FourGuys proposal to further improve the point-of-sale system: Checkout

Sven Veit
dept. of Economics
Kiel University of Applied Sciences
Kiel, Germany
svenveit@hanyang.ac.kr

Jiyong Lee
dept. of Educational Technology
Hanyang University
Seoul, Korea
jdragon2000@naver.com

Zabolotniaia Anna
dept.Business Administration
Hanyang University)
Seoul, South Korea
elijaltayer@gmail.com

Aneesa Shaki
dept. of Engineering and Computer Science
University of New South Wales
Sydney, Australia
a.shaki@student.unsw.edu.au

***Abstract*—This document will define our proposal requirements, and explain our motivations and ongoing roles throughout the initial development process of the project and fulfil client requirements with an application to improve the POS system. This project aims to develop an enhanced POS (Point of Sale) system that leverages smartphone-based barcode scanning technology to enable customers to complete their purchases without the need to wait in checkout lines. By using our mobile application, users can simply scan product barcodes as they shop, automatically adding items to a virtual shopping cart. When ready, they can complete the payment through the app's secure and user-friendly interface, eliminating the need for traditional cashier interactions. This system not only reduces customer wait times but also minimizes staff workload and enhances the shopping experience by providing a seamless and efficient checkout process.**

***Index Terms*—POS system, barcode scanning, mobile payment, contactless checkout**

I. Introduction

A. Motivation

Amid the ongoing cost of living crisis and the rapid shift in consumer behavior, large retailers are increasingly focused on improving operational efficiency and cutting costs to meet rising customer expectations. Recent disruptions, including global supply chain issues and changing labor dynamics, have amplified the urgency for more innovative, cost-effective solutions in the retail industry. While technological advancements have already revolutionized the checkout process, most notably with the introduction of contactless kiosks as the benchmark for modern Point of Sale (POS) systems, there is still untapped potential for further improvement.

By capitalizing on the most universally accessible tool, the smartphone, we have the opportunity to not only reduce costs but also redefine the customer experience. The current retail landscape demands flexibility, and utilizing smartphones as mobile POS systems is a natural progression. This approach can significantly reduce infrastructure costs, eliminate the need for fixed kiosks, and decrease reliance on labor-intensive processes. Existing solutions allow QR and barcode scanning, but this project goes beyond that, aiming to fully integrate smart devices into the shopping journey. By empowering consumers to scan and purchase items independently, we can deliver a frictionless, personalized retail experience that meets the evolving demands of today's shoppers, while future-proofing retail operations against future disruptions

B. Problem Statement

The current issues within the economy due to the current inflation globally has caused for a need to reduce costs within the retail industry to place less burden on the consumer with a lack of leeway within produce and goods adaptations to the current POS systems and how to increase the efficiency and costs for the business whilst simplifying the process for the consumer are concerns of leading industry giants.

The main issues are as follows:

1) Long wait times and congestion: After completing their shopping, customers often face long queues at the checkout, especially during peak hours. This not only causes discomfort for customers, but also forces stores to operate more checkout counters or hire additional staff to maintain efficiency. These measures lead to increased costs and reduced customer satisfaction.

2) Increased costs: Installing and maintaining traditional POS systems and kiosks incurs significant costs, which rise exponentially as the size of the store increases. Additionally, these systems require regular updates and maintenance, further driving up operational expenses.

3) Lack of flexibility: The current payment system relies on fixed checkout counters or kiosks, which makes it difficult to adapt to customer traffic flow or shopping patterns. This limitation hampers the store's ability to manage in-store congestion or offer more convenient options to customers.

4) Limited customer experience: Current payment systems do not provide customers with enough control over their shopping experience, such as real-time price checking or managing their shopping list. This limits customers' ability to self-manage information related to their purchases.

These issues lead to customer dissatisfaction and increase operational costs. Our project aims to address these problems by introducing a mobile-based payment system that increases efficiency, reduces costs and improves the customer experience by allowing a streamlined seamless method of payment.

C. Solution

Our proposal is to solve this current problem through a mobile application that has the capabilities of functioning as a POS system within physical retails stores.

With this solution, customers can avoid traditional checkout lines, significantly reducing congestion and wait times, especially during peak shopping hours. Since there is no need for physical POS kiosks or additional staff assistance, retailers can drastically lower their infrastructure and labour costs. Moreover, this system allows retailers to move beyond the limitations of fixed checkout counters, adapting more flexibly to fluctuating customer flows.

Using smartphones also empowers customers to have more control over their shopping experience. They can access real-time information on prices, promotions, and their total cart value, easily add or remove items, and apply discounts automatically, completing transactions without physical interaction. This enhances convenience and safety, especially in the post-pandemic era where contactless solutions are highly valued.

By integrating this mobile-based payment system, we aim to not only boost operational efficiency but also transform the overall shopping experience into a more flexible, cost-effective, and customer-centric retail environment.

D. Relate Software

1) Amazon Price Comparison and Product Scanner: Amazon's mobile app allows users to scan product barcodes and instantly access product information, reviews, and pricing within Amazon's marketplace. Available on both iOS and Android, this app uses barcode scanning to retrieve product details, enabling users to compare Amazon's prices with in-store prices. This functionality closely aligns with the goals of our project.

2) ShopSavvy Barcode Scanner: ShopSavvy is an app that allows users to scan barcodes to view pricing and availability across various online and local stores. Available on the App Store and Google Play Store, ShopSavvy's barcode scanning technology and its price comparison feature across multiple retailers align closely with the objectives of our app.

3) Google Lens: Google Lens can scan barcodes and provide detailed product information, including prices, from various online sources. Integrated within Google Search and Google Photos, Google Lens uses advanced image recognition and search capabilities to retrieve product information via barcodes, which mirrors the core functionality of our proposed app.

4) Barcode Scanner Algorithm: Algorithms like ZBar and pyzbar are widely used in Python for barcode detection and decoding in various apps and services. The pyzbar Python library, available at https://pypi.org/project/pyzbar/, offers an open-source solution for reading barcodes in images, making it a potential foundational component for implementing the barcode scanning functionality in our app.

These references illustrate existing solutions and technologies that provide similar barcode scanning and price comparison features. For our app, integrating similar barcode decoding libraries and developing a comparison feature across supermarkets would differentiate it by focusing on price comparisons specifically within the grocery retail sector

E. Role Assignments

TABLE I
ROLE ASSIGNMENTS

Role	Name	Task
Software Developer	Sven	Develop the fundamental back-end systems to allow fulfill the functional requirements.
Software Developer	Aneesa	Develop the fundamental backend systems to allow fulfill the functional requirements.
Front-end Developer	Jiyong	Develops and optimizes user interfaces (UI) for web and mobile applications, ensuring a seamless and responsive user experience (UX). Collaborates with designers to implement visually engaging and accessible components, focusing on usability and interactivity. Ensures cross-platform compatibility and addresses performance optimization for efficient rendering and responsiveness.
Graphics Designer	Anna	To construct a visual design for the user experience

II. REQUIREMENTS

A. Barcode Scanner Feature

1) Barcode Scanning via Camera: Users can conveniently scan product barcodes using their smartphone camera to retrieve detailed product information instantly. This feature supports fast and accurate product recognition, making it easy for users to view product descriptions, prices, and other relevant details without manually entering information. By scanning each item as they shop, users can quickly and seamlessly register products in their cart, enhancing the overall shopping efficiency.

2) QR Code Scanning Support: In addition to traditional barcodes, users can scan QR codes directly within the app to access special offers, discounts, or additional product information. This functionality allows for the automatic application of relevant coupons or promotional discounts, so users can effortlessly benefit from store promotions. The QR code scanning capability adds flexibility, enabling a smoother shopping experience where savings are automatically applied.

B. Shopping Cart Feature

1) Add Products: Each product scanned by the user is added to a virtual shopping cart in real time. The cart updates with the latest price and quantity, allowing users to keep track of what they've added and monitor their spending. With each scan, product details such as name, price, and available quantity are immediately visible, making it simple for users to manage their shopping list and streamline the purchasing process.

2) Modify and Remove Products: Users can easily manage items in their shopping cart by adjusting quantities or removing products as needed. Each modification automatically recalculates the total price, reflecting real-time changes and ensuring users have a clear and accurate view of their spending. This flexibility helps users control their shopping experience, making it easy to review and make changes before finalizing a purchase.

3) View total Payment Amount: The system calculates the total cost of all items, including any applicable taxes, discounts, and promotions, in real time. This feature provides a transparent view of the final amount due, helping users make informed purchasing decisions and preventing unexpected charges at checkout. The clear summary of the total payment amount gives users a quick overview of their cart balance at any time during the shopping process.

C. Payment Feature

1) Choose Payment Methods: Users can register multiple payment methods, such as credit cards, mobile payment options, or digital wallets, within the app. Each saved method can be securely managed, allowing users to select or change their preferred payment option quickly and efficiently at checkout. By offering flexible payment choices, the app supports a variety of user preferences, making the checkout process faster and more convenient.

2) Instant Payment: With a single click, users can complete their purchase securely and quickly. The instant payment feature leverages secure transaction protocols, ensuring each payment is processed safely. This option is designed for efficiency, reducing wait times and enabling a seamless checkout experience that helps users finalize their purchases with confidence.

3) View Payment History: After each transaction, users have access to detailed payment histories, including itemized receipts and comprehensive records of previous purchases. This feature allows users to track their spending, review past orders, and easily access receipts for future reference. Payment history is stored in the user's profile, ensuring that all past transactions are conveniently available for review.

D. Promotions and Coupon Management

1) Register Discount Codes/Coupons: Users can enter discount codes or coupons directly within the app, which are then stored in their account for future use. The system remembers these codes, automatically applying them to eligible purchases during checkout. This feature makes it easy for users to keep track of their savings and ensures that no discounts are overlooked when shopping.

2) Automatic Coupon Application: When a product is scanned, the system automatically checks for any applicable promotions or discounts. Relevant offers are applied in real time, with the discounted prices immediately visible in the shopping cart. This seamless integration of promotions enhances the user experience, as they can see their savings instantly without additional steps.

3) Reward Programs: Users can accumulate rewards or loyalty points from store purchases, which can be redeemed or applied to future transactions. The system tracks each user's reward status, making it easy for users to benefit from loyalty programs and store credits that encourage continued shopping. This feature adds value by promoting customer retention and offering users more ways to save.

E. Notifications and Push Features

1) Promotion Notifications: Users receive real-time push notifications about discounts, special offers, or promotions that relate to products they've scanned or shown interest in. These timely updates ensure that users never miss an opportunity to save and can stay informed of new deals without actively searching for them. The notifications are customizable, allowing users to control which alerts they receive.

2) Payment and Order Notifications: Once a transaction is completed, users are instantly notified of successful or failed payments, along with real-time updates on the status of their orders. This feature provides assurance and transparency, as users are kept informed of their transaction status and any changes to their order's processing or delivery timeline

F. User Profile Management

1) Manage Personal Information: Users can update and manage their personal information, such as name, contact details, and address, within the app. Keeping this information accurate ensures a seamless and personalized shopping experience. The app prioritizes data security, so users can trust that their information is protected while they make updates as needed.

2) Manage Payment Information: Users have full control over their registered payment options, allowing them to add new payment methods or edit existing details whenever necessary. This flexibility enables users to keep their payment preferences up-to-date, making checkout smooth and preventing any interruptions due to outdated information.

III. Development Environment

A. Development Platform

Web application platform is suitable for our software due to its accessibility, scalability and cross platform compatibility. Unlike traditional desktop applications, a web application can be accessed from any device with internet access, allowing users to interact with the software in real time without requiring specific hardware or operating system configurations. This flexibility significantly broadens the user base, as it eliminates the need for installations or updates on individual devices. Additionally, web applications are naturally scalable, making it easy to accommodate growing user demands and adapt to changing requirements with no requirement to accommodate to the OS system of a device reducing the complexity of the project. Development frameworks for web applications also offer extensive libraries and tools that accelerate development and provide built-in security features, which are crucial for protecting sensitive data. Altogether, these advantages make a web application an optimal platform for delivering robust, user-friendly, and secure software solutions.

B. Coding Langauges

1) Javascript: JavaScript is a coding language influenced by Java with syntax derived from C. It is enriched by a vast ecosystem of libraries and frameworks that enhance its capabilities in barcode scanning, data handling, and API integration. Libraries like QuaggaJS, jsBarcode, and Dynamsoft are specifically designed for barcode scanning, enabling web applications to process barcode data directly through the browser. For data handling, libraries such as Lodash and D3.js offer powerful tools for data manipulation, visualization, and transformation, making it easier to process large datasets and present them in interactive, user-friendly formats. JavaScript also offers comprehensive support for API integration, with libraries like Axios and Fetch simplifying asynchronous data requests and responses. For handling real-time data, libraries such as Socket.IO facilitate WebSocket connections, allowing developers to create applications that update dynamically without page reloads. Additionally, popular frameworks like React, Angular, and Vue streamline the development of user interfaces, providing reusable component architectures and state management tools, while libraries such as jQuery simplify DOM manipulation and event handling. These frameworks and libraries, along with JavaScript's cross-platform support, make it an essential language for building feature-rich, responsive, and data-driven web applications.Compatible software development frameworks include Node.js and React Native which offer an extensive library for building and developing software and a range of tools.

2) Typescript: Typescript is an extension of JavaScript that implements static typing, classes and interfaces. This allows for optimal code organisation and object-oriented programming techniques. Typescript allows for improved error detection with additional tooling support that enhances the IDE experience and ensures proper API documentation.

3) CSS: CSS frameworks and libraries complement JavaScript, offering pre-designed, responsive UI components that enhance the styling and layout of web applications. Libraries like Bootstrap, Tailwind CSS, and Bulma streamline styling by providing ready-made classes for layouts, typography, buttons, and more, allowing developers to quickly achieve consistent, visually appealing designs. Additionally, CSS animation libraries, like Animate.css and GreenSock (GSAP), enable smooth transitions and dynamic visual effects that bring interactivity to user interfaces.

4) HTML: HTML is the core language for structuring and presenting content on the web. It serves as the foundation of web pages, defining elements like headings, paragraphs, images, links, and forms, enabling browsers to interpret and display content in a coherent layout. The latest version, HTML5, introduced new semantic tags such as header, footer, article, and section, which help organize content more meaningfully while improving accessibility and search engine optimization (SEO). HTML5 also brought powerful features for embedding multimedia through audio and video elements, along with APIs that enable functionalities like local data storage and geolocation, enhancing interactivity and user experience in web applications. Working together with CSS for styling and JavaScript for interactivity, HTML is essential for building accessible, structured, and user-friendly websites and applications.

C. Cost Estimation

TABLE II
COST ESTIMATION TABLE

Expense attribute	**Cost per annum**	**reasoning**
Web Hosting and Server Costs	$1200-2400	Platforms such as AWS, Google Cloud or Heroku are required to host the website for use to deliver reliable, scalable and high-performance solutions.
Database Hosting	$240-600	Database hosting provides infrastructure needed to store, manage and access databases over the internet. Examples of database host include Firebase, MongoDB Atlas or MySQL.
Outsourced API	$600-1200	In the case of using a third party API to access barcode scanning libraries to reduce code development and focus on the websites functionalities.
Domain Name/registration	$0-20	The costs associated to secure a domain name.
SSL	$0-249	A digital authentication of the websites identity and enables encrypted connection for security purposes and needed to accept online payments.

D. Software in Use

1) VSCode: VSCode (Visual Studio Code) is our chosen code editor due to its flexibility and extensive support for various programming languages and extensions. It plays an important role in our project, which involves creating an application that allows users to scan barcodes with their phones for payment. VSCode allows us to customize our workspace and integrates well with Git, enabling us to efficiently track code changes. Additionally, it provides features like live sharing and remote development, enhancing collaboration and productivity among team members.

2) Git and Gitlab: Git serves as our core version control system, essential for tracking and managing code changes systematically. GitLab complements Git by providing a self-hosted platform with robust DevOps tools, allowing us to manage our barcode payment app project in one integrated environment. GitLab's CI/CD (Continuous Integration and Continuous Deployment) features are especially valuable for automating testing and deployment processes, which helps streamline our development pipeline. It also offers comprehensive project management tools, enabling us to oversee the entire lifecycle of the project in one place, from issue tracking to deployment. GitLab's integrated approach enhances our ability to handle tasks efficiently and keep the development process organized.

3) Github: While GitLab serves as our primary development platform, GitHub complements our workflow by providing a strong collaborative environment for code review and open-source visibility. GitHub's large community and social coding features, such as pull requests, make it ideal for reviewing and managing contributions from multiple team members. GitHub's user-friendly interface and extensive integrations also make it an excellent choice for code sharing and feedback. Unlike GitLab, which focuses heavily on integrated DevOps workflows, GitHub excels in fostering collaboration and is widely used by developers for code management. By using GitHub, we ensure code quality and organization, which aligns well with our project's focus on creating a robust barcode payment app.

IV. Specifications

A. Login page

A login feature is needed to store the user's payment methods and purchase history. When the user enters a correct username and password and presses the login button, a login POST API request is sent to the server.

B. Register page

The username must be 2 to 20 characters long, using only English letters and numbers. The password must be a combination of English letters and numbers, with a length between 8 and 15 characters, and it should meet additional conditions such as an option to include special characters. Once registration is complete, the user's username and password are stored in the database.

C. User information page

On the user information page, users can change their username or password and navigate to the card and account registration page or the purchase history page.

D. Card or account registration page

On the card or account registration page, users can link a card or account to enable instant payment when scanning a barcode. Users can register multiple cards or accounts and select one of them at the time of payment. When a user registers a card or account, the information is saved in the database. For future payments, the payment gateway API is integrated to send payment requests and receive approval or rejection responses. When storing card numbers or account information, encryption technology is always used to protect sensitive information, and security is reinforced for login sessions and API requests.

E. Purchase history page

On the purchase history page, users can view the items they have purchased so far, stored in the database. This information can be used to recommend items that may be of interest to the user.

F. Barcode scanning page

On the barcode scanning page, the application connects to the phone's camera, allowing users to take photos. Based on the photo of the barcode, it accesses the market database to retrieve the item's price, quantity, and type, which are then added to the shopping cart. When the user presses the checkout button in the cart, the payment is processed through a pre-registered card or account.

V. Architecture Design and Implementation

A. Overall architecture

Below is the system architecture design for our application, which will be deployed on Microsoft Azure Cloud. The architecture is modular, with clear relationships and functionality between components. The design incorporates frontend, backend, database, external services, and Azure cloud features.

1) Frontend: Description: This module allows users to interact with the application via a web-based interface. Required Classes: - - - ‘BarcodeScanner‘ (for scanning barcodes) ‘PaymentGatewayUI‘ (for processing payments) ‘ReceiptViewer‘ (for displaying receipts) Technologies: HTML, CSS, JavaScript, Node.js, SQL

2) API Gateway : Description: Acts as the bridge between the frontend and backend, handling all API requests and responses. Required Classes: - - ‘RequestHandler‘ (handles API calls from the frontend) ‘Authenticator‘ (validates user requests) Technologies: Node.js with Express.js

3) Backend: Description: Processes all requests, retrieves data, handles business logic, and communicates with external APIs and the database. Required Classes: - - - ‘ProductManager‘ (fetches product information from the database) ‘PaymentProcessor‘ (handles payment transactions) ‘ReceiptHandler‘ (stores and retrieves receipt data) Technologies: Node.js, Express.js

4) Database: Description: Stores product data, user data, receipts, and store details. Required Classes: - - - ‘ProductsTable‘ (product details like barcode, price, and name) ‘ReceiptsTable‘ (stores completed transactions) ‘StoresTable‘ (location-based store details) Technologies: Azure SQL Database

5) Azure Cloud Services: Description: Hosts the application and provides additional cloud services for scalability and reliability. Components: - - - - Azure App Service: Hosts the frontend and backend. Azure SQL Database: Stores all structured data. Azure Blob Storage: For storing assets like receipts, images or JSON files. Azure Key Vault: Manages sensitive information like API keys securely.

6) Third-Party Services: Description: Integrates external APIs and libraries for specific functionalities such as payment processing. Required Components: - - JavaScript Barcode Library: Used directly within the frontend for barcode scanning. No external API is required. Payment Gateway API: For processing payments (e.g., Stripe, PayPal).

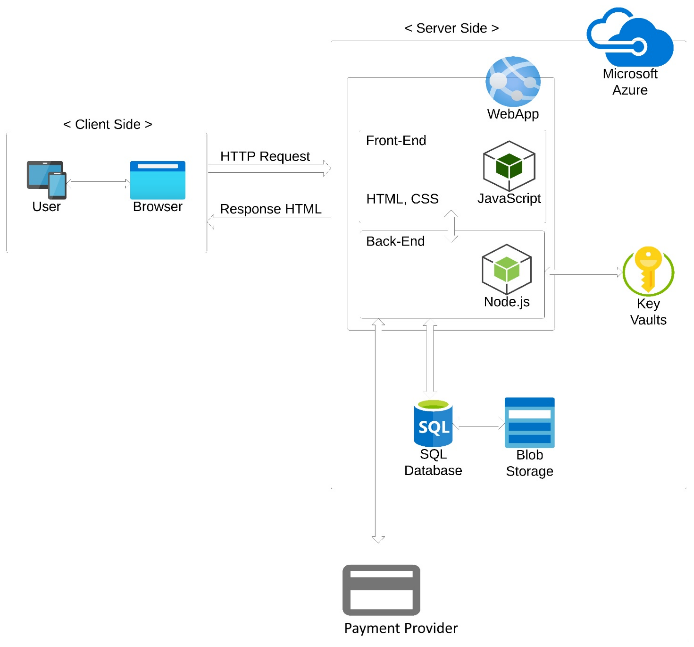

Fig. 1. Overall Architecture

B. Directory organization

TABLE III
DIRECTORY ORGANIZATION

Directory	File names	Description
Undefined	README.md	Overview and instructions for the project.
Undefined/Software Engineering	Assignment – Project [2] (Suggestion).pdf	Project suggestions.
Undefined/Software Engineering	Assignment – Project [2].docx	Detailed assignment document.
Undefined/Software Engineering/Project	app.js	Main JavaScript file for application functionality.
Undefined/Software Engineering/Project	database.sql	SQL file for setting up and managing the database.
Undefined/Software Engineering/Project	main.html	Frontend HTML file.
Undefined/Software Engineering/Project	package-lock.json	Project dependencies and metadata.
Undefined/Software Engineering/Project	package.json	Exact dependency versions for reproducibility.
Undefined/Software Engineering/Project	server.js	API server for the application.
Undefined/Software Engineering/Project/ode-modules	.package-lock.json	Directory for all installed Node.js dependencies (autogenerated).
Undefined/backend/ Authentication	auth.ts	Handles authentication logic.
Undefined/backend/Data	DataStore.json	A JSON file containing application data.
Undefined/backend/Data	dataStore.ts	TypeScript file for managing data storage or database interaction.
Undefined/backend/Other	helper Functions.ts	Provides utility functions used across the project.
Undefined/backend/Server	config.json	Configuration file for server settings.
Undefined/backend/Server	requests.ts	Handles HTTP requests and API endpoints.
Undefined/backend/Server	server.ts	Backend server logic (TypeScript version).
Undefined/backend/tests	adminAuth Login.test.ts	Tests for admin login functionality.
Undefined/backend/tests	adminAuth Logout.test.ts	Tests for admin logout functionality.
Undefined/backend/tests	adminAuth PasswordUpdate.test.ts	Tests for admin password update functionality.
Undefined/backend/tests	adminAuth Register.test.ts	Tests for admin registration functionality.
Undefined/backend/tests	adminAuth UpdateUserDetails.test.ts	Tests for updating user details by admin.
Undefined/backend/tests	adminAuth UserDetails.test.ts	Tests for retrieving admin user details.

C. Module 1: User Authentication

The User Authentication module is responsible for managing the secure login, registration, and password recovery processes. It ensures that users can access the system securely, while protecting sensitive information like personal details and payment methods. This module also supports role-based access, differentiating between customers, administrators, and other user types. This module facilitates user registration with email addresses and passwords, social media logins, or phone numbers. It verifies user credentials during login and encrypts sensitive data like passwords to ensure security. Additionally, it includes password recovery functionality, allowing users to reset forgotten passwords via email verification. The source code is located in /authentication directory of the project and consists of the file auth.ts utilising open-source libraries available in Typescript and JavaScript. The module is critical for ensuring secure access to the application, protecting user information, and enabling a personalized shopping experience.

D. Module 2: Barcode recognition

The Barcode Recognition module enables the system to scan and identify products in the store by decoding their barcodes. It acts as a bridge between the physical items in the store and their digital representations in the system. This module uses device cameras or dedicated scanners to capture barcodes, which are then decoded into unique product identifiers, such as UPC or SKU codes. Once decoded, the system retrieves the corresponding product details from the database, including the product name, price, and stock status.

E. Module 3: Product Management

The Product Management module serves as the backbone for maintaining all product-related information in the system. It ensures that the product database is accurate and up-to-date, facilitating smooth integration with other modules like inventory and checkout. This module allows administrators to add, edit, or delete products, including details such as product name, category, price, and stock quantity. It also integrates with the Inventory Management module to automatically update stock levels.

F. Module 4: Product Purchase

The Product Purchase module manages the addition of scanned items to a virtual shopping cart and prepares them for checkout. It enables customers to review and modify their selections before completing a purchase. This module maintains a virtual cart where scanned products are stored. It displays the product details, quantity, and total cost, allowing users to modify quantities, remove items, or apply discounts. It also calculates the total price, including taxes.

G. Module 5: Payment Gateway Integration

The Payment Gateway Integration module allows users to securely make payments for their purchases. This module handles the connection between the website or kiosk app and third-party payment processors like Stripe, PayPal, or Square. It ensures that user transactions are processed securely and efficiently, managing everything from payment authorization to finalizing the purchase. This module facilitates secure payment transactions by integrating with external payment gateways. It manages the entire payment flow, starting with gathering payment information, verifying the payment details with the payment provider, processing the transaction, and sending a confirmation once the payment is successful. Additionally, it handles refunds, payment errors, and provides users with receipts. It also supports multiple payment methods, such as credit and debit cards, digital wallets, or bank transfers.

H. Module 6: Inventory Management

The Inventory Management module is responsible for tracking and updating the availability of products in real time. It ensures that users can only purchase products that are available in stock and helps store administrators monitor and maintain appropriate inventory levels. The module also tracks sales and stock replenishment, helping prevent stockouts or overstocking.

This module constantly updates product stock levels based on purchases, returns, or restocks. When an item is scanned during checkout, the system verifies that there is enough inventory to fulfill the purchase. The module also generates notifications for low stock and automatically updates the online inventory. In some cases, it integrates with external systems for restocking or reporting purposes. It ensures that the inventory remains synchronized between the physical store and the online database.

I. Module 7: Search and Filtering

The Search and Filtering module allows users to quickly locate products based on specific search criteria, such as product name, category, or price range. This module enhances the shopping experience by making it easier for customers to find items in the store, especially in a self-checkout setting where they may want to search for products before scanning them.

This module provides functionality for both keyword-based searches and advanced filtering options. Users can search for products by name, description, or category, and further refine their search using filters such as price range, product type, brand, or other attributes. The search results are then presented in an organized manner, often with pagination, to ensure a smooth user experience. Additionally, the module can integrate with a product recommendation engine to display relevant items based on the search query.

J. Module 8: Recommendation Engine

The Recommendation Engine module analyzes user behavior and preferences to provide personalized product recommendations. In a self-checkout environment, this module helps suggest related or frequently purchased products to enhance the shopping experience and increase sales.

This module uses algorithms to analyze customer data, such as past purchases, browsing history, and demographic information. Based on this data, the system generates personalized recommendations, displaying them on the user interface. The recommendations may include related products, promotional items, or other goods that the customer might be interested in, which can be added directly to the shopping cart. It also uses machine learning models to improve the accuracy of its suggestions over time.

K. Module 9: Geolocation Services

The Geolocation Services module allows the system to track the location of the user or kiosk in real time. This is especially useful in a grocery store setting where location-based features like store-specific promotions or proximity-based product recommendations can improve the customer experience.

This module uses the user's device location (via GPS or IP-based geolocation) to provide context-aware services. For example, it can track the user's position within the store and offer location-based promotions, such as discounts for items near the user's current location. It also allows the system to direct customers to specific aisles or product categories within the store based on where they are located. Additionally, it can assist in creating a more efficient self-checkout experience by providing real-time updates on nearby available products.

VI. Use Cases

A. Sign Up

The first page that users see is the sign-up or login page. On the sign-up page, users can enter their information such as username, password, name, email, and address to register. Once they fill out the necessary details and complete the sign-up process, their information is saved in the database for future logins.

B. Login

Users can log in using the username and password they created during the sign-up process. When a user attempts to log in, the system verifies if the information matches the data in the database. If the credentials are incorrect, a login failure message is shown. If they are correct, the user is logged in successfully and redirected to the main page of Checkout.

C. Main Page

Once logged into Checkout, the first page users will see is the main page. At the top of the Checkout main page, users can find their profile picture, name, and a welcome message. If it's the user's first time logging in, clicking "Start Checking Out" will show a tutorial for new users, explaining how to use Checkout and how to register a card. The main page also displays recommended products, nearby markets where users can shop, and a list of recently purchased items. Users can click on a recommended product or a past purchase to see more details, including prices and purchase history. When users click on a nearby market, the app integrates with a map application to show the route to the nearest store. The bottom navigation bar provides easy access to other pages within Checkout.

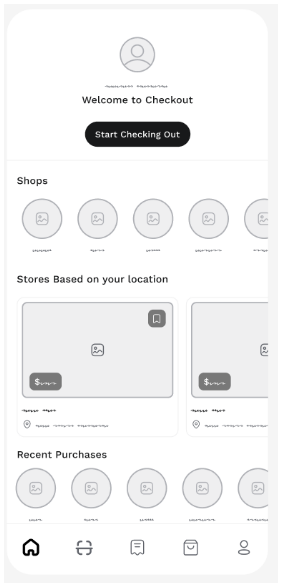

Fig. 2. Main Page

D. Profile

Once logged in, users can manage their personal information on the profile page. The profile feature allows users to update payment information, addresses, and other personal details. Users can view and edit their name, email, address, and payment methods in the 'My Profile' section. Additionally, users can add or modify payment methods, allowing them to keep their profile up-to-date for a smoother experience. If users select a payment method in their profile, they can use that payment method for checkout when they scan a barcode for payment.

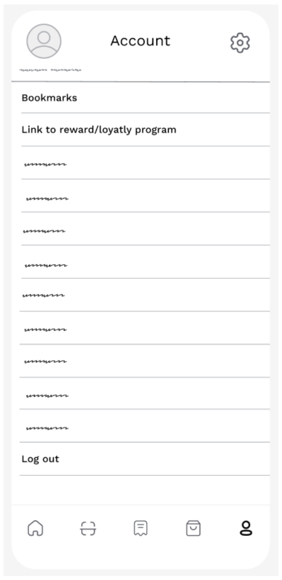

Fig. 3. Profile Page

E. Barcode Scan

Users can scan the barcode of products sold in supermarkets through the Checkout mobile web app. When they click the 'Barcode Scan' button, the mobile device's camera is activated to recognize the product's barcode. Once the barcode is scanned, the system displays real-time information about the product, such as its name, price, and stock. If users want to purchase the product, they can add it to their shopping cart.

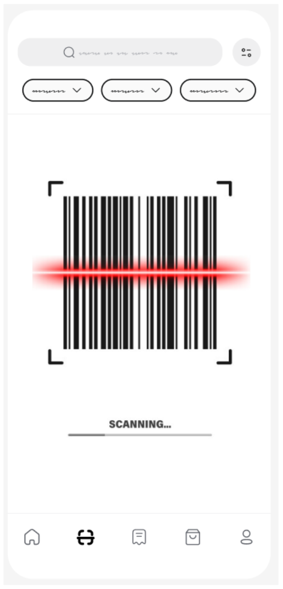

Fig. 4. Barcode Scanning Page

F. Shopping Cart

Users can add various products to their shopping cart either by scanning barcodes or searching for items. The shopping cart allows users to see all the products they've selected at a glance. Clicking on the 'Shopping Cart' menu shows a list of items in the cart and the total price. Users can adjust the quantity of items or remove unwanted products. This feature helps users review their selections before proceeding to checkout. After reviewing the cart, users can move to the checkout page to complete the purchase.

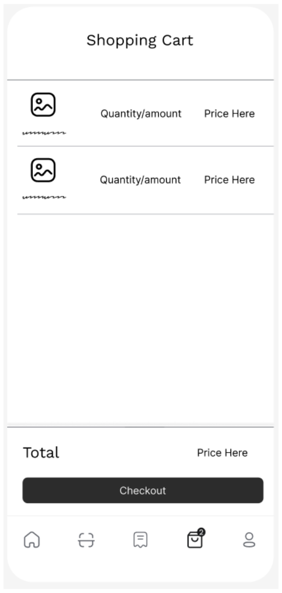

Fig. 5. Shoppingcart Page

G. Payment

Payment is the process in which users purchase the items in their shopping cart. On the payment page, users can choose from various payment methods, such as credit cards or mobile payment. After selecting the payment method (credit card, points, etc.), users enter their payment details, and the system processes the transaction. If everything is correct, the payment is completed successfully, and users receive a confirmation message along with a receipt. If the user has made previous payments or pre-selected a payment method in their profile, the system will automatically suggest the pre-selected method.

H. Receipts

Users can view their past purchases by accessing the 'My Receipts' section. The receipts include details such as the purchased items, prices, payment date, and payment method. This feature allows users to track their purchasing history and verify transaction details.

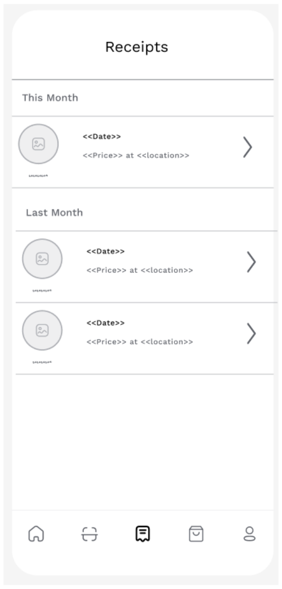
Receipts
This Month
<<Date>>
<<Price>> at <<location>>
Last Month
<<Date>>
<<Price>> at <<location>>
<<Date>>
<<Price>> at <<location>>

Fig. 6. Receipts Page

I. Card Registration

To speed up the checkout process, users can register their credit or debit card as a payment method. In the 'Card Registration' section, users can securely enter their card information (card number, expiration date, cardholder's name, etc.). Once registered, users can easily select the card for future payments without needing to input the information every time they make a purchase.

J. Coupon Usage

During the payment process, users can apply valid promotional codes or coupons to receive discounts. Users select the coupon they wish to apply on the payment page. The system verifies the code's validity and automatically applies the discount if it's valid. This allows users to save on their purchase, and the system ensures that the most relevant discount is applied in real time.