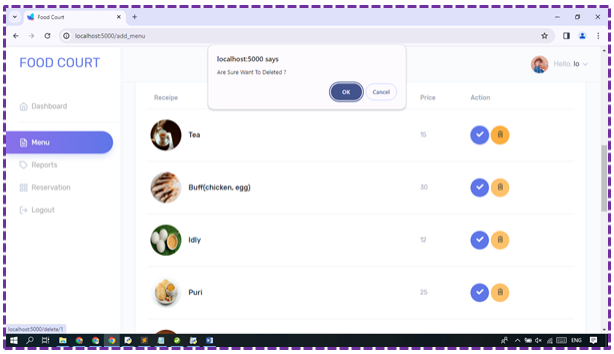

FOOD COURT
Dashboard
Menu
Reports
Reservation
Logout
Receipe
Price
Action
Tea
Buff(chicken, egg)
Idly
Puri

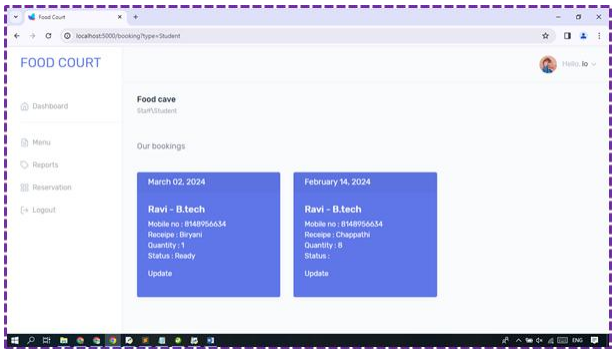

FOOD COURT
Dashboard
Student
Staff
Food cave
Food cave
Staff registration
Username
lokesh
Password
Sign In

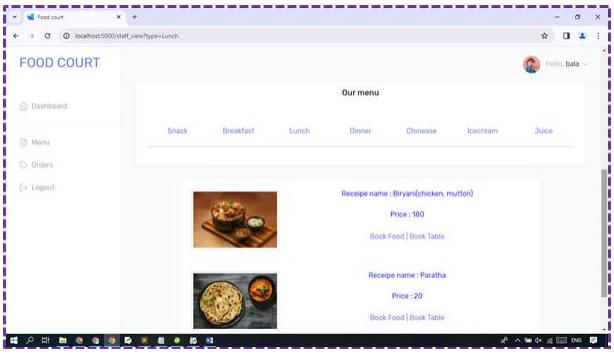

FOOD COURT
localhost:5000 says
Are Sure Want To Deleted ?
OK
Cancel
Dashboard
Menu
Reports
Reservation
Logout
Receipe
Price
Action
Tea
Buff(chicken, egg)
Idly
Puri

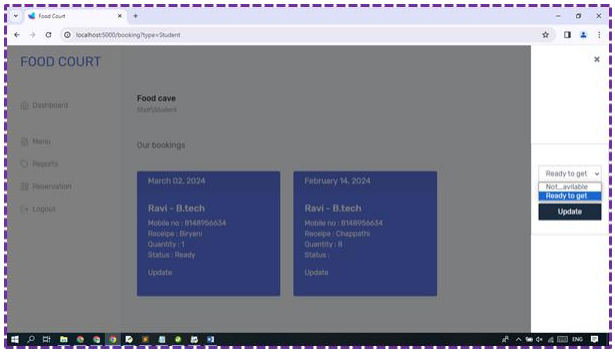

FOOD COURT
Dashboard
Menu
Reports
Reservation
Logout
Today's receipe
Receipe name
Choose File
No file chosen
Price
-Select one-
-Select one-

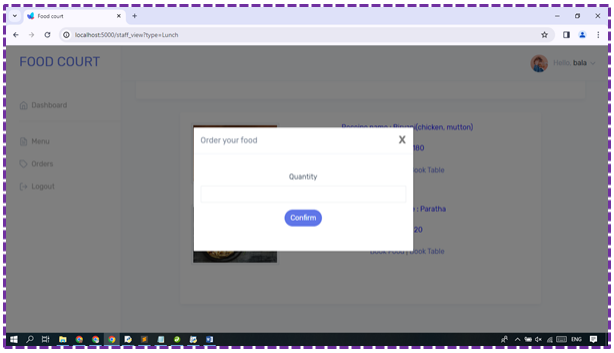

FOOD COURT
Dashboard
Menu
Reports
Reservation
Logout
Food cave
Staff\Student
Our bookings
March 02, 2024
Ravi - B.tech
Mobile no : 8148956634
Receipe : Biryani
Quantity : 1
Status : Ready
Update
February 14, 2024
Ravi - B.tech
Mobile no : 8148956634
Receipe : Chappathi
Quantity : 8
Status :
Update

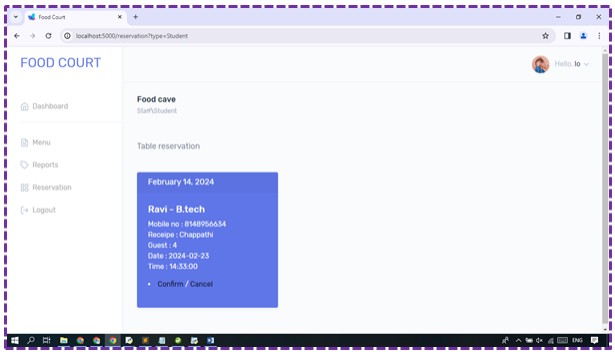

FOOD COURT
localhost:5000 says
Are Sure Want To Deleted ?
OK
Cancel
Dashboard
Menu
Reports
Reservation
Logout
Receipe
Price
Action
Tea
Buff(chicken, egg)
Idly
Puri

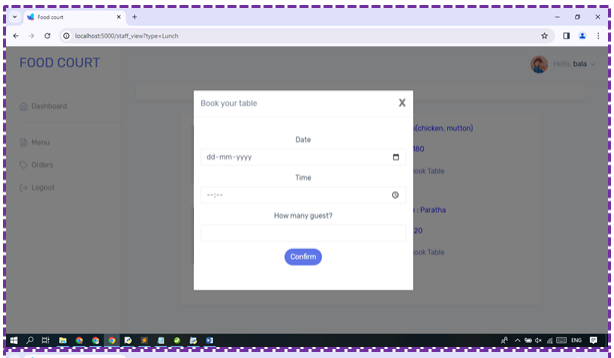

FOOD COURT
Food cave
Our bookings
March 02, 2024
Ravi - B.tech
Mobile no : 8148956634
Receipe : Biryani
Quantity : 1
Status : Ready
Update
February 14, 2024
Ravi - B.tech
Mobile no : 8148956634
Receipe : Chappathi
Quantity : 8
Status :
Update

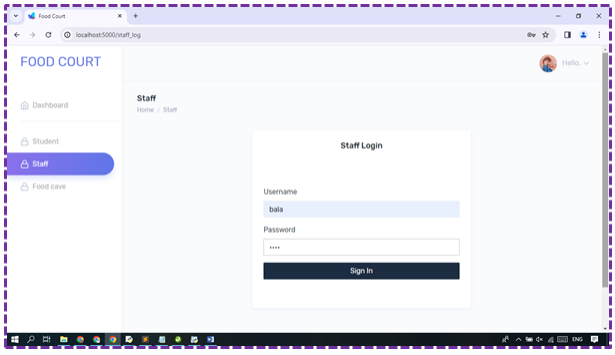

FOOD COURT
Our menu
Snack
Breakfast
Lunch
Dinner
Chinease
Icecream
Juice
Receipe name : Biryani(chicken, mutton)
Price : 180
Book Food | Book Table
Receipe name : Paratha
Price : 20
Book Food | Book Table

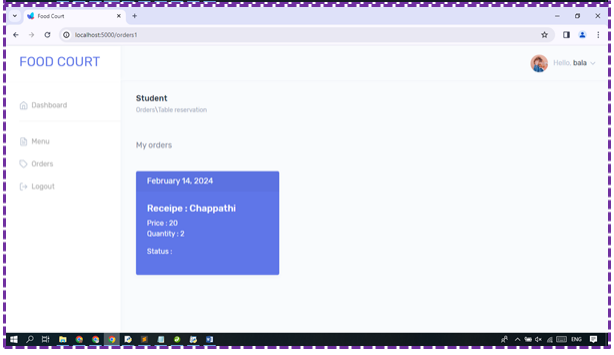

FOOD COURT
Food cave
Our bookings
Ready to get
Not_avilable
Ready to get
Update

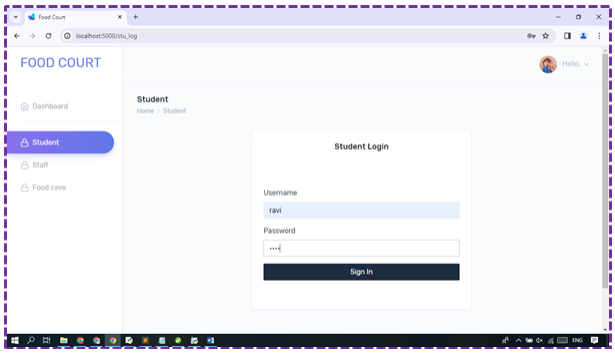

FOOD COURT
Order your food
Quantity
Confirm

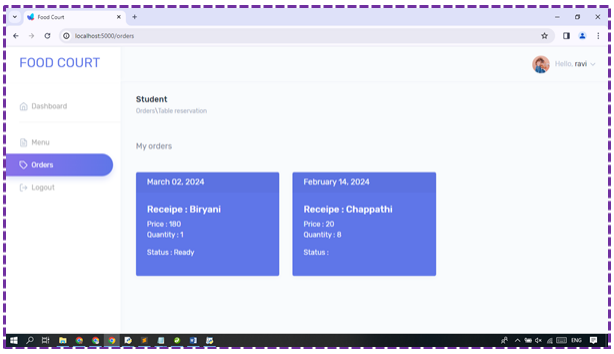

FOOD COURT
Food cave
Table reservation
February 14, 2024
Ravi - B.tech
Mobile no : 8148956634
Receipe : Chappathi
Guest : 6
Date : 2024-02-23
Time : 14:33:00
Confirm / Cancel

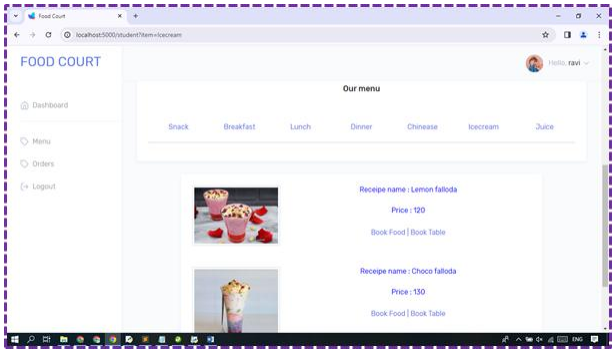

FOOD COURT
Book your table
Date
dd-mm-yyyy
Time
How many guest?
Confirm

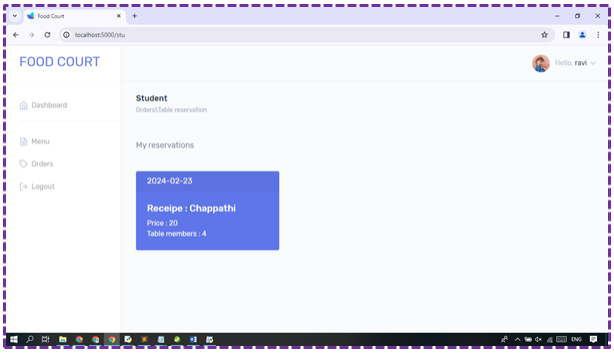

FOOD COURT
Staff
Staff Login
Username
bala
Password
Sign In

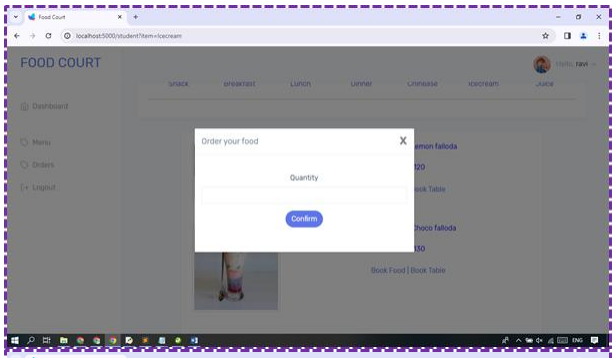

FOOD COURT
Student
My orders
February 14, 2024
Receipe : Chappathi
Price : 20
Quantity : 2
Status :

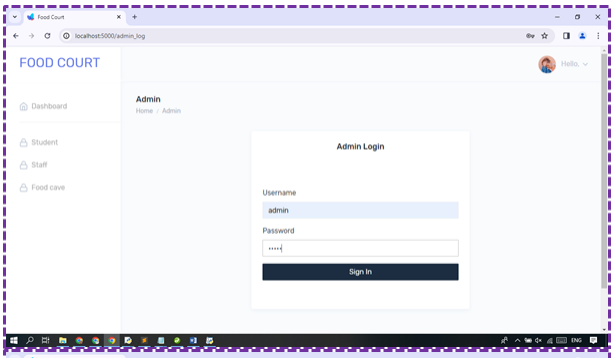

FOOD COURT
Student
Student Login
Username
ravi
Password
Sign In

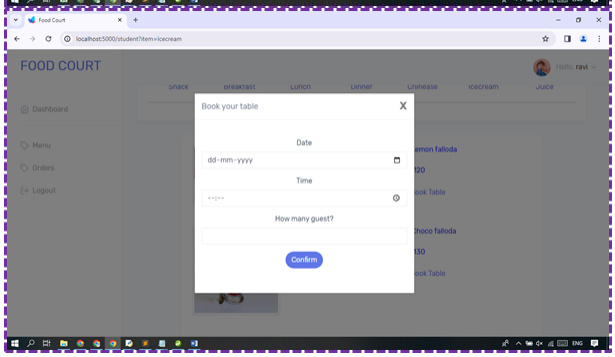

FOOD COURT
Student
My orders
March 02, 2024
Receipe : Biryani
Price : 180
Quantity : 1
Status : Ready
February 14, 2024
Receipe : Chappathi
Price : 20
Quantity : 8
Status :

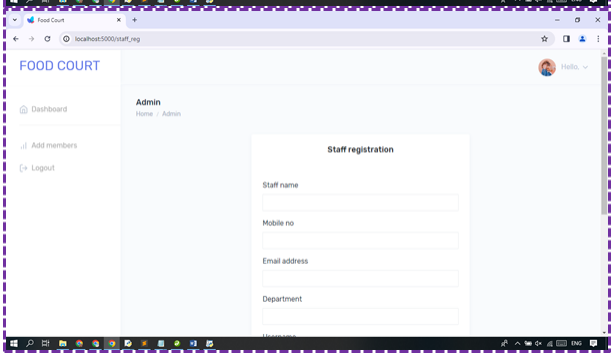

FOOD COURT
Our menu
Snack
Breakfast
Lunch
Dinner
Chinease
Icecream
Juice
Receipe name : Lemon falloda
Price : 120
Book Food | Book Table
Receipe name : Choco falloda
Price : 130
Book Food | Book Table

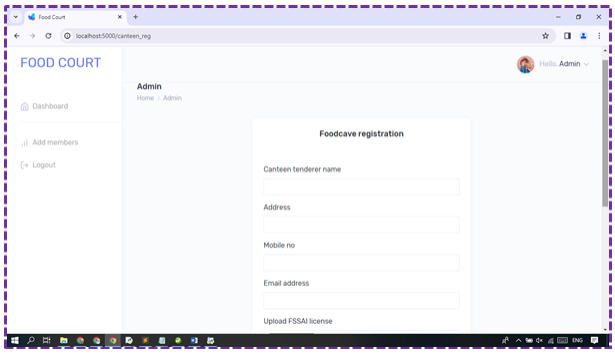

FOOD COURT
Student
My reservations
2024-02-23
Receipe : Chappathi
Price : 20
Table members : 4


FOOD COURT
Order your food
Quantity
Confirm


FOOD COURT
Admin
Admin Login
Username
admin
Password
Sign In


FOOD COURT
Book your table
Date
dd-mm-yyyy
Time
How many guest?
Confirm


FOOD COURT
Admin
Staff registration
Staff name
Mobile no
Email address
Department



IX. CONCLUSION

In conclusion, the development of the project has been a significant endeavor aimed at enhancing the food ordering experience for students, staff, and administrators within the college campus. Through meticulous planning, design, implementation, and testing, we have successfully created a robust and user-friendly platform that fulfills the needs and requirements of our users. The project has involved the collaboration of multidisciplinary teams, including developers, designers, testers, and stakeholders, who have worked tirelessly to ensure the success of the application. By leveraging technologies such as Python, Flask, MySQL, Bootstrap, and WampServer, we have built a scalable and efficient solution that meets the demands of our users while providing a seamless experience across different devices and browsers. The project offers a wide range of features, including user authentication, menu browsing, order placement, payment handling, feedback submission, and system management, all of which have been thoroughly tested and validated to ensure reliability and functionality. Additionally, the application incorporates security measures to protect user data and privacy, as well as performance optimization techniques to ensure a smooth user experience even under heavy load. Overall, the completion of the Campus Dining Web App represents a significant milestone in our efforts to modernize and streamline the food ordering process within the college campus. We are confident that the application will greatly benefit our users by providing them with a convenient and efficient way to order food, ultimately enhancing their overall college experience. We are excited about the potential impact of the Campus Dining Web App and look forward to its successful deployment and adoption by the college community.

X. FUTURE ENHANCEMENT

In the future, the project can undergo several enhancements to elevate its functionality and user experience. One pivotal advancement could be the development of a mobile application, enabling users to conveniently place orders from their smartphones, thereby enhancing accessibility and engagement. Additionally, implementing a promotions and rewards system could incentivize user interaction and loyalty by offering discounts, coupons, and loyalty points for recurring orders or referrals. Furthermore, incorporating multi-language support would cater to a broader user base, ensuring inclusivity and accessibility for individuals from diverse linguistic backgrounds. These enhancements collectively aim to fortify the Campus Dining experience, fostering user satisfaction and engagement.

REFERENCES

1. M. C. Boliko, "FAO and the situation of food security and nutrition in the world", Journal of nutritional science and vitaminology, no. 65, pp. S4-S8, 2019.
2. B. Garske, K. Heyl, F. Ekardt, L.M. Weber and W. Gradzka, "Challenges of food waste governance: An assessment of European legislation on food waste and recommendations for improvement by economic instruments", MDPI Land, vol. 9, no. 7, pp. 231, 2020.
3. A. C. Stenmarck, T. Jensen, T. Quested and G. Moates, "FUSIONS Reducing Food Waste through Social Innovation", Full Report. IVL Swedish Environmental Research Institute; Estimates of European food waste levels, 2016.
4. C.V. Khoie and A. Soletti, "Nutritional status of elderly in the old age homes: A study in Pune city", Current Research in Nutrition and Food Science Journal, vol. 6, no. 1, pp. 234-240, 2018.
5. D. Lovesley, R. Parasuraman and A. Ramamurthy, "Combating hospital malnutrition: Dietitian-led

quality improvement initiative", Clinical nutrition ESPEN, vol. 30, pp. 19-25, 2019.

6. S.J. GeetinderKaur and G. Singh, "Food Sustainability Using Wireless Sensors Networks: Waspmote and Meshlium", (IJCSIT) International Journal of Computer Science and Information Technologies, vol. 5, no. 3, pp. 4466-446, 2014.
7. N. Salim, S. Zeebaree, M. Sadeeq, A. Radie, M. Shukur and N. Rashid, "Study for Food Recognition System Using Deep Learning", Journal of Physics: Conference Series 2nd International Conference on Physics and Applied Sciences (ICPAS 2021), vol. 1963, May 2021.
8. G. Ciocca, G. Micali and P. Napoletano, "State Recognition of Food Images Using Deep Features", IEEE Access, vol. 8, pp. 32003-32017, 2020.
9. P. Pouladzadeh and S. Shirmohammadi, "Mobile Multi-Food Recognition Using Deep Learning", ACM Transactions on Multimedia Computing Communications and Applications, vol. 13, no. 3s, pp. 1-21, August 2017.
10. K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning", 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 1-6, 2015.