CASHLESS CANTEEN AUTOMATION SYSTEM

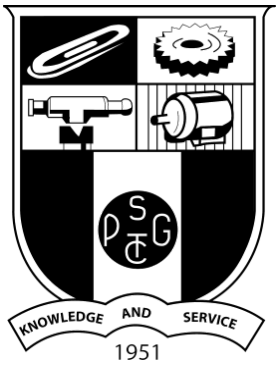
BRINDHA S V **23MX105**
MANASA DEVI KRISHNA T **23MX116**
PRIYADHARSHINI A **23MX223**

23MX18 Web Application Development

REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY



/

DECEMBER 2023

DEPARTMENT OF COMPUTER APPLICATIONS
PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

COIMBATORE - 641 004

CASHLESS CANTEEN AUTOMATION SYSTEM

Bonafide record of work done by

BRINDHA S V	**23MX105**
MANASA DEVI KRISHNA T	**23MX116**
PRIYADHARSHINI A	**23MX223**

23MX18 Web Application Development

Report submitted in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY

DECEMBER 2023

Faculty Guide

……

Mrs. A. KALYANI

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We immensely take this opportunity to express our sincere gratitude to **Dr. K. Prakasan**, Principal, PSG College of Technology, for providing us all the facilities within the campus for the completion of the project.

We profoundly thank **Dr. A. Chitra**, Professor and Head, Department of Computer Applications, PSG College of Technology, for her moral support and guidance.

We owe an extremely unbound gratitude and extend our thanks to our Programme Coordinator, **Dr. R. Manavalan**, Associate Professor, Department of Computer Applications, PSG College of Technology, whose motivation and support encouraged us to take up and complete this project work.

We are overwhelmed in all humbleness and gratefulness in acknowledging our subject Coordinator and Project Guide **Mrs. A. Kalyani**, Assistant Professor, Department of Computer Applications, PSG College of Technology, for her priceless suggestions and unrelenting support in all our efforts to improve our project and for piloting the right way for the successful completion of our project.

We also express our sincere thanks to all the staff members of the Department of Computer Applications for their encouragement. We also thank our parents and all the hands that helped us.

SYNOPSIS

The Cashless Canteen Automation System represents a highly advanced and innovative solution for cafeteria management in educational institutions, harnessing the power of QR code technology to revolutionize the payment process. The system aims to address the inefficiencies of the current manual payment system, particularly the long queues that form during break times, causing delays and disruptions in the academic schedule.

The primary objective of this innovative system is to streamline and modernize cafeteria operations by introducing computerized tools and comprehensive software applications. The integration of personalized QR codes and web applications plays a pivotal role in achieving this goal. Students and faculty often experience delays at payment counters, leading to tardiness for lectures. Recognizing the urgency to eliminate or reduce these waiting times, both teachers and students are eager to adopt a solution that enhances efficiency in the canteen.

The key feature of the system is the replacement of traditional cash transactions with QR code payments, offering a faster and more convenient alternative. This not only reduces waiting times but also contributes to a seamless and efficient canteen experience. Overall, the Cashless Canteen Automation System is poised to transform the way payments are made in the institution's canteen, ensuring a more streamlined and modern approach to cafeteria management.

CHAPTER 1

INTRODUCTION

The Cashless Canteen Automation System, harnessing the power of QR code technology for individual student payments, marks the significant advancement in cafeteria management across various institutions. The primary goal of this system is to streamline and modernize the current manual system by incorporating computerized tools and comprehensive software applications. In canteens, long queues form during break times, leading to significant delays from the payment counters to the serving area. This waiting period causes students and faculty to be late for their lectures. Both teachers and students are eager to find a solution to eliminate or reduce this waiting time. This innovative system integrates personalized QR codes and web applications to revolutionize payment processes and elevate canteen efficiency. By replacing traditional cash transactions with QR code payments, it reduces waiting time and enhances convenience.

1.1 Project Overview

The Cashless Canteen Automation System represents a groundbreaking initiative aimed at advancing cafeteria management in various educational institutions through the integration of QR code technology. The primary objective of this project is to address and rectify the inefficiencies inherent in the current manual system prevalent in the institution's canteen. Long queues during break times have been identified as a significant bottleneck, causing delays from payment counters to the serving area and subsequently leading to students and faculty being tardy for lectures. In response to this challenge, the project endeavors to streamline and modernize the entire process by incorporating cutting-edge computerized tools and comprehensive software applications. The core innovation lies in the integration of personalized QR codes and web applications, ushering in a new era of efficiency and convenience. This transformative system not only aims to eliminate or substantially reduce waiting times but also promises to revolutionize payment processes, providing a seamless experience for both students and faculty. By replacing traditional cash transactions with QR code payments, the project anticipates a significant enhancement in canteen efficiency and a positive impact on the overall academic schedule.

1.2 Tools and Technology

HTML5 (HyperText Markup Language 5)

HTML is a markup language for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML standard. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices.

Many reasons could make us use HTML, and here are the most five features to mention:

- **It is the future**

With all the big players embracing HTML already, it looks a safe bet that it will be the future. How it will exactly pan out, no one quite knows (or when), but at some point, it will land as a standard and is the next logical step.

- **It can help to correctly index the content in search engines**

This may not be the case now, certain tags will help such as Microformats, but Google search engines are getting smarter and smarter.

- **It can help accessibility**

As with Web Standards, just by producing a site that uses CSS for presentation, it does not mean it is accessible. HTML is the same story, just by coding in HTML doesn't mean the site is more accessible. Instead, it is another arrow in the bow to help make the site become more accessible.

- **It has bundles of new features**

HTMLdoes not just allow users to mark-up documents with meaningful semantic tags, it also offers new functionality. These are not all in HTMLspec per se but part of the new movement towards funky new web applications, hence why they are mentioned. From watching videos without having to have a plug in, native form features, the 39 canvas, application caches so information can be stored offline to geolocation. Whilst these are in no way the finished article, there is no harm being ready to embrace them

.

CSS3 (Cascading Style Sheets 3):

CSS is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVVG, MathML, XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. HTML largely determines textual content, whereas CSS determines visual structure, layout, and aesthetics.
There are number of benefits of CSS, including:

- **Pages load faster**

CSS does not require the writing of HTML tag attributes every time. There is the writing of rule just once for a tag, which can be applied to all the occurrences of the corresponding tag. So using CSS, there is less code, which means faster downloading.

- **Easier website maintenance**

CSS makes the maintenance of the website easier. If users require a global change in the file, it can be simply done by changing the style by which all the elements on the web page will update automatically. The CSS file provides a flexible look to the website, which can be altered in a convenient way. It also makes HTML formatting and modification of corresponding data elements easier.

- **Saves a lot of time**

CSS style definitions are saved in external CSS files, so it is possible to change the entire website by changing just one file. Write CSS once and then reuse the same sheet in multiple HTML pages. Then define a style for each HTML element and apply it to as many web pages as users want.

- **Platform independence**

The Script offers consistent platform independence and can support the latest browsers as well.

- **Multiple device compatibility**

CSS is compatible with the older language versions so that users can use CSS with the earlier language versions. Because of this, if the CSS application is developed with the older programming language versions and if the developer combines the same with new improvements, then CSS can be easily implemented with the corresponding changes so the developer can update the existing code successfully.

Javascript:ECMAScript 2021 (ES12)

JavaScript serves as the dynamic force behind Cashless canteen, injecting vitality into the platform with its capabilities as a client-side scripting language. As the backbone of real-time interactivity, JavaScript breathes life into dynamic dashboards for learners, fostering seamless and responsive user interactions. One of its pivotal roles is in form validation, where it ensures accuracy and reduces errors by validating user inputs in real time. Beyond this, JavaScript showcases its versatility by contributing to personalized learning experiences. Through algorithms that analyze learning preferences, the platform can offer tailored skill recommendations, elevating the engagement level and ensuring Cashless canteen is learner-centric. The multifaceted nature of JavaScript, encompassing real-time functionality, validation prowess, and personalization features, establishes it as an indispensable element in shaping an interactive and learner-focused experience on Cashless canteen.

Features of JavaScript:

- **Real-time Interactivity**

  JavaScript enables real-time updates and interactions, creating a dynamic and responsive user interface.

- **Dynamic Dashboards**

  The language is utilized to build dynamic dashboards for learners, providing real-time insights and enhancing their ability to track progress efficiently.

- **Seamless User Interactions**

  JavaScript ensures smooth and seamless user interactions, enhancing the overall learning experience.

- **Form Validation**

  Accurate form validation is achieved through JavaScript, reducing errors and improving data integrity by validating learner inputs in real time.

- **Versatility**

  JavaScript's versatility is showcased in its ability to handle a variety of tasks, from real-time updates to personalized learning experiences.This strategic use of JavaScript contributes to a dynamic, engaging, and learner-centric platform, ensuring a seamless and responsive experience for skill enhancement.The below Fig1.1 shows the diagrammatic representation of technologies used.

Fig. 1.1 Html+Css+Java

MongoDB(Version-7.0.4):

MongoDB is a popular NoSQL database management system that uses a document-oriented data model. It is designed to store, query, and process large amounts of data in a flexible, schema-free format. MongoDB is classified as a NoSQL database because it doesn't rely on the traditional relational database structure.

Features of MongoDB :

- **Document-Oriented:** Data is stored in BSON (Binary JSON) documents, which are similar to JSON objects.
- **Scalability**: MongoDB is horizontally scalable, meaning users can add more servers to handle increased load.
- **Flexible Schema:** Unlike traditional relational databases, MongoDB doesn't require a predefined schema. Fields can vary in each document.
- **Indexing:** Supports various types of indexes to improve query performance.
- **Aggregation Framework:** Provides powerful and flexible aggregation capabilities for data processing.

NodeJS(Version:18.12.0):

Node.js is a runtime environment that allows users to execute JavaScript code outside of a web browser. It is built on the V8 JavaScript runtime and enables server-side development using JavaScript. Node.js uses an event-driven, non-blocking I/O model, making it efficient for building scalable network applications.

Features of Node.js :

- **Asynchronous I/O**: Node.js is designed to handle asynchronous operations, allowing it to efficiently manage many connections simultaneously.
- **Single-Threaded:** Although the core of Node.js is single-threaded, it uses an event loop and non-blocking I/O to handle concurrent requests.
- **npm (Node Package Manager):** npm is a package manager for Node.js that simplifies the process of installing, sharing, and managing third-party libraries.
- **Cross-Platform:** Node.js is cross-platform and can run on various operating systems, including Windows, macOS, and Linux.

ExpressJS(version: 4.18.2):

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It is one of the most popular and widely used frameworks for building server-side applications in the Node.js environment.

Features of Express.js:

- **Middleware:**

Express uses middleware functions that have access to the request and response objects. This allows users to perform tasks such as logging, authentication, and data parsing.

- **Routing:**

Express provides a simple and expressive way to define routes for handling different HTTP methods and URLs.

- **Template Engines:**

Although Express does not include a template engine, it can easily integrate with popular ones like EJS, Pug, or Handlebars to render dynamic content on the server.

- **Static Files:**

Express makes it easy to serve static files, such as images, CSS, and JavaScript, using the `express.static` middleware.

MVC Architecture:

The MVC framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. The Fig.1.2 shows the MVC architecture diagram.

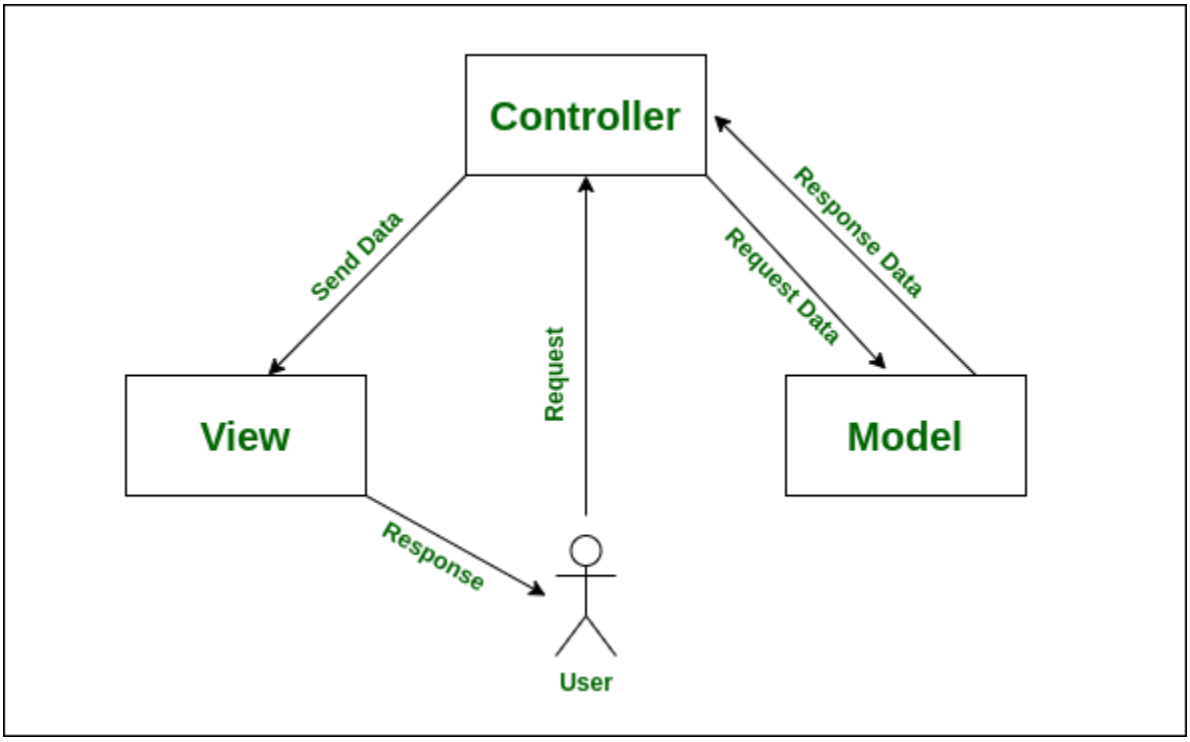


Fig 1.2 MVC architecture diagram

Visual Studio(Version:1.83.1)

Visual Studio stands as a robust integrated development environment (IDE) developed by Microsoft, providing comprehensive tools and features to streamline the software development process. Offering support for multiple programming languages such as C#, C++, and Python, Visual Studio caters to a diverse range of developers and projects. Its user-friendly interface and powerful code editing capabilities, including features like IntelliSense for auto-completion and code navigation, enhance

developer productivity. Visual Studio also facilitates collaborative development through integrated version control systems like Git. The IDE's debugging and profiling tools assist developers in identifying and rectifying issues efficiently. Moreover, Visual Studio supports the creation of a wide array of applications, including web, mobile, and desktop applications, making it a versatile and indispensable tool for software development across various domains. With its continuous updates and a rich ecosystem of extensions, Visual Studio remains a cornerstone in the toolkit of developers worldwide, contributing significantly to the creation of high-quality software. Here are some key features that contribute to making Visual Studio a preferred choice for many developers:

- **IntelliSense**

Visual Studio's IntelliSense is an intelligent code completion feature that provides context-aware suggestions as developers type, improving coding efficiency and reducing errors. It offers insights into available functions, variables, and methods, enhancing the overall coding experience.

- **Live Share**

Live Share is a real-time collaborative coding feature in Visual Studio that enables multiple developers to work on the same codebase simultaneously, regardless of their physical location. This fosters teamwork and facilitates seamless remote collaboration, enhancing productivity and code quality.

- **Integrated Debugger**

Visual Studio's integrated debugger is a robust toolset that includes breakpoints, watch windows, and step-through debugging. It allows developers to identify and resolve issues efficiently, providing a comprehensive set of features for effective debugging during the development process.

- **Git Integration**

The built-in Git support in Visual Studio streamlines version control, enabling developers to manage source code efficiently within the IDE. It simplifies collaboration by providing tools for branching, merging, and handling repositories directly from the development environment.

- **Live Server**

The live server extension is a standout feature for web developers, enabling real-time updates of web pages as code changes are made, and providing an interactive preview of web applications.

CHAPTER 2

SYSTEM ANALYSIS

This chapter discusses the analysis of the system design used in “Cashless Canteen Automation System” project. System design is the process of defining elements of a system like modules, architecture, components and their interfaces, and data for a system based on the specified requirements.

2.1 Existing System

The current canteen system in the institution combines manual cash transactions with RFID technology. Students make cash payments at designated counters, and transactions are linked to their RFID cards. Following a manual cash transaction, students swipe their RFID cards, associating the payment with their accounts. While RFID aids in identification and tracking, the manual handling of cash leads to queue formation and potential inefficiencies during peak times. Delays at payment counters can disrupt canteen operations. Despite RFID's tracking capability, maintaining real-time transparency and addressing potential errors remain challenges. The system's reliance on cash may limit user convenience as digital payment methods become more prevalent. Considering these aspects, an assessment of more advanced, streamlined, and possibly cashless systems, such as those utilizing QR codes, could offer solutions and enhance overall canteen efficiency.

Demerits of Existing System

- Queue Formation and Delays
- Inefficiencies in Manual Processes
- Limited Transparency
- Risk of Errors and Discrepancies
- Security Concerns
- Dependency on Cash
- Adaptability to Modern Trends
- Tracking Limitations

2.2 Proposed System

The core objective of this initiative is to modernize and streamline the existing manual payment system prevalent in the institution's canteen. The conventional approach has resulted in long queues during break times, causing significant delays from payment counters to the serving area. This delay, in turn, disrupts the schedules of both students and faculty, leading to tardiness in attending lectures.Recognizing the urgency to address this issue, the proposed system integrates cutting-edge computerized tools and comprehensive software applications. The cornerstone of this innovation lies in the incorporation of personalized QR codes and web applications, designed to revolutionize payment processes and elevate overall canteen efficiency.The traditional cash transactions will be replaced with QR code payments, aiming to drastically reduce waiting times and enhance convenience for both students and faculty. This transformative approach is poised to eliminate or significantly diminish the prolonged queues during peak hours, ensuring a more seamless and time-efficient experience for everyone involved.This system leverages the prowess of QR code technology to redefine the way individual student payments are handled, representing a pivotal advancement in cafeteria operations.

Advantages of Proposed System

- Enhanced Efficiency
- Convenience for Users
- Time Savings
- Real-time Transparency
- Enhanced Security
- Adaptability to Trend

2.3 Functional Requirements

Home Page:

This page contains the description about the canteen website and has a Login through which the customers are taken to the Login module.

Login Module :

The Login functional module is designed to manage user authentication and access control within the canteen website. Its primary purpose is to verify the identity of users and grant them appropriate access based on their credentials. The module ensures that only authorized users can log in to the website, and they Login to the system through their customer Id.

Customer portal Module :

The Customer portal Module is responsible for managing account details of the customers to store their bank information to the collection and for generation of QR codes for their particular amount entered and then on scanning , the amount is debited from the bank automatically.

Payment Module :

The Payment module supports the scanning of QR codes at the canteen's point of sale (POS) devices.The scanned QR code should identify the user and deduct the appropriate amount from their account. Payments should be processed in real-time to ensure immediate and accurate deduction from the user's account.

2.4 Non Functional Requirements

Performance Requirements:

Response Time: Specify that the system should provide a quick response time for QR code transactions to ensure efficient and timely processing at the payment counters.

Throughput: Define the number of transactions the system should handle per minute during peak hours to maintain optimal performance.

Security:

Data Encryption: Specify the need for end-to-end encryption of transaction data to ensure the security and privacy of student payment information.

Authentication: Define robust authentication mechanisms to prevent unauthorized access to the QR code generation and payment functionalities.

Reliability and Availability:

System Uptime: Specify a high percentage of system uptime, ensuring that the cashless canteen system is available to students and faculty during canteen operating hours.

Mean Time Between Failures (MTBF): Define the average time the system should operate without failures, ensuring reliability in day-to-day operations.

Usability:

User Interface (UI) Consistency: Specify that the web applications should provide a consistent and intuitive user interface for both students and canteen staff.

Accessibility: Define the system's accessibility features to ensure that it can be easily used by individuals with varying levels of technological expertise.

Scalability:

Horizontal Scaling: Specify the system's ability to scale horizontally by accommodating an increasing number of users and transactions, ensuring it can handle growth.

Vertical Scaling: Define the system's ability to scale vertically, ensuring it can handle additional features or increased functionality in the future.

CHAPTER 3
SYSTEM DESIGN

System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. The design activities are of main importance in this phase, because in these activities decisions ultimately affect the success of the software implementation and its maintenance is made. This decision has the final bearing on the reliability and maintainability of a system.

3.1 System Flow Diagram

A system flow diagram, also known as a flowchart, is a graphical representation of the steps or flow of a process or algorithm in software engineering. It is a visual tool used to understand, analyze, and communicate the sequence of activities or decisions involved in a software system. Flow diagrams help in visualizing the overall structure and logic of a software system, making it easier to identify bottlenecks, potential issues, and areas for optimization. The system flow diagram for the cashless canteen automation system is shown in Fig. 3.1.

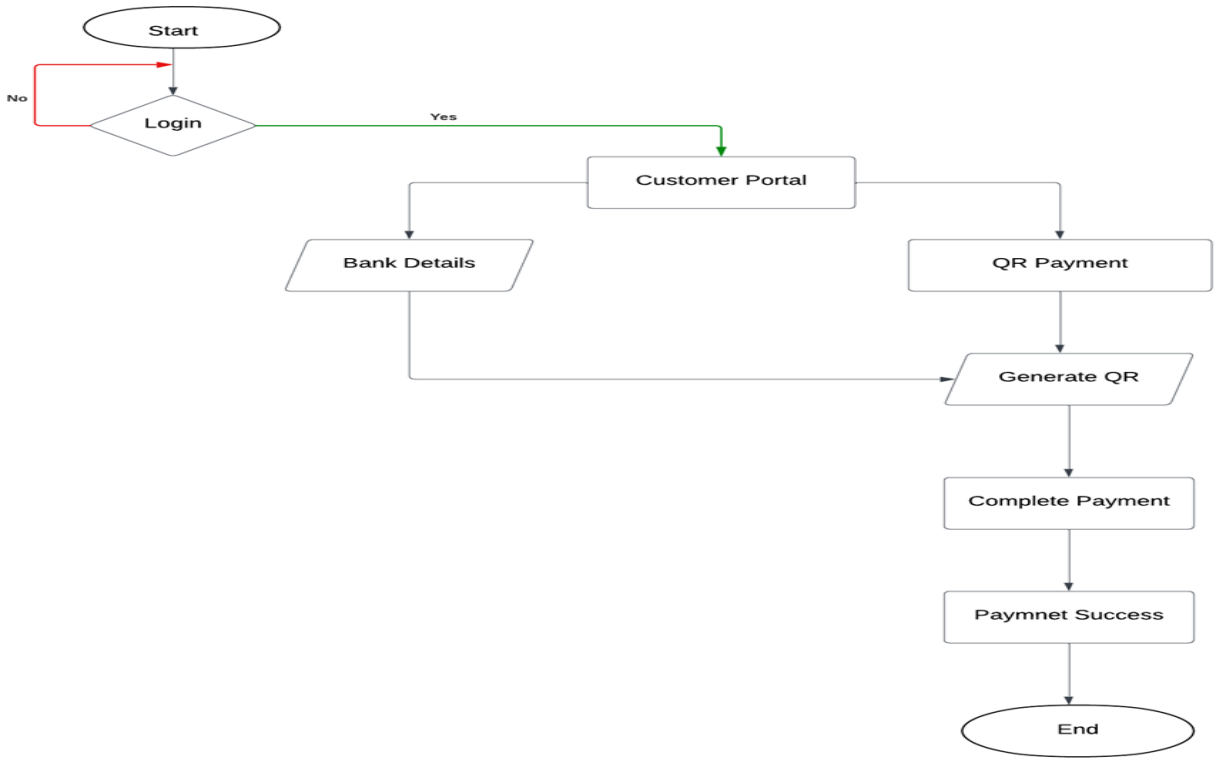


Fig. 3.1 System Flow Diagram for Cashless Canteen

3.2 ER diagram

An Entity-Relationship (ER) diagram is a visual representation of the data model that illustrates the relationships between entities in a database system. It serves as a powerful tool in database design, providing a clear and concise depiction of how different entities relate to each other and the nature of these relationships.

The primary components of an ER diagram include entities, which represent real-world objects or concepts, attributes that define the properties of these entities, and relationships that depict the associations between entities. Entities are typically nouns, attributes are the characteristics or properties of entities, and relationships indicate how entities interact. The diagram uses symbols such as rectangles for entities, ovals for attributes, and diamond shapes for relationships, facilitating a standardized and easily understandable visual language for both database designers and stakeholders.

ER diagrams are instrumental in the initial stages of database development, aiding in communication between designers and stakeholders, and laying the groundwork for the subsequent creation of a well-structured and efficient relational database. The ER diagram is shown in Fig. 3.2.

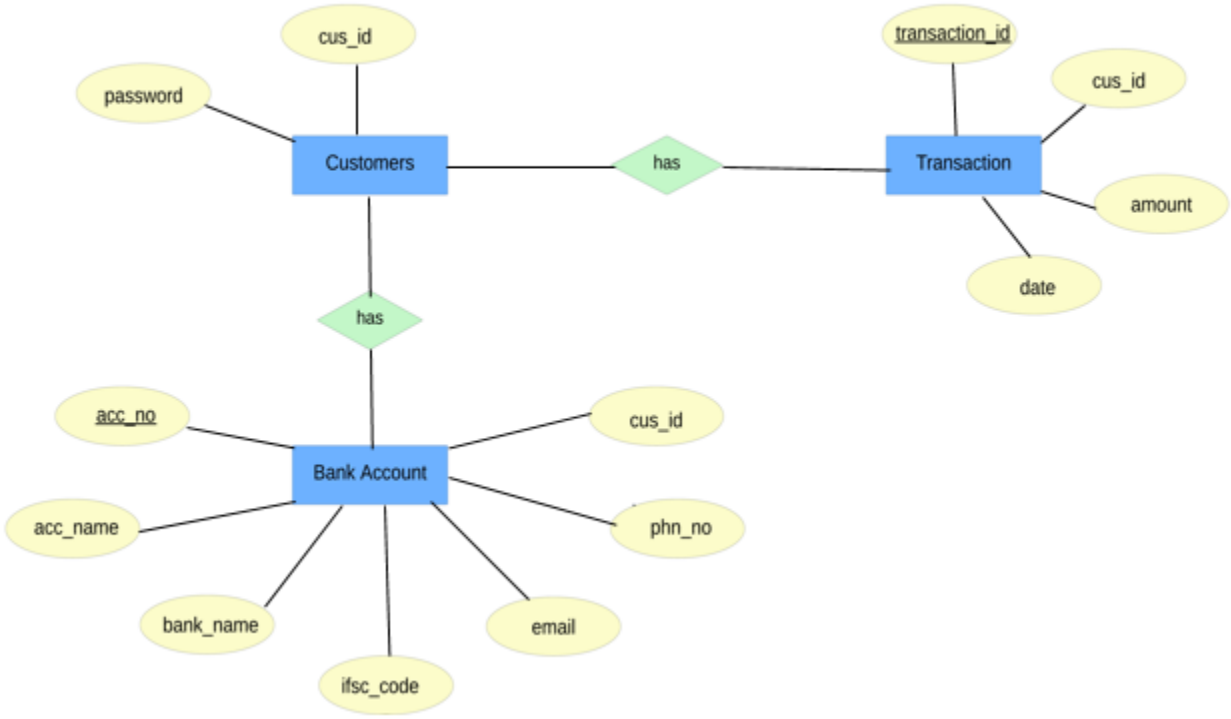


Fig. 3.2 ER Diagram for Cashless Canteen

3.3 Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

Collections used in Application

A collection is a grouping of documents. Documents within a collection can have different fields. The database schema is shown in Fig. 3.3.

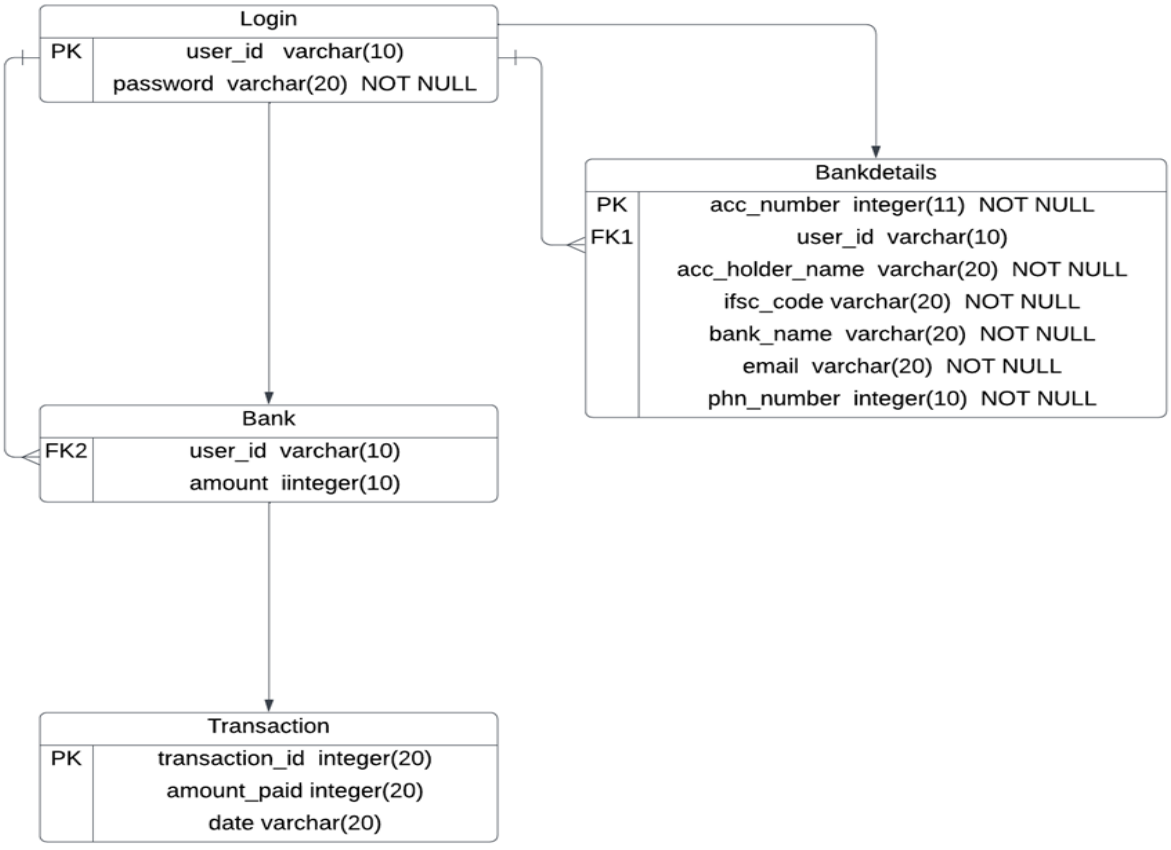


Fig. 3.3 Database schema for Cashless Canteen

A collection is the equivalent of a table in a relational database system. A collection exists within a single database.

CHAPTER 4

SYSTEM IMPLEMENTATION

This chapter discusses the system implementation. System implementation is a set of procedures performed to complete the design contained in the systems design document.

4.1 Design Approach-MVC:

The Fig.4.1 shows the MVC design approach for the cashless canteen system.

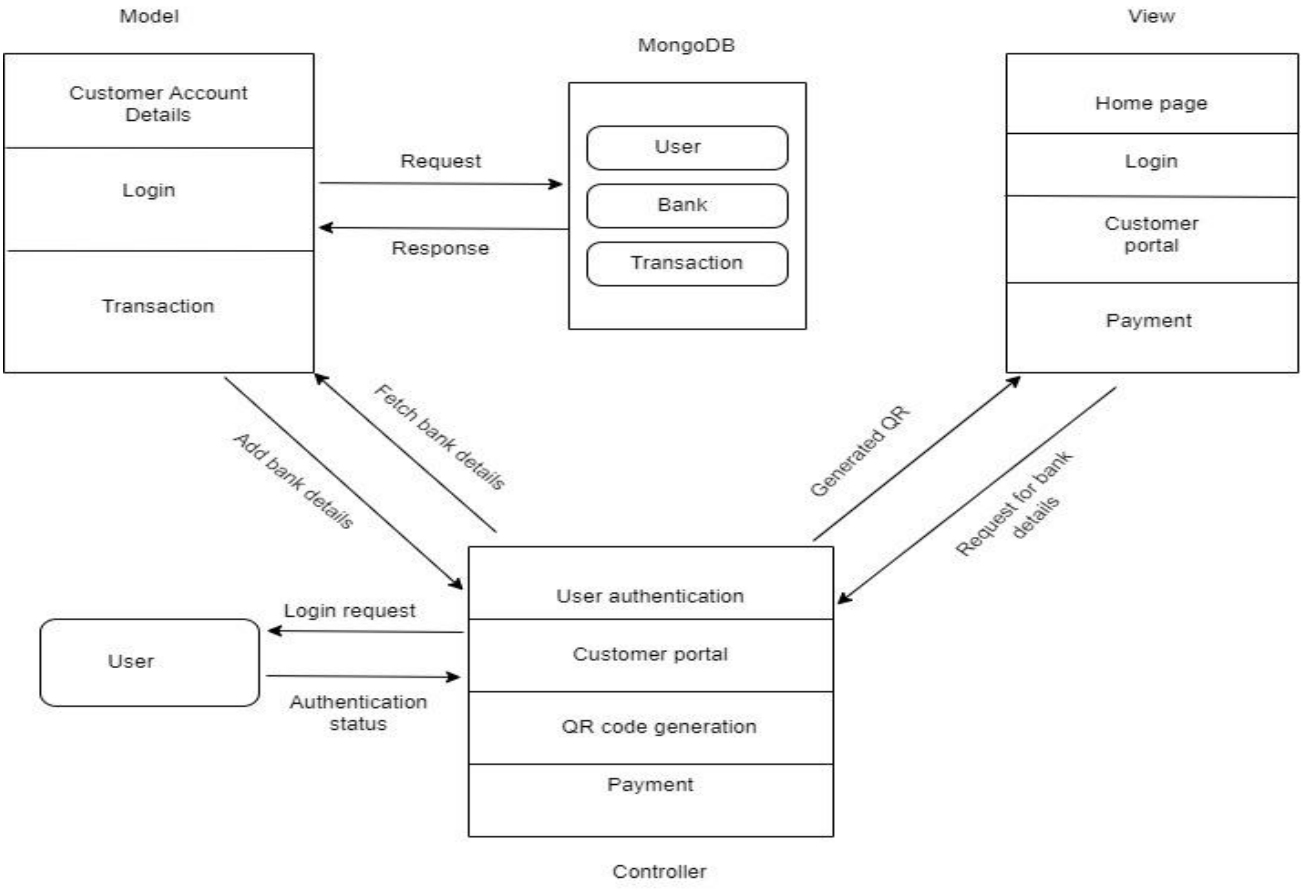


Fig 4.1 MVC design approach

4.2 Home Page:

This module contains the description about the canteen website and has a Login through which the customers are taken to the Login module. The Fig. 4.2 shows the home page of the canteen system.

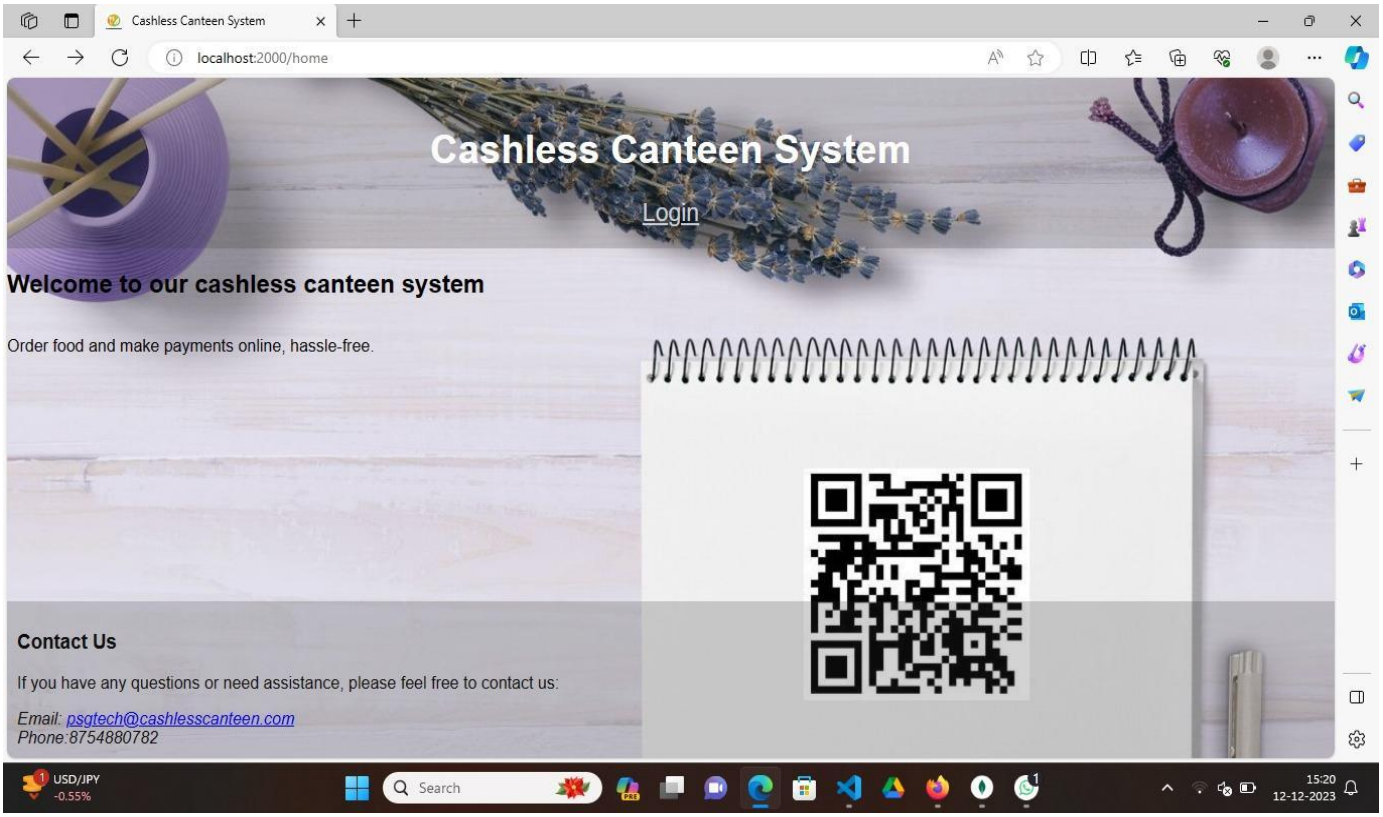


Fig. 4.2 Home Page

Implementation of Home page module:

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="icon" type="image/png" href="/css/favicon.jpg">
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet"  href="/home.css">
<title>Cashless Canteen System</title>
</head>
```

```
<body>
<header>
<h1>Cashless Canteen System</h1>
<nav>
<a href="/login">Login</a>
</nav>
</header>
<h2>Welcome to our cashless canteen system</h2>
<p>Order food and make payments online, hassle-free.</p>
<footer>
<h3>Contact Us</h3>
<p>If you have any questions or need assistance, please feel free to
contact us:</p>
<address>
Email: <a href="mailto:psgtech@cashlesscanteen.com">
psgtech@cashlesscanteen.com</a><br>
Phone:8754880782
</address>
</footer>
</body>
</html>
```

4.3 Login Module :

The Login functional module is designed to manage user authentication and access control within the canteen website. Its primary purpose is to verify the identity of users and grant them appropriate access based on their credentials. The module ensures that only authorized users can log in to the website, and they Login to the system through their customer Id. The Fig. 4.3 shows the login page.

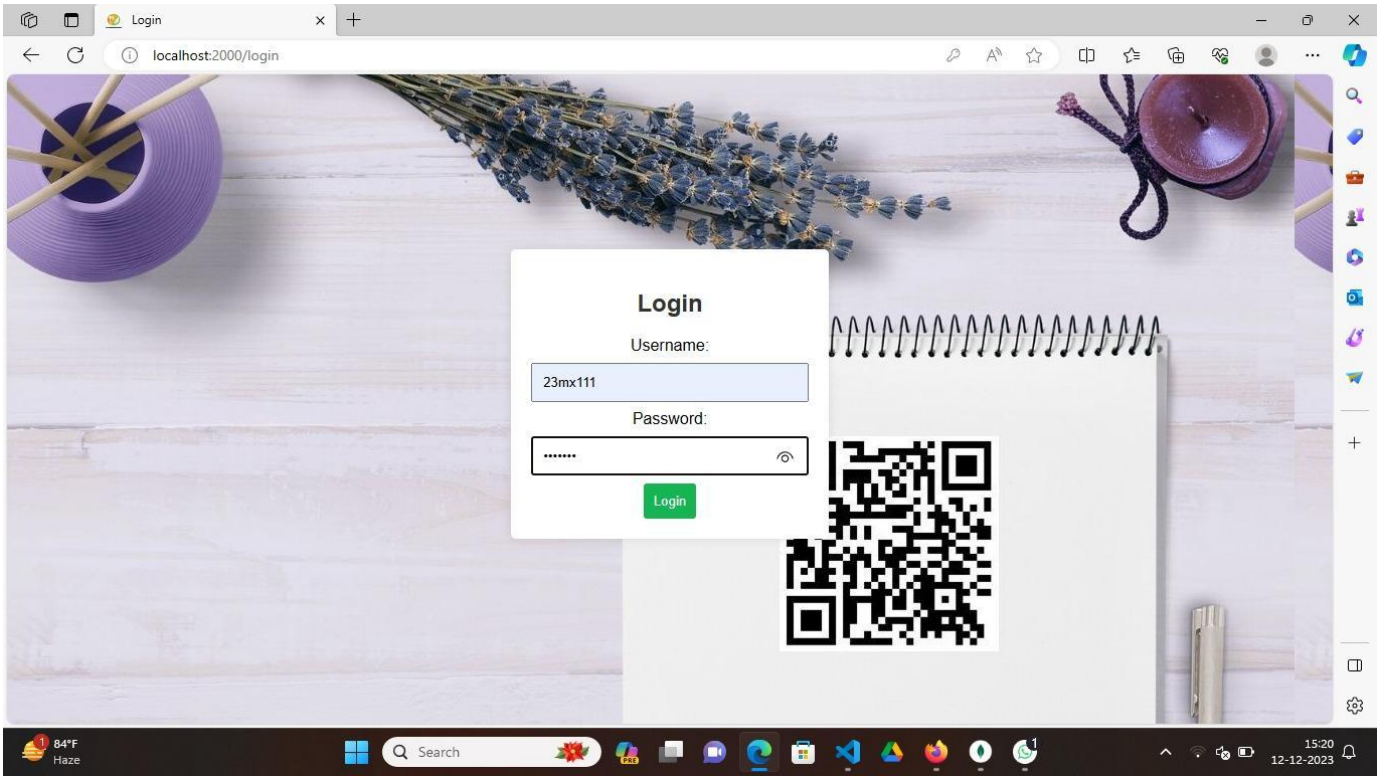


Fig. 4.3 Login Page

Implementation of Login page module:

```
const express = require("express");
const path = require("path");
const bcrypt = require("bcrypt");
const mongoose = require("mongoose");
const bodyParser = require('body-parser');
const uuid = require('uuid');
const app = express();
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.set("view engine", "ejs");
app.use(express.static(__dirname + '/public'));
app.set("views", path.join(__dirname, "views"));
mongoose.connect("mongodb://127.0.0.1:27017/scanner", {
useNewUrlParser: true,
useUnifiedTopology: true,});
```

```
const db = mongoose.connection;
db.on("error", (err) => {
console.error("MongoDB connection error:", err);});
db.once("open", () => {
console.log("Connected to MongoDB");});
const userSchema = new mongoose.Schema({
name: String,
password: String,});
const User = mongoose.model("User", userSchema);
app.get("/", (req, res) => {
res.redirect("/home"); // Redirect to the home page by default});
app.get("/login", (req, res) => {
res.render("login", { errorMessage: null });});
app.get('/views/payment', (req, res) => {
res.render('payment');});
app.get("/customerlink", (req, res) => {
res.render("customerlink");});
app.get("/qr", (req, res) => {
res.render("qr");});
app.get("/bankingDetails", (req, res) => {
res.render("bankingDetails");});
app.post("/login", async (req, res) => {
try {
const username = req.body.username;
const user = await User.findOne({ name: username.toLowerCase() });
if (!user) {
console.log("Username not found:", username);
return res.render("login", { errorMessage: "Username not found" });}
const inputPassword = req.body.password;
const storedPassword = user.password;
console.log('Input Password:', `'${inputPassword}' (length:
${inputPassword.length})`);
console.log('Stored Password:', `'${storedPassword}' (length:
${storedPassword.length})`);
if(inputPassword==storedPassword){
```

```
res.redirect("/customerlink"); }
else {
console.log("Wrong password for username:", username);
res.render("login", { errorMessage: "Wrong password" });}} catch
(error) {
console.error("Error during login:", error);
res.render("error", { errorMessage: "An error occurred. Please try
again." });}
});
const port = 2000;
app.listen(port, () => {
console.log(`Server running on port: ${port}`);
});
```

4.4 Customer portal Module :

The Customer portal Module is responsible for managing account details of the customers to store their bank information to the collection and for generation of QR codes for their particular amount entered and then on scanning , the amount is debited from the bank automatically. The Fig. 4.4 shows the customer portal after successful login.

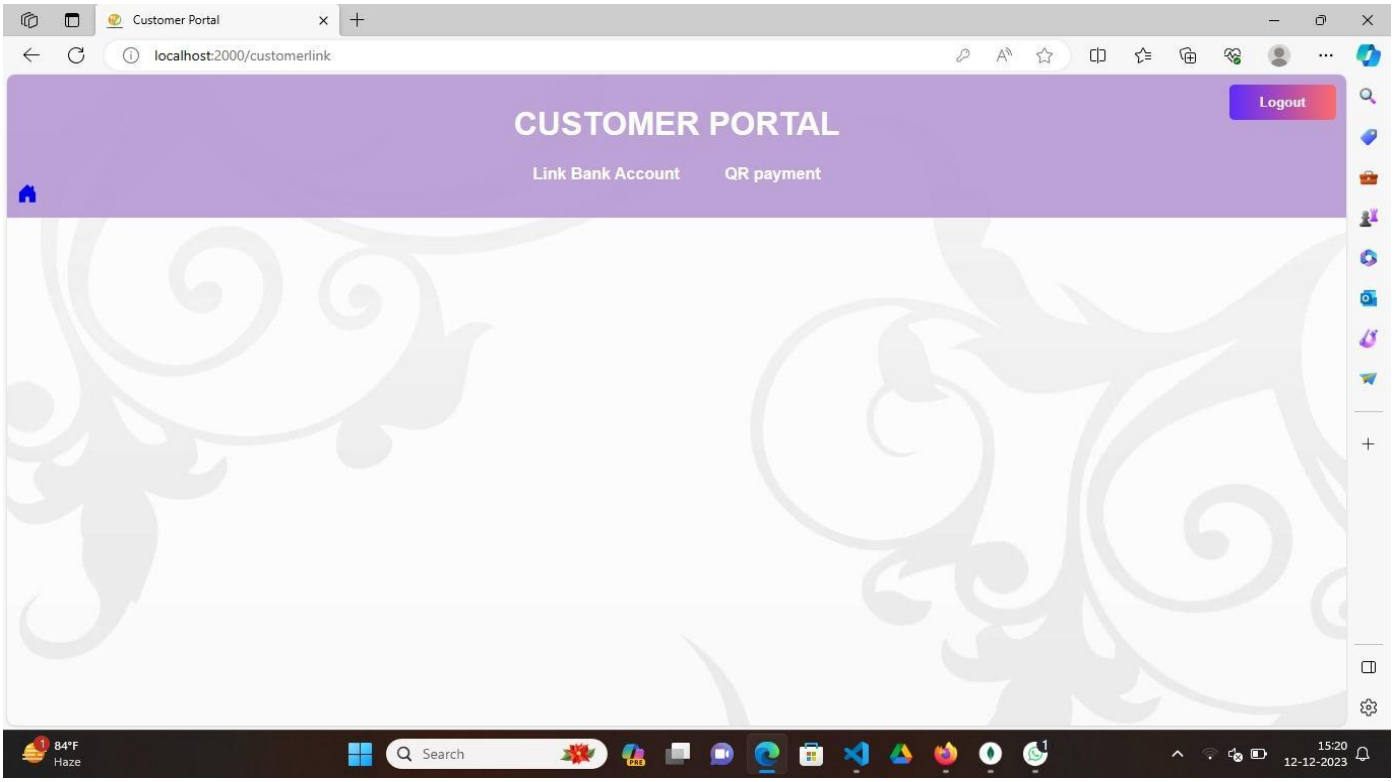


Fig. 4.4 Customer portal

Implementation of Customer portal page module :

```
<!DOCTYPE html>
<html>
<head>
<link rel="icon" type="image/png" href="/css/favicon.jpg">
<title>Customer Portal</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet" type="text/css" href="/bankingDetails.css">
</head>
<body>
<header>
<h1>CUSTOMER PORTAL</h1>
<nav>
```

```
<a class="nav-link" href="/bankingDetails">Link Bank Account</a>
<a class="nav-link" href="/qr">QR payment</a>
</nav>
<div >
<a href="home page.html">
<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20"
fill="currentColor" class="bi bi-house-door-fill" viewBox="0 0 16
16">
<path d="M6.5 14.5v-3.505c0-.245.25-.495.5-.495h2c.25
0.5.25.5.5v3.5a.5.5 0 0 0 .5.5h4a.5.5 0 0 0 .5-.5v-7a.5.5 0 0
0-.146-.354L13 5.793V2.5a.5.5 0 0 0-.5-.5h-1a.5.5 0 0
0-.5.5v1.293L8.354 1.146a.5.5 0 0 0-.708 0l-6 6A.5.5 0 0 0 1.5
7.5v7a.5.5 0 0 0 .5.5h4a.5.5 0 0 0 .5-.5Z"/>>
</svg></a></div>
<a href="home page.html">
<div class="logout-container">
<button class="logout-button" onclick="logout()" ><a class="nav-link"
 href="/views/home">Logout</button></div></a></a>
</div>
</header>
<script>
function logout() {
alert("Logout button clicked!");}
</script>
</body>
</html>
```

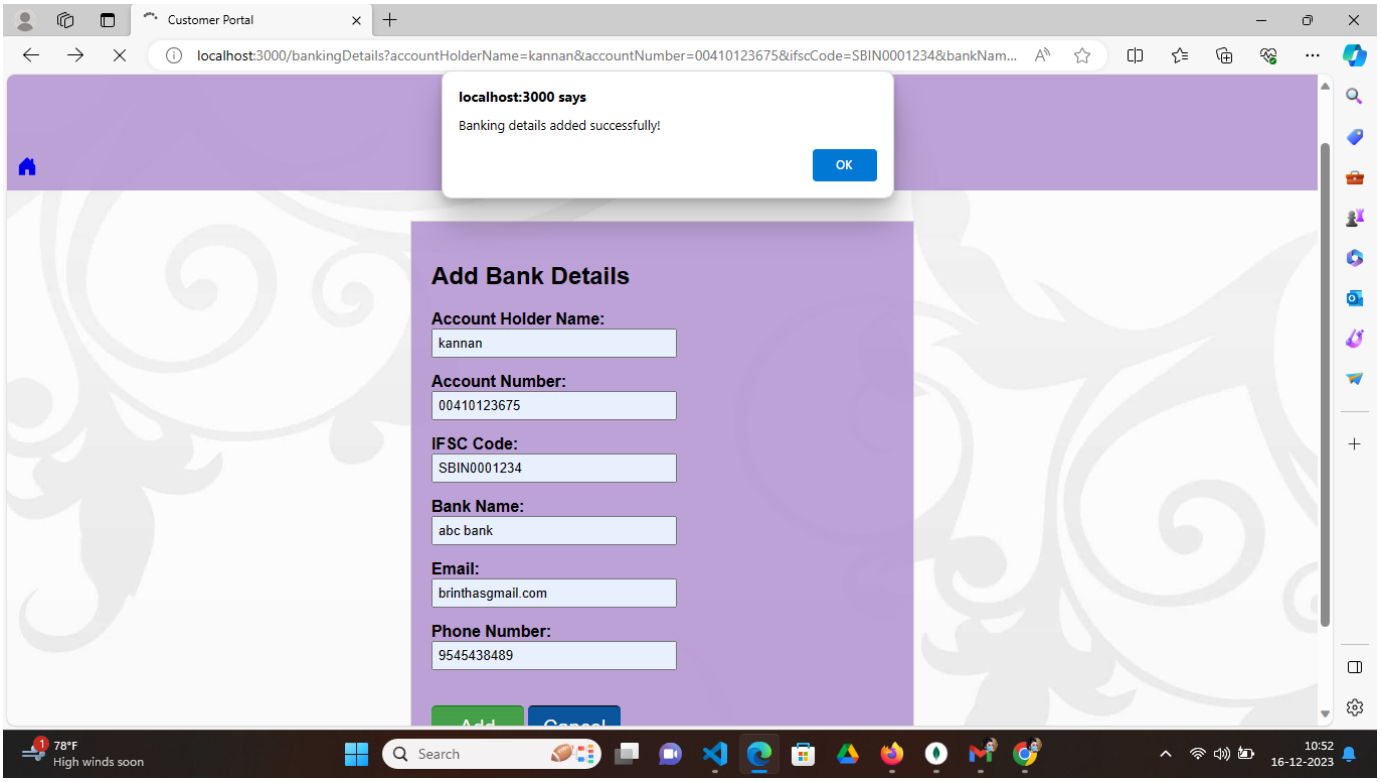


Fig. 4.5 Adding bank details

Implementation of Bank details page:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const BankingDetails = require('./models/bankingDetailsModel');
const app = express();
const PORT = process.env.PORT || 3000;
mongoose.connect('mongodb://127.0.0.1:27017/bookingApp', {
useNewUrlParser: true,
useUnifiedTopology: true,});
mongoose.connection.on('error', (err) => {
console.error('MongoDB connection error:', err);});
mongoose.connection.once('open', () => {
console.log('Connected to MongoDB');});
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
```

```
app.set('view engine', 'ejs');
app.set('views', __dirname + '/views');
app.use(express.static('public'));
app.get('/', (req, res) => {
res.render('bankingDetails'); // Change 'booking' to 'bankingDetails'
});
app.post('/api/save-banking-details', async (req, res) => {
const {
accountHolderName, accountNumber, ifscCode, bankName, email,
phoneNumber,} = req.body;
console.log('Received form data:', req.body);
try {
if (
!accountHolderName || !accountNumber || !ifscCode || !bankName ||
!email|| !phoneNumber) {
throw new Error('All required fields must be provided.');}
const newBankingDetails = {
accountHolderName, accountNumber, ifscCode, bankName, email,
phoneNumber,};
const savedBankingDetails = await
BankingDetails.create(newBankingDetails);
console.log('Banking details saved:', savedBankingDetails);
res.status(200).json({ message: 'Banking details added
successfully!'});}
catch (err) {
console.error('Error adding banking details:', err);
res.status(500).json({ message: 'Error adding banking details.' });
}
});
app.listen(PORT, () => {
console.log(`Server is running on http://localhost:${PORT}`);
});
```

4.5 Payment Module :

The Payment module supports the scanning of QR codes at the canteen's point of sale (POS) devices.The scanned QR code should identify the user and deduct the appropriate amount from their account. Payments should be processed in real-time to ensure immediate and accurate deduction from the user's account.The Fig. 4.6 shows the generated QR for the entered amount and Fig. 4.7 shows the scanning of the generated QR. Fig 4.8 shows the transaction was successful.

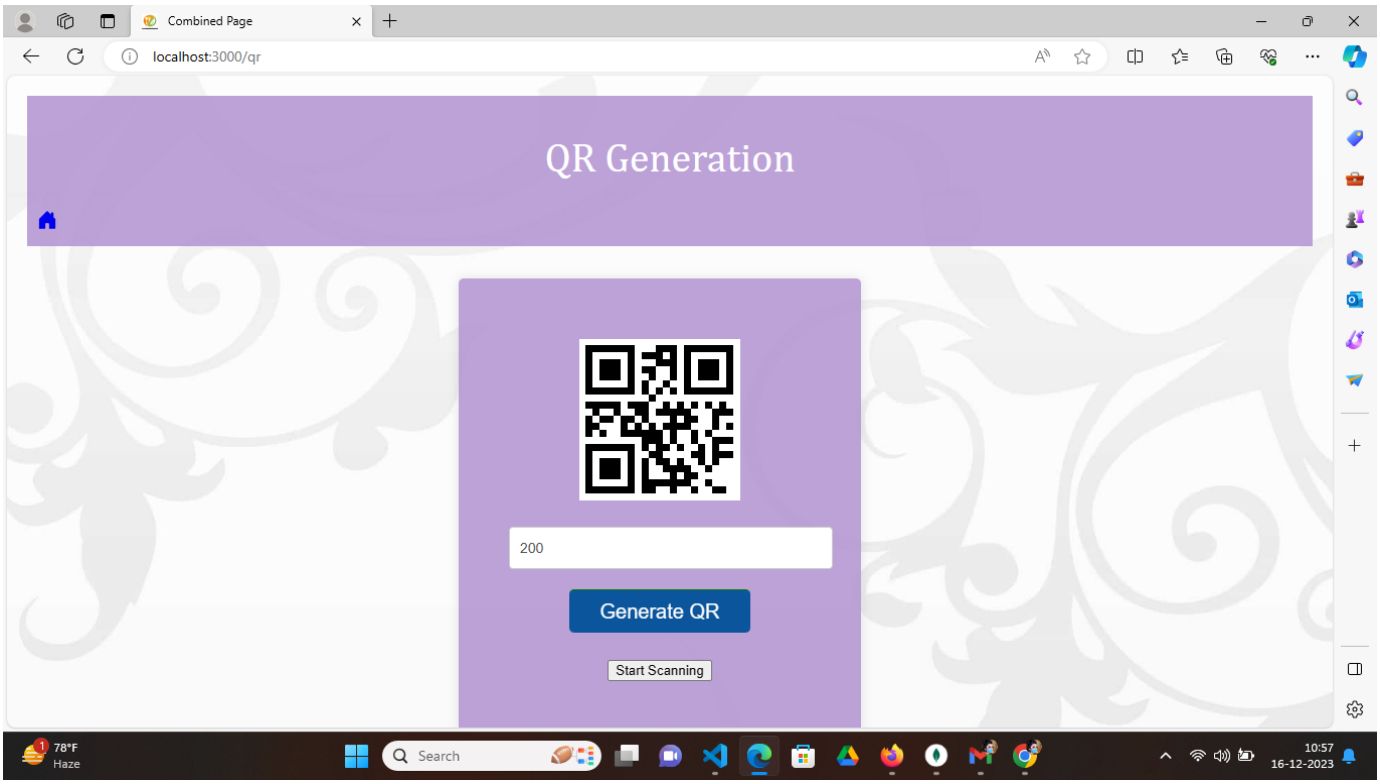


Fig. 4.6 QR generator

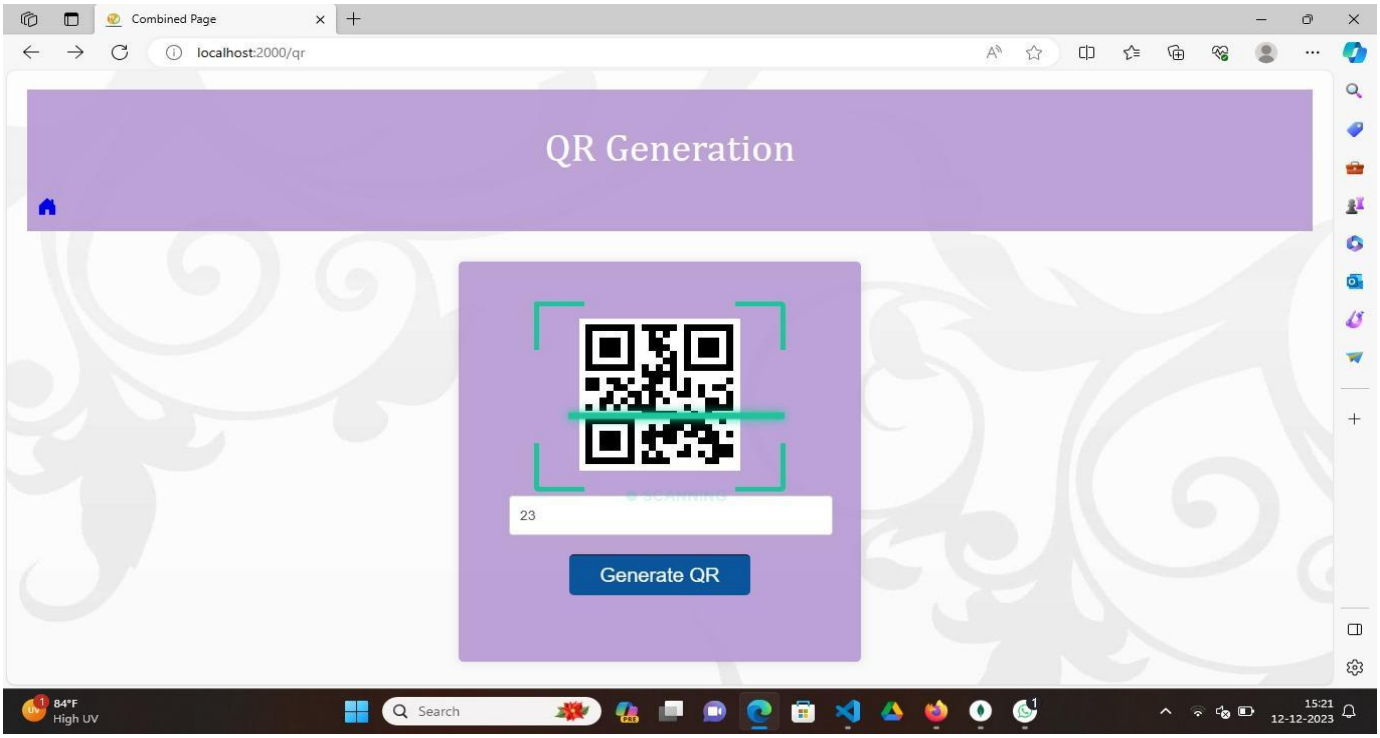


Fig. 4.7 QR Scanning

Implementation of QR generation page module:

```
<script
src="https://cdn.jsdelivr.net/gh/jquery/jquery/dist/jquery.min.js">
</script>
<script
src="https://cdn.jsdelivr.net/gh/neocotic/qrious/dist/qrious.min.js">
</script>
<script
src="https://cdn.rawgit.com/serratus/quaggaJS/6.4.0/dist/quagga.min.j
"> </script>
<script
src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.min.js">
</script>
<script>
jQuery(document).ready(function () {
function htmlEncode(value) {
```

```
return jQuery('<div/>').text(value).html();}
  jQuery('#generate').click(function (event) {
  event.preventDefault();
  const amount = jQuery('#content').val();
  console.log("Amount:", amount);
  const plainTextValue = amount;
  const qr = new QRious({
  element: document.getElementById('qrCode'),
  value: plainTextValue,
  size: 160,});
  });
  jQuery('#startButton').click(function (event) {
  event.preventDefault();
  const amount = jQuery('#content').val();
  console.log("Amount for scanning:", amount);
  jQuery(this).hide();
  jQuery('#scannerContainer').show();
  jQuery('.ocrloader').addClass('animate');
  window.location.href = '/views/payment'; });
  });
  </script>
```

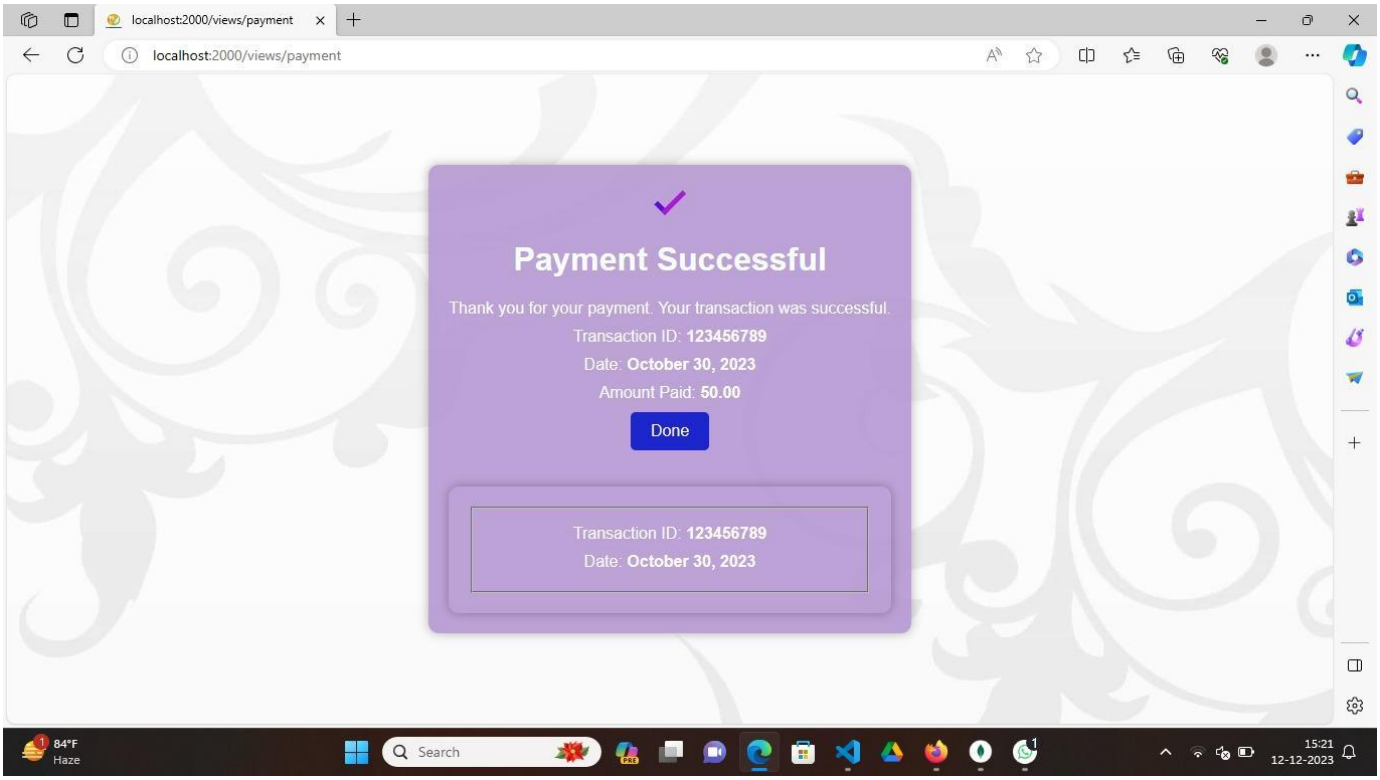


Fig. 4.8 Successful transaction

Implementation of successful transaction page module:

```
const uuid = require('uuid');
mongoose.connect('mongodb://127.0.0.1:27017/banktrans', {
useNewUrlParser: true,
useUnifiedTopology: true,});
const db = mongoose.connection;
db.on('error', (err) => {
console.error('MongoDB connection error:', err);});
db.once('open', () => {
console.log('Connected to MongoDB');});
const paymentSchema = new mongoose.Schema({
amount: {
type: Number, required: true,},
user  id:Number,});
const Payment = mongoose.model('Payment', paymentSchema);
```

```
app.get('/views/payment', async (req, res) => {
try {
const latestPayment = await Payment.findOne().sort({_id: -1
}).exec();
if (!latestPayment) {
return res.status(404).json({ message: 'No payment found' });}
const currentDate = new Date().toLocaleDateString();
const transactionId = latestPayment._id;
const amountPaid = latestPayment.amount;
res.render('payment', { currentDate, transactionId, amountPaid });}
catch (error) {
console.error('Error fetching payment details from MongoDB:', error);
res.status(500).json({ message: 'Error fetching payment details from
MongoDB.' });}
});
app.post('/views/payment', async (req, res) => {
try {
const amount = req.body.amount;
if (!amount) {
return res.status(400).json({ message: 'Amount is required' });}
const newPayment = new Payment({ amount });
await newPayment.save(async (error, savedPayment) => {
if (error) {
console.error('Error saving payment to MongoDB:', error);
return res.status(500).json({ message: 'Error saving payment to
MongoDB.' });}
const transactionId = savedPayment._id;
const currentDate = new Date().toLocaleDateString();
const amountPaid = savedPayment.amount;
res.render('payment', { currentDate, transactionId, amountPaid
});});};}            catch(error) {
console.error('Error generating transaction details or saving payment
to MongoDB:', error);
res.status(500).json({ message: 'Error generating transaction details
or saving payment to MongoDB.' });}});
```

CHAPTER 5
SYSTEM TESTING

System testing is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because web-based systems and applications reside on a network and interoperate with many different operating systems, browsers, hardware platforms, and communication protocols, the search for errors represents a significant challenge for web applications.

The distributed nature of client-server environments, the performance issues associated with transaction processing, the potential presence of a number of different hardware platform, the complexities of network communications, the need to serve multiple clients from a centralized database and the requirements imposed on the server all combine to make testing of client-server architectures.

5.1 Testing Methodologies

The testing process intricately examines the software's logical internals, affirming the completeness of code execution and scrutinizing functional aspects. It guarantees that specified inputs yield results consistent with the anticipated outcomes. Testing is integral to the development cycle, its extent contingent upon the application's size and complexity. This chapter elucidates the diverse testing strategies embraced in this project.

Unit Testing:

Unit testing involves evaluating individual units or components of the software in isolation to ensure they function as expected. For the Cashless Canteen Automation System, unit testing would focus on testing individual modules such as QR code generation, payment processing algorithms, and account management functionalities. The goal is to identify and fix any defects within these isolated components before they are integrated into the larger system. The Table 5.1 shows the sample test cases for unit testing.

Module	Test Case	Input Data	Expected Outcome	Result
Login	Valid Input	Customer ID: 23MX111	Valid QR code generated successfully	Pass
	Invalid Input	Customer ID: ABC	Error message: Invalid input	Pass
QR generation	Valid Input	Enter amount:30	Valid	Pass
Payment processing	Sufficient Balance	Amount:30	Payment processed successfully	Pass
	Insufficient Balance	Amount:100	Error message: “Insufficient balance”	Pass

Table 5.1 Sample test cases for unit testing

Integration Testing:

Integration testing assesses the interactions and interfaces between different modules or components of the system to ensure they work together seamlessly. In the context of the canteen system, integration testing would involve verifying that QR code generation integrates correctly with payment processing, and that account management functions interact effectively with the overall system. This testing phase aims to detect and rectify issues that may arise when combining individual components, ensuring the smooth operation of the integrated system.

Functional Testing :

Functional testing focuses on ensuring that the system's functionalities work as intended. It involves evaluating individual features, use cases, and user interactions to confirm that the software performs the specified functions accurately. In the context of the Cashless Canteen Automation System, functional testing would include verifying that QR code generation, payment processing, and account management functionalities operate correctly.

Performance Testing:

Performance testing assesses the system's responsiveness, scalability, and stability under different conditions. This includes load testing to determine how the system handles concurrent users, response time testing to evaluate speed, and stress testing to assess system behavior under extreme conditions. In the canteen system, performance testing ensures that QR code transactions are processed efficiently, even during peak hours.

Security Testing:

Security testing is essential to identify vulnerabilities and ensure the confidentiality, integrity, and availability of the system. This includes testing authentication mechanisms, data encryption, and authorization processes. For the Cashless Canteen Automation System, security testing confirms that QR code data is securely encrypted, and only authorized users can access and perform transactions.

User Interface (UI) Testing:

User interface testing evaluates the clarity, consistency, and ease of use of the system's graphical elements. It ensures that the UI is intuitive and accessible to the target users. In the canteen system,UI testing would involve validating the design of web applications, the simplicity of QR code generation, and the overall user experience for both students and staff. UI testing in the Cashless Canteen Automation System aims to provide a seamless and user-friendly experience.

5.2 Test Case Report

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

Test Case: User Authentication

Case 1:

Description: Verify that users can authenticate using their User id.

Expected Outcome: Users should be able to log in successfully.

Case 2:

Description: Verify that incorrect login credentials result in an error message.

Expected Outcome: An error message should be displayed for incorrect login attempts.

Test Case: Bank details adding

Case 1:

Description: Verify that the bank details are stored.

Expected Outcome:The bank details collection should be successfully created and stored in the database.

Case 2:

Description: Verify that mandatory details fields are validated.

Expected Outcome: Users should be prompted if required fields are not filled.

Test Case: QR generation

Case 1:

Description: Verify that users can generate the qr code by entering the amount.

Expected Outcome: The QR generation page should accept the entered amount and generate qr.

Case 2:

Description: Verify that the qr is working properly and should be scanned by Pos.

Expected Outcome: QR should be scanned and detect amount.

Test Case: Transaction complete

Case 1:

Description: Verify that the transaction details are displayed after completion.

Expected Outcome: Users should see transaction status.

CHAPTER 6
CONCLUSION

The Cashless Canteen Automation System represents a transformative leap forward in the realm of cafeteria management within educational institutions. The system, driven by the efficiency of QR code technology, is poised to address the persistent challenges faced by our institution's canteen, notably the long queues and consequential delays during break times.

The primary objective of this innovative system is to modernize and streamline the existing manual processes, which have proven to be a bottleneck, causing inconvenience for both students and faculty. The detrimental impact on punctuality for lectures is a pressing concern that the proposed solution seeks to mitigate.

By incorporating personalized QR codes and user-friendly web applications, the system not only offers a technologically advanced alternative but also promises to revolutionize payment processes. The envisaged reduction in waiting times and the seamless transition from traditional cash transactions to QR code payments mark a significant enhancement in canteen efficiency and overall user convenience.

The collaborative enthusiasm of both teachers and students to find a solution to the prolonged waiting times underscores the significance and urgency of implementing such a cashless system. In essence, the Cashless Canteen Automation System is poised to contribute to a more streamlined, efficient, and user-centric cafeteria experience, aligning with the evolving expectations and preferences in today's digital age.

BIBLIOGRAPHY

- **BOOK REFERENCES:**

1. Sasha Vodnik, " HTML5 and CSS3 Complete", Cengage Learning, 2015.

2. AchyutGodbole, AtulKahate, "Web Technologies", Tata McGraw Hill, 2013.

3. Thomas Powell , Fritz Schneider, "JavaScript 2.0: The Complete Reference", Tata McGraw Hill, 2016.

4. Greg Lim , " Beginning Node.JS , Express and MongoDB development, Greg Lim , 2020

- **WEBSITE REFERENCES:**

1.https://www.geeksforgeeks.org/web-development/

2.https://developer.mozilla.org/en-US/docs/Learn

3.https://www.simplilearn.com/tutorials/nodejs

4.https://www.redhat.com/en/topics/api/

5.https://www.qr-code-generator.com/

6.https://superuser.com/questions/1055832/how-to-link-a-bank-account-to-my-website-for-payment

7.https://dribbble.com/tags/add-bank-account

8.https://www.tutorialspoint.com/mongodb/index.htm

9.https://www.w3schools.com/mongodb/