Republic of Tunisia

Ministry of Higher Education and Scientific Research

University of Sousse

Higher Institute of Management of Sousse (ISGS)

INTERNSHIP REPORT

For the completion of the License Degree in

Business Intelligence

POINT OF SALE SYSTEM FOR RETAIL MANAGEMENT

Internship carried out at:

Synthron Lab

Prepared by:
Ghassen Benali

Academic Supervisor:
Hana Rabbouch

Professional Supervisor:
Ahmed Haroun Baccar

Academic Year: 2023–2024

Acknowledgements

First and foremost, I would like to express my sincere gratitude to **Dr. Hana Rabbouch**, PhD in Business Computing and Assistant Professor at ISG Sousse, for her academic guidance and encouragement throughout this project.

I am also grateful to **Mr. Ahmed Haroun Baccar**, my professional supervisor at **Synthron Lab**, for trusting me with the project and providing the opportunity to work on a real-world business application.

My thanks also go to the faculty and staff of the **Institut Supérieur de Gestion de Sousse (ISGs)** for their support during my three years of study in Business Intelligence.

Finally, I would like to thank my family and close friends for their continued support, understanding, and motivation throughout my academic journey.

Contents

List of Figures

Chapter 1

General Project Framework

1.1 Introduction

In a constantly evolving digital economy, businesses are increasingly turning to software systems to automate operational processes, improve productivity, and gain competitive advantage. Among these, Point of Sale (POS) systems have become essential in the retail sector, providing functionality for transaction processing, inventory tracking, invoicing, and decision-making support.

This report documents the development of a modern web-based POS system created as part of a final-year internship required for the Business Intelligence degree. The internship was carried out remotely in collaboration with **Synthron Lab**, a software development company specializing in custom web applications and data-driven solutions.

1.2 Host Organization

Synthron Lab is a Tunisian software company that offers development and consulting services focused on modern technologies. The company builds full-stack web applications and integrates third-party platforms such as Power BI, Auth0, and cloud services to support both operational and strategic needs.

Its team is known for working with cutting-edge tools like React, Next.js, PostgreSQL, and scalable data architecture. Synthron Lab collaborates with both local and international clients in the retail, logistics, and analytics sectors.

1.3 Project Description

The project consists of designing and developing a web-based **Point of Sale (POS)** system targeted at small and medium-sized retailers. It allows:

- **Cashiers** to log in, browse products, add them to a cart, process payments, and generate invoices.
- **Administrators** to manage products, update inventory, and access live dashboards for business insights.
- **Secure access control** using authentication with dynamic role-based redirection.

The system offers a responsive and user-friendly interface and integrates business intelligence through an embedded Power BI dashboard for the admin users.

1.4 Study of Existing Solutions

An analysis of existing POS solutions highlights several limitations:

- Many interfaces are outdated and not mobile-friendly.
- Most systems lack built-in business intelligence capabilities.
- Traditional POS tools are often heavy, require installation, and are not cloud-accessible.
- Local solutions may be difficult to maintain or scale.

The proposed system addresses these problems by delivering a lightweight, web-accessible POS application that's easy to deploy, maintain, and integrate.

1.5 Proposed Solution

The proposed solution is a fullstack JavaScript web application built with:

- **Next.js** for routing, backend API, and page rendering.
- **React** for modular, component-based UI development.
- **Tailwind CSS** for fast, responsive UI styling.

- **PostgreSQL** for database management and transactional integrity.
- **Auth0** for secure authentication and session handling.
- **Power BI** for embedded analytics and decision dashboards.

Upon login, users are redirected based on their role. Cashiers access the POS dashboard, and administrators access an analytics panel with product and transaction management tools. Invoices are generated as downloadable PDFs with QR codes, and the system updates stock quantities in real time.

1.6 Project Objectives

The core objectives of the project are:

- Develop a responsive, modular POS application.
- Provide secure authentication with dynamic role management.
- Enable real-time stock management and product control.
- Automate invoice generation with integrated QR codes.
- Deliver integrated decision support through Power BI dashboards.

1.7 Project Management Methodology

The project was conducted using the **Scrum** agile methodology. Scrum promotes iterative development cycles known as *sprints*, each delivering a working software increment. This allowed for focused development, fast feedback, and continuous improvement.

Sprint Planning

The project was divided into the following five sprints:

- **Sprint 0** – Requirements gathering, existing system analysis, architecture planning.
- **Sprint 1** – Implementation of authentication and role-based redirection.
- **Sprint 2** – Development of cashier dashboard and product cart system.
- **Sprint 3** – Invoice generation, stock updates, and PDF generation.

- **Sprint 4** – Admin dashboard integration with Power BI analytics.

Each sprint lasted approximately one week and followed the Scrum lifecycle: sprint planning, execution, testing, and review.

1.8 Modeling Languages Used

To plan and document the architecture and flow of the system, **UML (Unified Modeling Language)** was used. The following types of diagrams were created using PlantUML:

- **Use Case Diagrams** – To identify system actors and interactions.
- **Sequence Diagrams** – To describe system workflows and communication.
- **Class Diagrams** – To define the data structure and relationships between entities.

BPMN (Business Process Model and Notation) was not used in this project, but is acknowledged as a standard for process modeling. If the project were extended to include business process automation, BPMN diagrams could be applied for workflow definition.

1.9 Conclusion

This first chapter has introduced the overall framework of the project, including the host organization, project scope, existing system analysis, goals, and methodology. The following chapter will present the expression of functional and technical requirements, technical architecture, and development environment setup as part of Sprint 0.

SWOT Analysis – POS System

Strengths

- Web-based and responsive design accessible from any device
- Secure authentication and role-based access control via Auth0
- Embedded analytics with Microsoft Power BI for real-time insights
- Modern tech stack (React, Next.js, Tailwind CSS, PostgreSQL)

Weaknesses

- No native mobile application
- Does not currently support barcode scanning or receipt printers
- Limited user role granularity beyond admin/cashier
- Requires internet connectivity for all features (no offline mode)

Opportunities

- Add support for new hardware: barcode scanners, printers, cash drawers
- Expand to mobile app or PWA (Progressive Web App)
- Introduce role editor and multi-store functionality
- Enhance analytics with predictive dashboards or AI integration

Threats

- Security risks related to token handling or authentication leaks
- Performance issues under large-scale use without optimization
- Competitive POS platforms offering similar features
- Risk of dependency on external tools like Power BI or Auth0 pricing

Chapter 2

Requirements and Technical Study (Sprint 0)

2.1 Introduction

Before implementation began, a detailed analysis phase was conducted to identify functional needs, define technical architecture, and prepare the development environment. This phase, referred to as **Sprint 0**, laid the foundation for the subsequent iterative sprints and ensured that the project was aligned with both business objectives and technical constraints.

2.2 Identification of Requirements

2.2.1 Actors and User Roles

The system defines two distinct roles:

- **Cashier**: Executes sales operations, adds products to the cart, processes payments, and generates invoices.
- **Administrator**: Manages product inventory, updates stock, and visualizes sales data through analytics dashboards.

2.2.2 Functional Requirements

- Login interface with secure authentication.

- Role-based redirection to either POS or admin dashboard.
- Product browsing with dynamic cart system.
- Invoice generation with QR code.
- Admin interface for product and stock management.
- Embedded analytics dashboards for real-time insights.

2.2.3 Non-Functional Requirements

- Responsive design for desktop and tablet devices.
- Secure access with role-based control.
- Fast data fetching and UI rendering.
- Clean and intuitive user interface.

2.2.4 Decision-Oriented Requirements

- Power BI integration for sales trends and product performance.
- Ability to view top-selling items and total revenue per user.
- Dashboard embedded directly in admin panel.

2.3 Scrum Planning and Project Setup

2.3.1 Team Structure

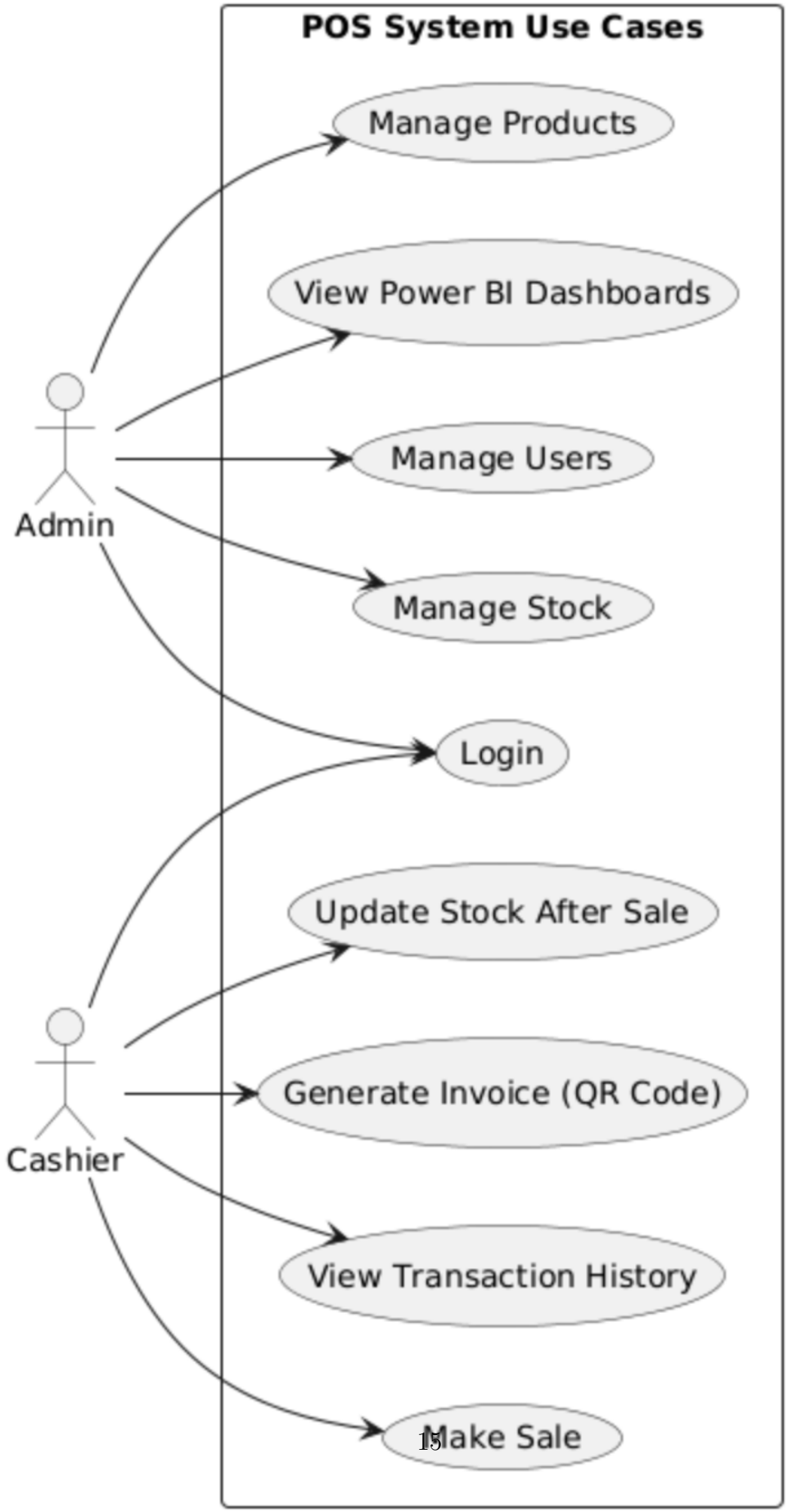
Although the internship was individual, the Scrum methodology was simulated with the following roles:

- **Developer (intern)**: Handles the entire software development lifecycle.
- **Professional Supervisor**: Guides technical decisions and validates deliverables.
- **Academic Supervisor**: Follows academic progression and report quality.

2.3.2 Sprint 0 Objectives

- Define user stories and identify actors.
- Specify system features.
- Design global use case diagram.
- Set up development tools and repository.
- Choose technology stack and define architecture.

2.4 Global Use Case Diagram



2.5 Technical Environment

2.5.1 Project Methodology – Scrum

The Scrum framework was used to manage the project across five iterative sprints. Each sprint lasted approximately one week and included planning, execution, testing, and review.

Scrum Sprint Structure:

- Sprint 0 – Planning and environment setup
- Sprint 1 – Authentication and role management
- Sprint 2 – POS dashboard and cart logic
- Sprint 3 – Invoicing and stock updates
- Sprint 4 – Analytics dashboard with Power BI

2.5.2 Tools and Technologies Used

React

Figure 2.2: React Logo

React is a JavaScript library for building user interfaces. It powered the cashier/admin dashboards and rendered dynamic views.

Next.js

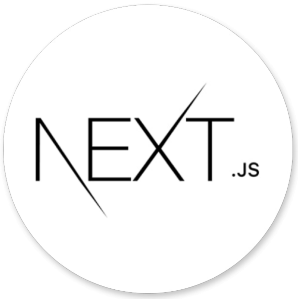


Figure 2.3: Next.js Logo

Next.js served as the fullstack framework, providing routing, backend API endpoints, and deployment support.

Tailwind CSS

Figure 2.4: Tailwind CSS Logo

Tailwind CSS was used for UI styling and responsiveness across devices.

PostgreSQL

Figure 2.5: PostgreSQL Logo

PostgreSQL stored user, product, invoice, and transaction data.

Power BI

Figure 2.6: Power BI Logo

Power BI dashboards were embedded into the admin panel for live decision-making.

Auth0

Figure 2.7: Auth0 Logo

Auth0 handled secure login, session management, and access control.

Visual Studio Code

Figure 2.8: Visual Studio Code Logo

VS Code was the main IDE used for development and debugging.

PlantUML

Figure 2.9: PlantUML Logo

PlantUML generated all UML diagrams used in the documentation.

2.5.3 Tool Summary Table

Logo	Tool Name	Purpose / Use Case
Visual Studio Code	VS Code	Source code editing, React/Tailwind development
React	React	Frontend UI for POS and admin dashboards
NEXT.js	Next.js	Fullstack framework, backend API, routing
Tailwind CSS	Tailwind CSS	Styling UI with responsive design classes
PostgreSQL	PostgreSQL	Relational DB for product, stock, invoices
Power BI	Power BI	Embedded sales and analytics dashboards
auth0	Auth0	Secure login and role-based routing
UML	PlantUML	UML diagram generation for documentation

Table 2.1: Summary of Technologies and Tools

2.5.4 System Architecture Overview

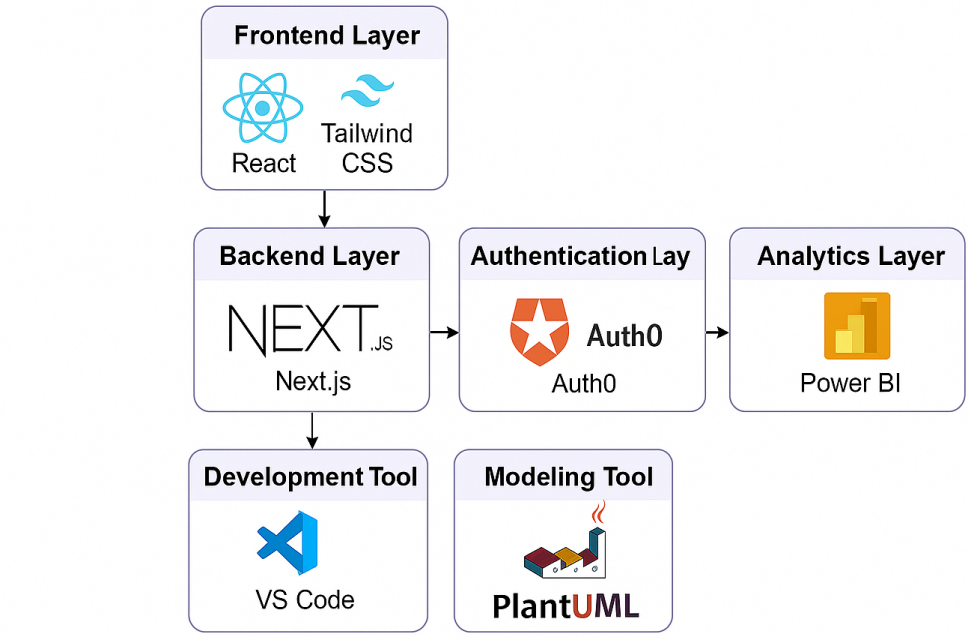


Figure 2.10: System Tool Integration Diagram – POS Architecture

The architecture demonstrates:

- **Frontend Layer:** React and Tailwind CSS in a Next.js app.
- **Backend Layer:** Next.js API routes with database access.
- **Authentication Layer:** Auth0 for secure login and role-based redirection.
- **Analytics Layer:** Embedded Power BI dashboard for admins.

2.6 Conclusion

Sprint 0 helped define the project's direction by formalizing the requirements, identifying user roles, and setting up the full technical environment. This foundation allowed the development phase to proceed in sprints, each targeting a functional milestone. The next chapter details the work carried out in Sprint 1: user authentication and secure role-based redirection.

Chapter 3

Sprint 1 – Authentication and Role Management

3.1 Introduction

The first sprint focused on implementing secure user authentication and redirecting users to the appropriate interface based on their role. This sprint was essential for laying the groundwork of access control and system segmentation between cashiers and administrators.

The goal was to allow users to log in through a trusted identity provider (Auth0), verify their role securely, and then route them to the corresponding interface without exposing protected content to unauthorized users.

3.2 Sprint 1 Backlog

Task ID	Task Description	Status
S1-T1	Integrate Auth0 authentication into the login system	Completed
S1-T2	Define and assign user roles (admin, cashier) via metadata	Completed
S1-T3	Implement role-based redirection after login	Completed
S1-T4	Protect frontend and backend routes using user role checks	Completed

Table 3.1: Sprint 1 – Backlog

3.3 Functional Specification

Users accessing the system are prompted to log in via a secure OAuth provider (Auth0). Once authenticated, the application retrieves their role and conditionally renders either:

- The cashier dashboard: a simple POS interface focused on transactions.
- The admin dashboard: an interface with access to management and analytics.

Roles are determined using email-based matching or predefined metadata in Auth0.

3.4 Use Case Diagram

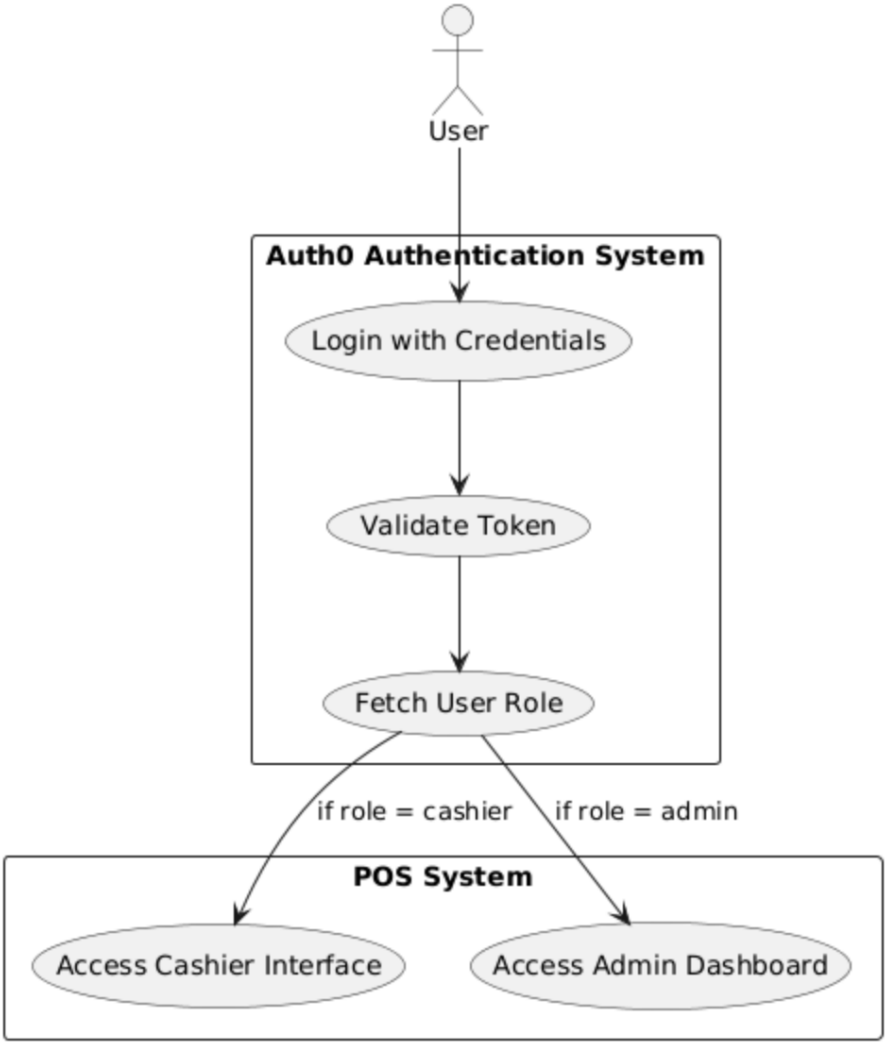


Figure 3.1: Use Case Diagram – Authentication and Role Routing

3.5 Sequence Diagram

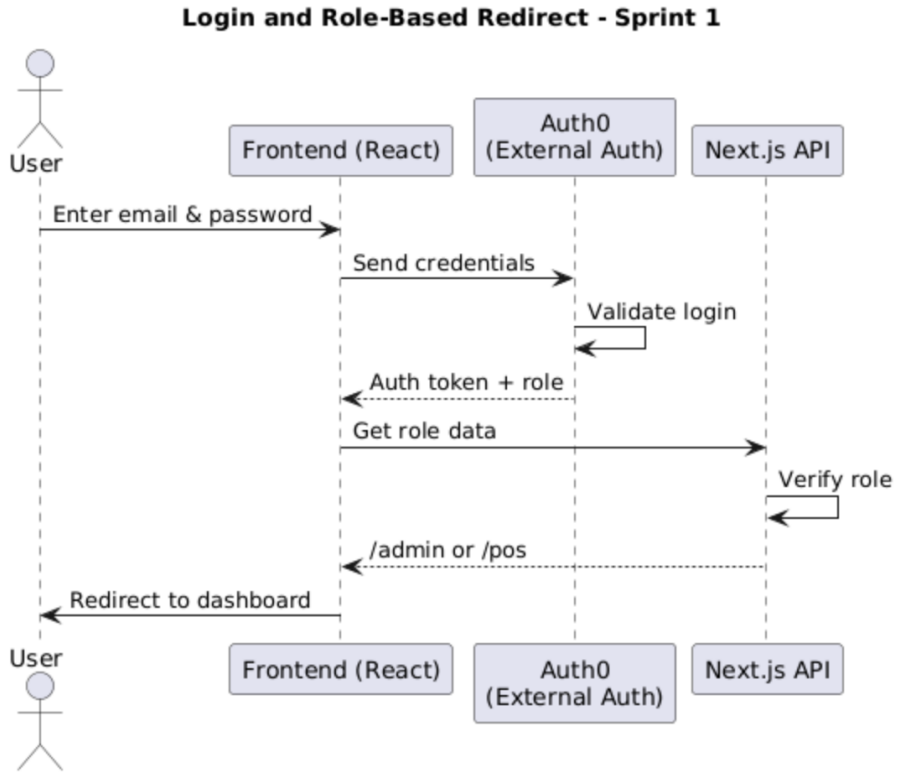


Figure 3.2: Sequence Diagram – Improved Role-Based Login Flow

3.6 Implementation Highlights

The integration process was straightforward but required careful handling of session state and route protection. Here's a breakdown of key implementation tasks:

Auth0 Integration

The Auth0 React SDK was used to enable secure OAuth login. After successful login, a JSON Web Token (JWT) is generated and used to verify the user's identity across protected routes.

Role Definition and Routing

Each authenticated user is assigned a role. A backend rule determines whether the user should access the cashier interface or the admin dashboard. Roles are either stored in

user metadata or inferred from email domains.

Protected Routes

Using Next.js middleware and client-side checks, unauthorized access is prevented. Attempting to access admin content without the right role results in an automatic redirect or access denial.

UI Feedback and Session Handling

Sessions persist during browser refresh, and the logout process fully clears the user token. A visual feedback component informs users of their login status and destination.

3.7 Testing and Results

Manual testing was conducted on local builds to simulate both user roles. The following scenarios were verified:

- Login as a cashier → redirected to POS dashboard.
- Login as an admin → redirected to admin dashboard.
- Unauthorized access to /admin or /pos routes → blocked or redirected.
- Logout → session cleared and returned to login screen.

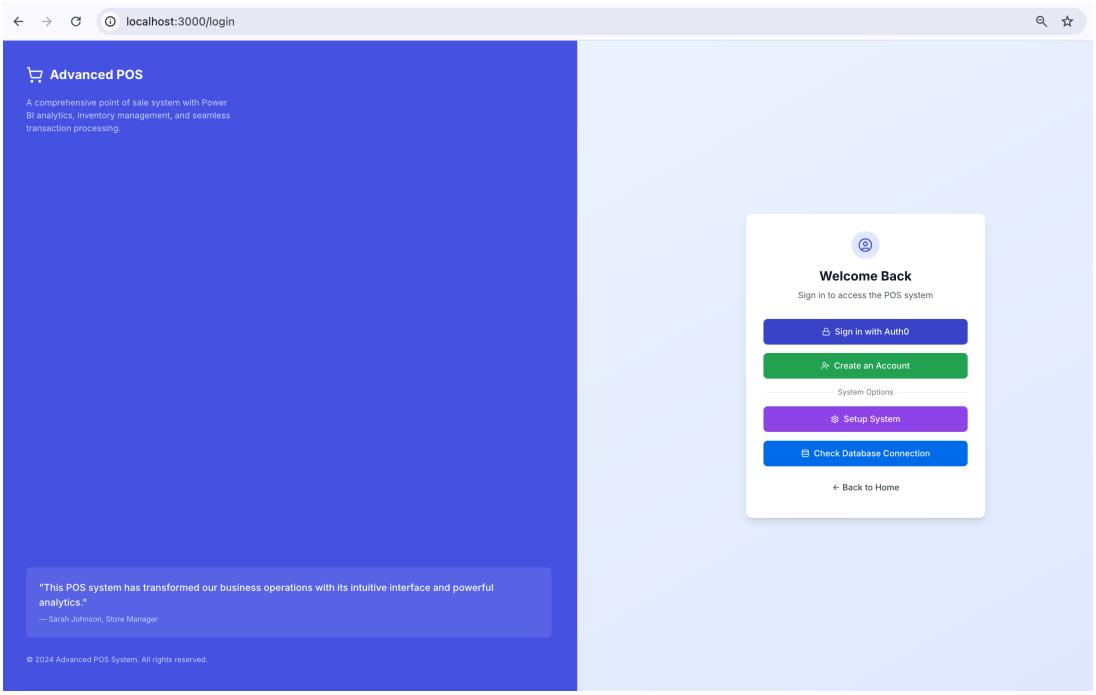


Figure 3.3: Login Interface – Auth0 Integration Test

The authentication and routing system worked reliably throughout this sprint. No major issues were encountered once session persistence and route guards were in place.

3.8 Conclusion

Sprint 1 successfully delivered the core authentication layer and dynamic role-based routing logic. This functionality not only enhances security but also defines the foundation for all user interactions in later sprints. With user access in place, the next sprint will focus on building the main cashier interface and enabling product selection and transaction logic.

Chapter 4

Sprint 2 – POS Interface and Product Management

4.1 Introduction

The second sprint focused on building the core interface for cashiers: a clean, responsive dashboard for browsing products, managing a cart, and preparing transactions. While Sprint 1 established secure access and user redirection, Sprint 2 addressed how cashiers interact with products in real time.

This sprint was especially important in shaping the user experience (UX) of the POS system, as the cashier dashboard would be used frequently and required both performance and visual clarity.

4.2 Sprint 2 Backlog

Task ID	Task Description	Status
S2-T1	Design product grid with image, name, and price	Completed
S2-T2	Fetch product data dynamically via API	Completed
S2-T3	Implement cart logic with quantity updates and removal	Completed
S2-T4	Calculate and display total amount in real time	Completed
S2-T5	Ensure UI responsiveness for different screen sizes	Completed

Table 4.1: Sprint 2 – Backlog and Implementation Plan

4.3 Functional Specification

The POS interface was designed for cashiers to carry out sales operations efficiently. The dashboard layout includes:

- A dynamic product grid fetched from the database.
- Quantity selectors and “add to cart” buttons.
- A side panel showing the current cart with items, prices, and total.
- Action buttons for confirming or clearing a sale.

Each element was styled using Tailwind CSS to ensure a modern and responsive layout.

4.4 Class Diagram

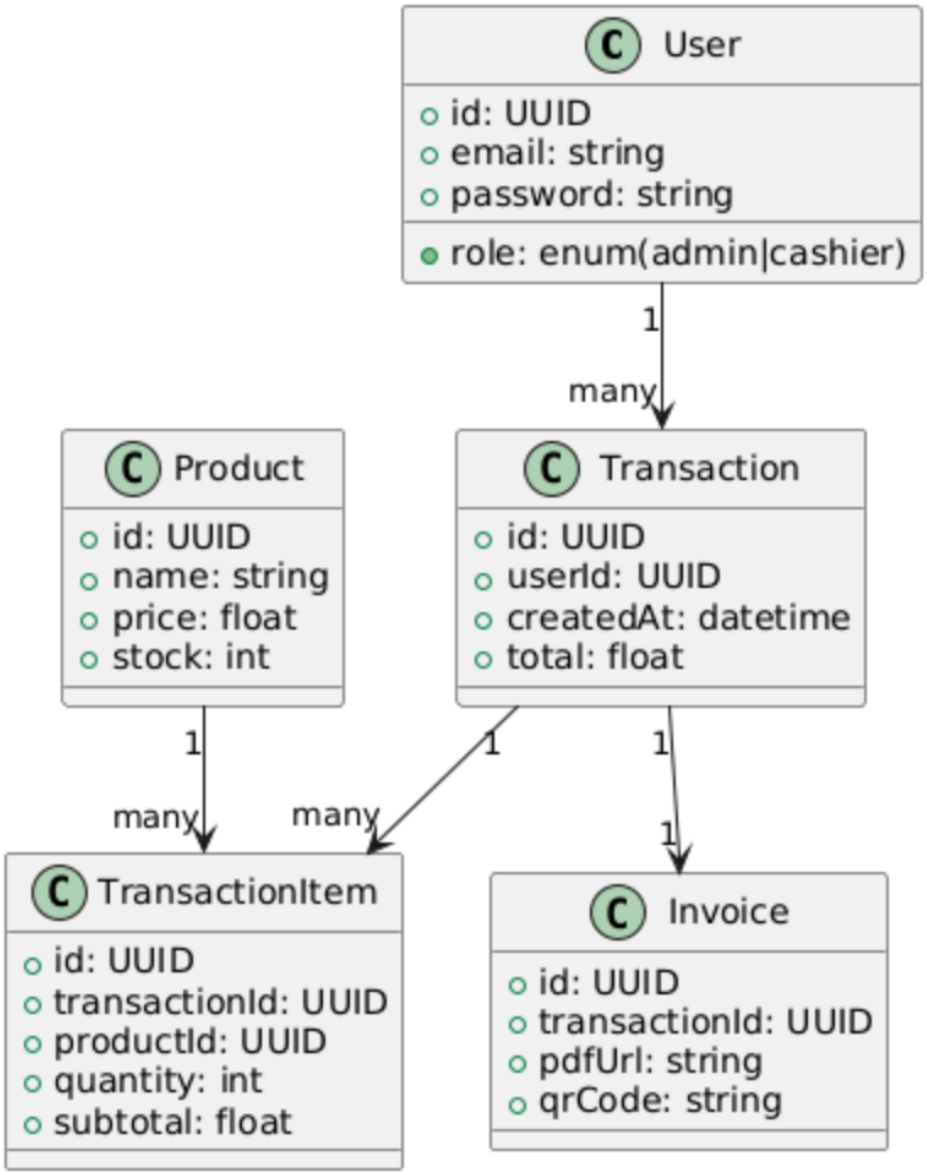


Figure 4.1: Class Diagram – Product and Cart Relationships

This diagram shows the structure of key entities: `Product` stores attributes like name, image, stock, and price. `CartItem` maintains selected quantity and total for each product. `Cart` aggregates multiple `CartItem` instances for the current transaction.

4.5 Sequence Diagram

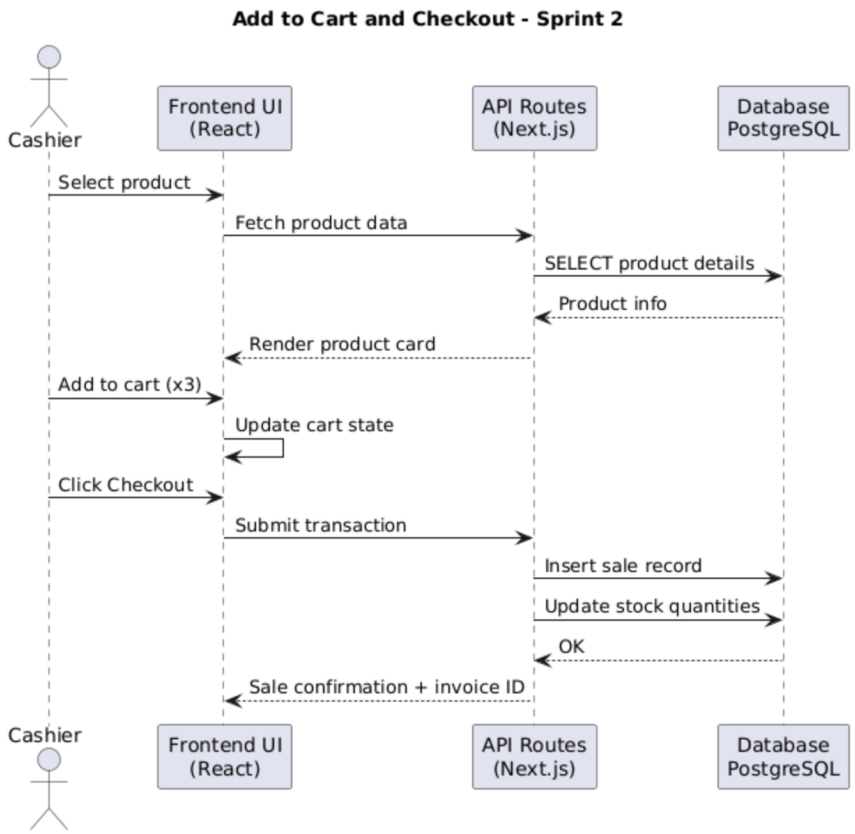


Figure 4.2: Sequence Diagram – Adding a Product to Cart (Improved)

This diagram outlines the flow:

1. Cashier selects a product.
2. Quantity is adjusted.
3. Cart is updated and UI is re-rendered.
4. Total amount is recalculated automatically.

4.6 Implementation Highlights

Frontend Design with Tailwind

A responsive product grid was implemented using Tailwind CSS grid classes. Each product was displayed in a card with its image, name, and price, along with quantity buttons.

State Management

React's `useState` hook was used to maintain cart state in memory. When a product was added, the cart array was updated and re-rendered.

API Integration

Products were fetched from the database using a custom Next.js API route. The response returned an array of product objects with image paths and pricing.

Edge Case Handling

Edge cases such as "out of stock" products or attempts to add the same product multiple times were handled through validation logic and UI feedback.

4.7 Mockup and Interface Screens

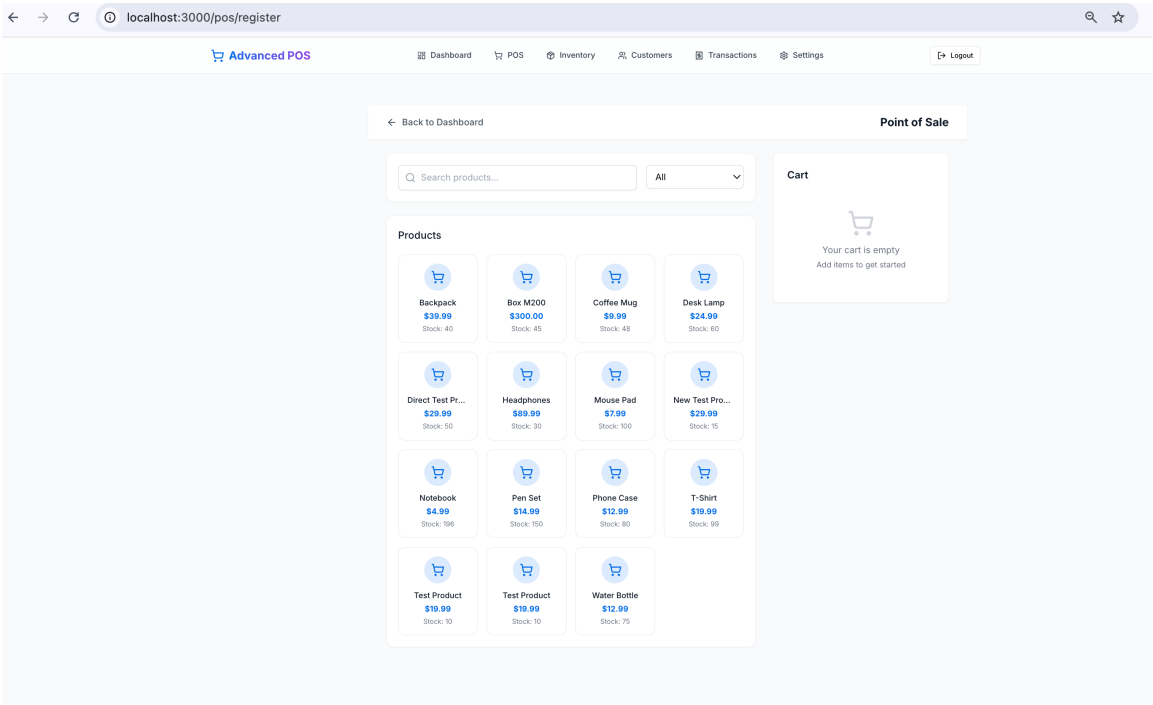


Figure 4.3: Sprint 2 – Cashier POS Dashboard and Cart System

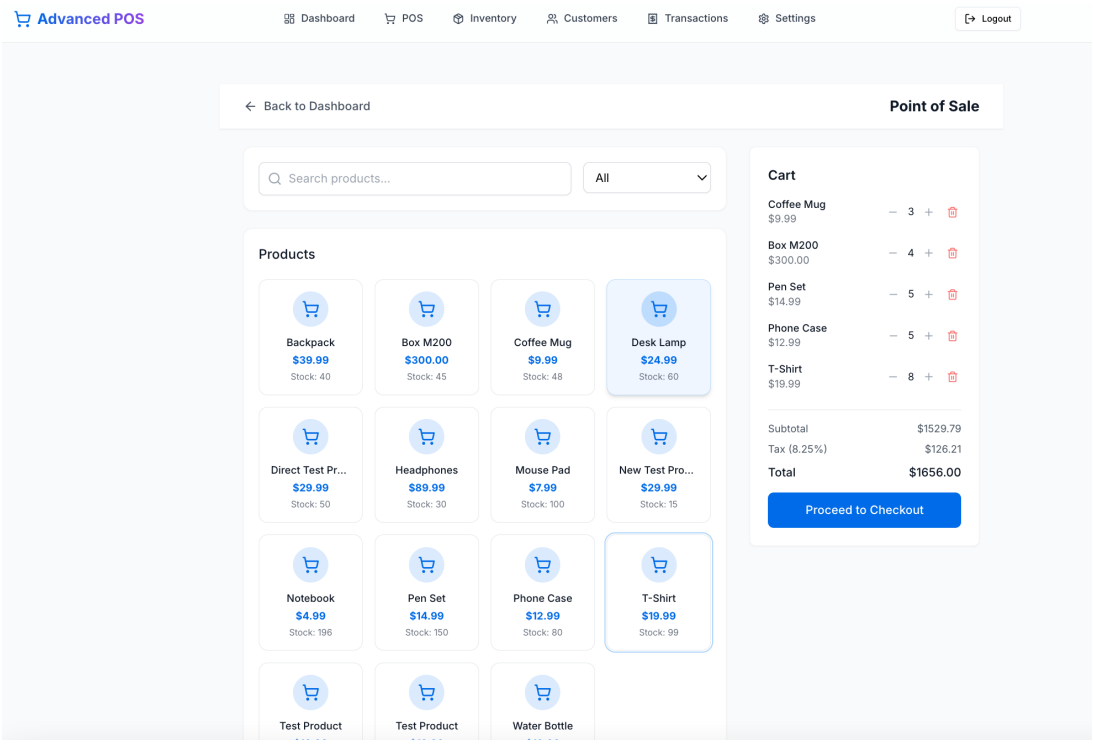


Figure 4.4: Cart View – After Adding Multiple Products

4.8 Testing and Feedback

Manual testing was performed on:

- Adding multiple products and adjusting quantities.
- Removing products from the cart.
- Ensuring total amount updates immediately.
- Responsive layout on different screen widths.

Feedback from supervisors suggested keeping the UI minimal to prioritize performance and visual clarity. The final version met these expectations.

4.9 Conclusion

Sprint 2 delivered the full cashier interface, marking a major milestone in the project. Cashiers can now browse products, manage a cart, and prepare a transaction in an intuitive and responsive layout. The next sprint will focus on payment confirmation, PDF invoice generation, and automatic stock updates.

Chapter 5

Sprint 3 – Invoicing and Stock Management

5.1 Introduction

In Sprint 3, the focus shifted from user interface development to the core transactional logic of the system. This phase introduced the checkout flow: confirming sales, generating invoices, updating inventory, and exporting PDF receipts. These features are essential to the core purpose of any POS system — completing a sale accurately and reliably.

This sprint also reinforced business rules, such as ensuring product quantities are valid and stock levels are kept up to date across all transactions.

5.2 Sprint 3 Backlog

Task ID	Task Description	Status
S3-T1	Implement checkout modal and payment confirmation flow	Completed
S3-T2	Generate invoice with QR code in PDF format	Completed
S3-T3	Automatically deduct product quantities from stock	Completed
S3-T4	Handle errors and success states in UI	Completed

Table 5.1: Sprint 3 – Backlog and Planning

5.3 Functional Specification

The goal of this sprint was to simulate the full lifecycle of a sale:

1. Cashier clicks the “Pay” button.
2. A modal appears to select payment type (cash, card, wallet).
3. A unique invoice is generated, assigned an ID, and saved.
4. A PDF version is created and downloaded, including a QR code.
5. The product stock is updated automatically in the database.

The process had to be fast, error-free, and simple for the cashier to use under pressure.

5.4 Sequence Diagram

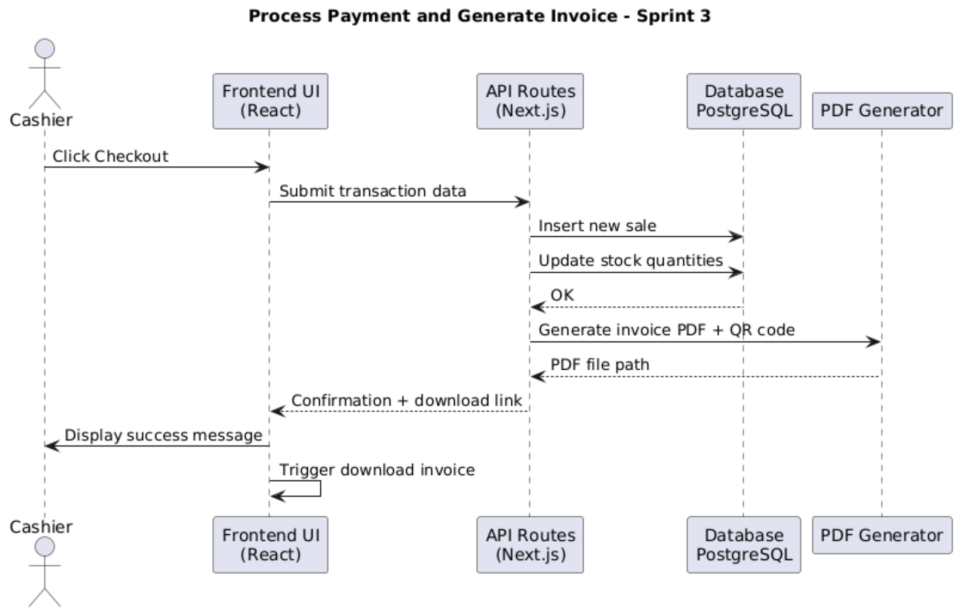


Figure 5.1: Sequence Diagram – Invoice Generation and Stock Update (Improved)

5.5 Implementation Highlights

Invoice Structure

Each invoice includes:

- Date and time of sale

- Product names and quantities
- Total amount
- Payment method
- QR code with transaction reference

PDFs are generated on the fly using a server-side library and styled for thermal printing if needed.

QR Code Generation

A unique identifier is included in the invoice and encoded as a QR code. This can later be scanned to retrieve transaction details, validate payment, or issue refunds.

Stock Synchronization

After each confirmed payment, the system calculates the purchased quantities and immediately updates the corresponding product entries in the database to reflect the new stock level.

Payment Modal and UX Flow

The payment modal was built to support keyboard navigation and prevent duplicate submissions. Success and failure messages are visually distinct, and the cart is cleared only after confirmation.

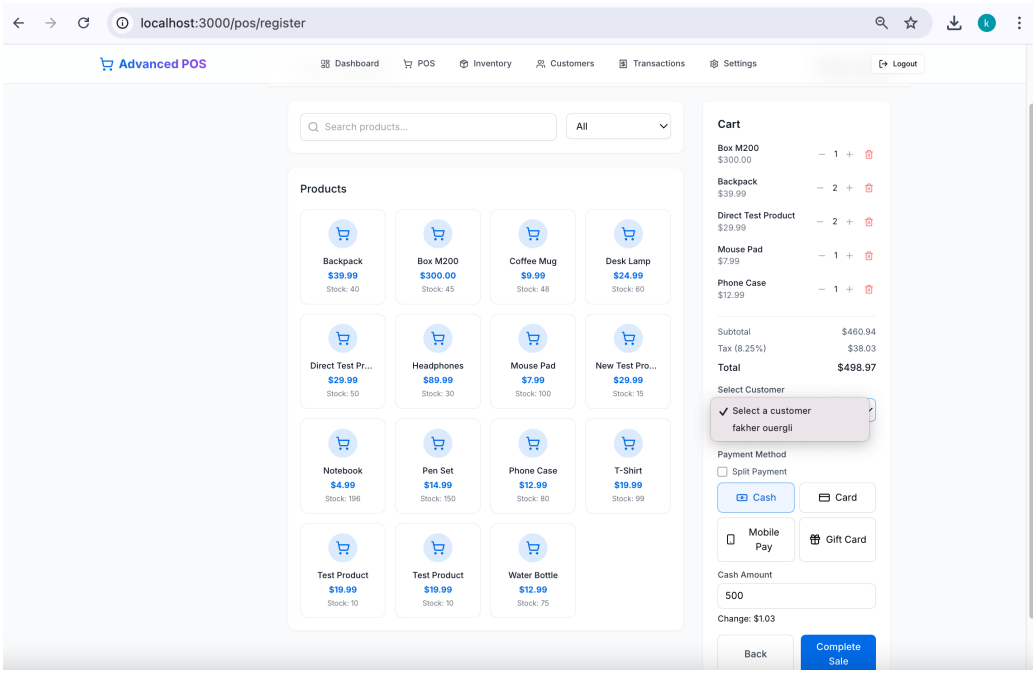


Figure 5.2: Checkout Modal – Payment Selection

5.6 Testing and Result Validation

Test cases covered:

- Invoice is created only if a product is in the cart.
- Stock updates are accurate after each transaction.
- PDF is generated and downloads without delay.
- QR code is scannable and decodes to transaction ID.
- Sale cannot proceed if stock is insufficient.

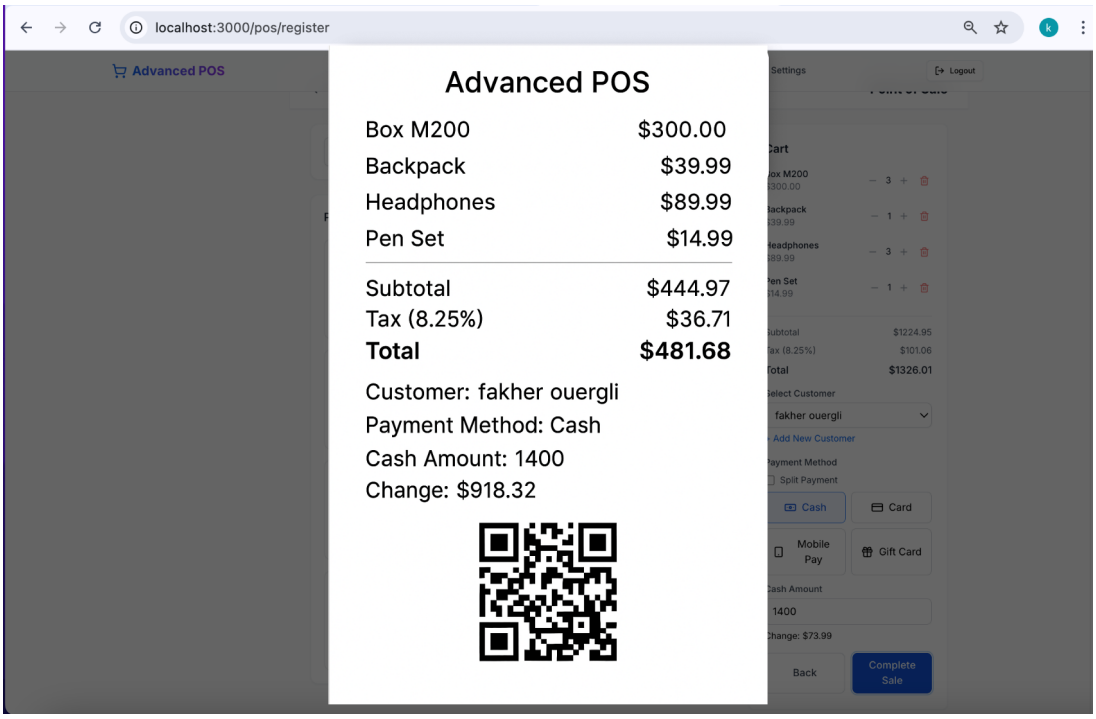


Figure 5.3: Generated Invoice with Embedded QR Code

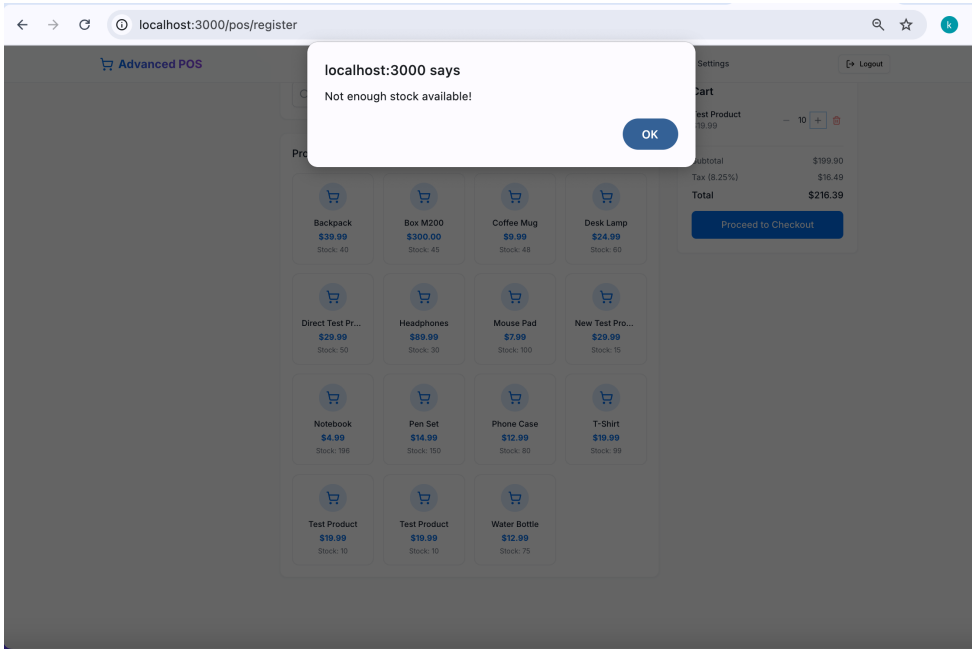


Figure 5.4: Stock Error Message – Out of Stock Validation

5.7 Conclusion

This sprint delivered a functional checkout workflow, including invoice generation and automatic stock updates — two of the most important features of the system. With this

logic in place, the POS can now be used end-to-end. The next and final sprint will focus on business analytics and embedding Power BI dashboards into the admin panel.

Chapter 6

Sprint 4 – Business Intelligence and Finalization

6.1 Introduction

In the final sprint, the focus moved from operational functionality to strategic visibility. The goal was to provide administrators with a visual interface to monitor performance, track sales trends, and make data-informed decisions. This was achieved by integrating dashboards built using Microsoft Power BI directly into the application.

Additionally, this sprint included final UI polishing, user experience refinements, and manual system-wide testing to ensure the platform was production-ready.

6.2 Sprint 4 Backlog

Task ID	Task Description	Status
S4-T1	Integrate Power BI dashboards into the admin panel	Completed
S4-T2	Create transaction history view for admin and cashier roles	Completed
S4-T3	Conduct manual testing and fix critical bugs	Completed
S4-T4	Review UI/UX and prepare for final delivery	Completed

Table 6.1: Sprint 4 – Backlog and Finalization Tasks

6.3 Business Intelligence Goals

The integration of BI dashboards aimed to:

- Monitor product performance (top sellers, low stock).
- Track total revenue and payment breakdowns.
- Provide per-cashier activity summaries.
- Enable real-time visibility for business stakeholders.

Power BI was selected due to its cloud accessibility, ease of integration, and robust data visualization features.

6.4 Sequence Diagram – Analytics Integration

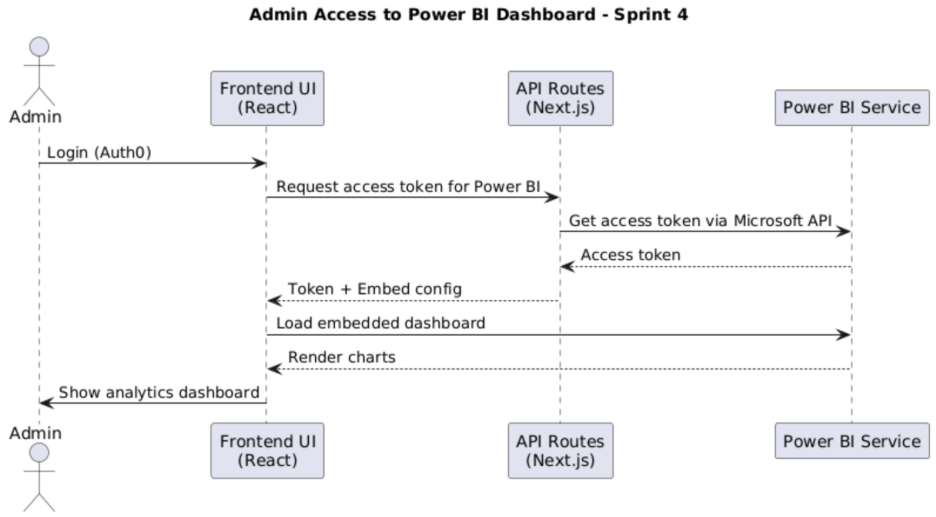


Figure 6.1: Sequence Diagram – Analytics Dashboard Integration (Improved)

6.5 Integration of Power BI

Power BI reports were embedded into the admin interface using secure embed tokens and iframe components. The dashboard loads automatically upon admin login and displays charts sourced from the PostgreSQL database.

Dashboard Highlights

- Total sales overview with date filters

- Bar chart of most sold products
- Pie chart by payment type (cash, card, wallet)
- Line graph showing sales per day/week

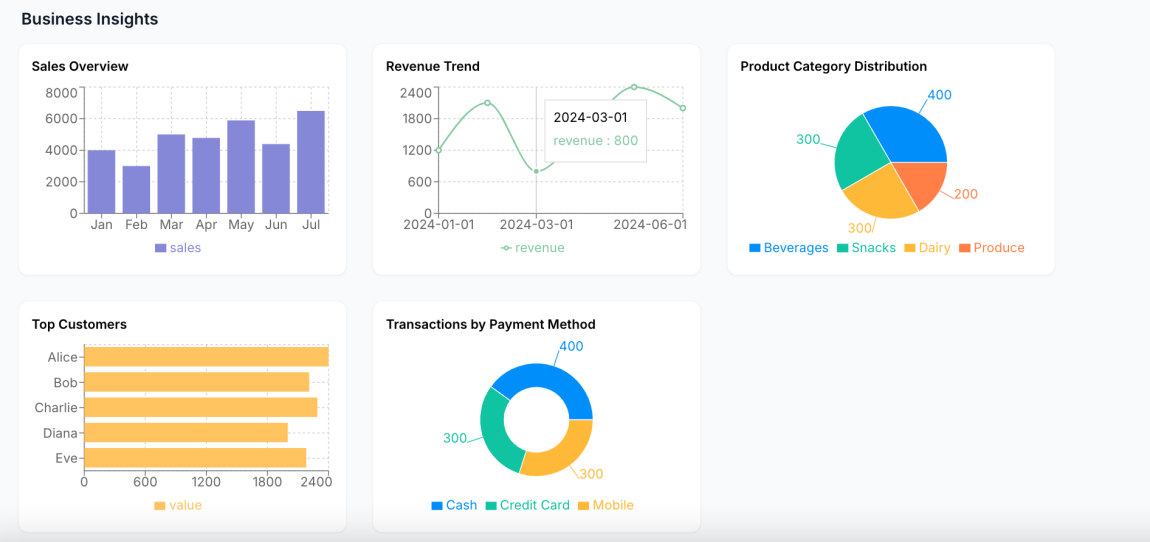


Figure 6.2: Power BI Dashboard Embedded in Admin Panel

6.6 Transaction History Page

To complement the BI features, a transaction history table was created. Cashiers can see only their own transactions, while admins can view all. This log includes date, total amount, and payment method.

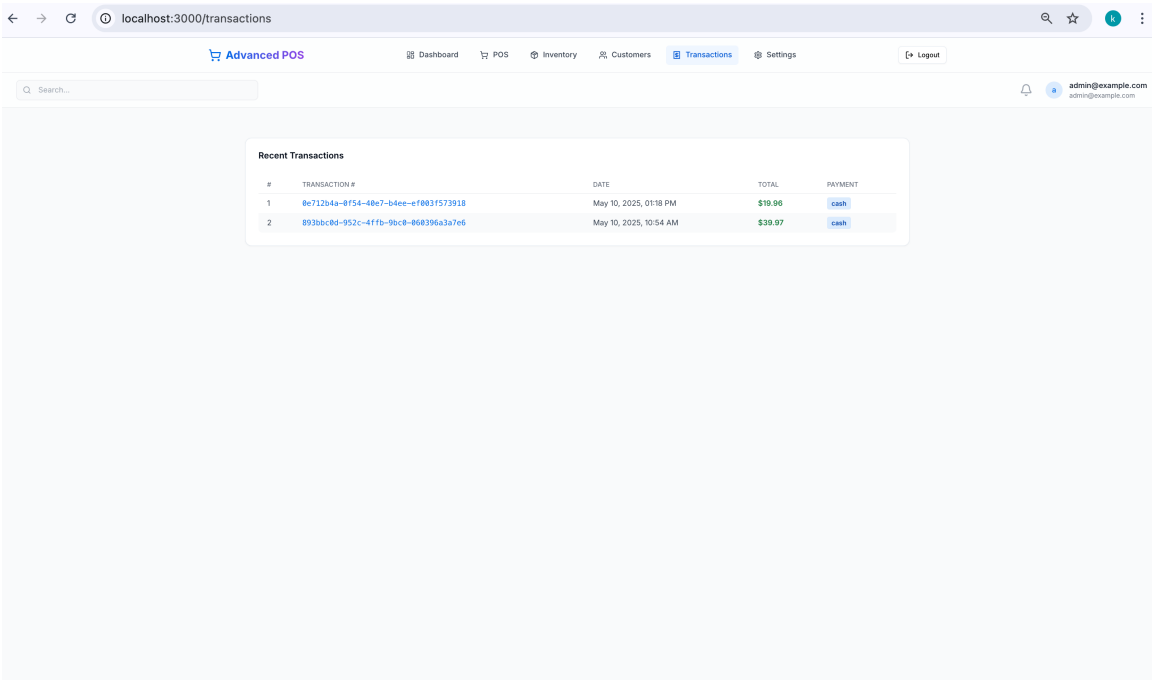


Figure 6.3: Transaction History Table – Admin View

6.7 Testing and Polishing

A final testing session was conducted to:

- Navigate through all flows with both roles
- Confirm consistent UI responsiveness
- Verify data accuracy in dashboards
- Ensure PDF generation and QR code functionality still worked

Minor bugs were fixed related to:

- Cart not clearing after payment
- Inconsistent invoice numbering on refresh
- Dashboard not loading correctly on first login

6.8 Burndown and Validation

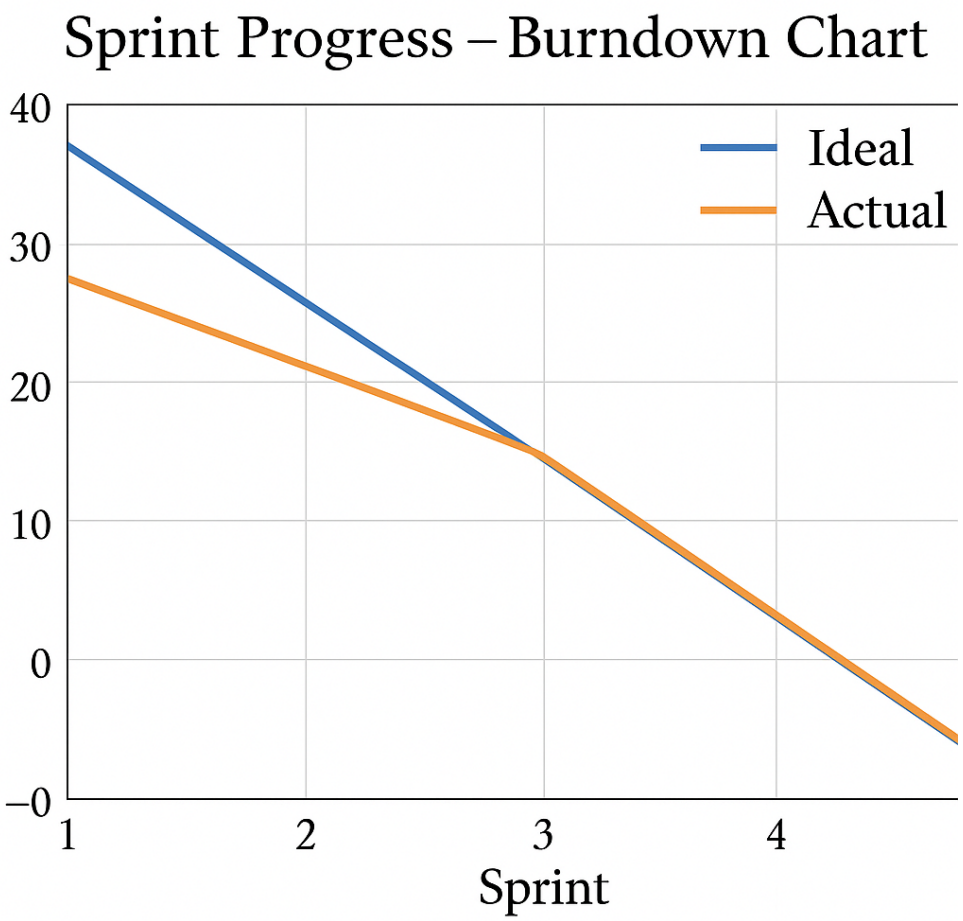


Figure 6.4: Burndown Chart – Overall Sprint Progress

All features targeted at the beginning of the sprint were completed and validated by the professional supervisor.

6.9 Conclusion

Sprint 4 marked the final phase of the POS system development. By embedding Power BI, the application now supports not only operational flows but also decision-making. The platform is fully functional, visually consistent, and aligned with the project objectives.

General Conclusion

This internship project marked the culmination of my academic journey in Business Intelligence and offered a valuable opportunity to apply both technical and analytical skills in a practical setting. Throughout the development of the POS system, I was able to explore full-stack web technologies, agile project management, and decision-support integration — all within the scope of a real-world application.

The initial phase of the project allowed me to analyze business needs, define clear requirements, and plan technical architecture using tools like UML and Scrum methodology. Each sprint was structured to address a specific feature of the system, and with each iteration, I gained more insight into planning, building, and validating a scalable web application.

Among the key achievements were:

- A secure login and role-based routing system via Auth0.
- A functional and responsive POS interface for cashiers.
- Automatic invoice generation with real-time stock updates.
- Embedded Power BI dashboards for business analytics.

Working remotely with Synthron Lab allowed me to develop independently while following professional standards. I learned how to manage time effectively, test iteratively, and produce reliable and readable code. I also experienced the value of structured documentation and sprint-based development in keeping complex projects under control.

While the application is already functional and complete in its current form, several improvements can be explored in future versions.

Future Enhancements

- Add multi-store and multi-cashier support for larger retail networks.

- Integrate barcode scanning for faster item lookup.
- Implement offline mode with local caching and sync.
- Connect with payment APIs for real-time transaction validation.
- Expand BI dashboards to include predictive metrics and export options.

Closing Remarks

This internship has been both a technical challenge and a personal milestone. I had the chance to turn an abstract concept into a complete, usable solution, and this has significantly reinforced my confidence in software design, backend logic, and user-centered thinking.

I am grateful to my academic supervisor and professional mentor for their support, and I leave this experience with a stronger foundation for my career in tech and analytics.

Glossary

This glossary defines key technical terms, acronyms, and tools referenced throughout the report.

API Application Programming Interface – A set of rules that allow different software components to communicate with each other.

Auth0 A secure authentication service used to manage user login, sessions, and role-based access control.

Backlog A prioritized list of tasks or features to be implemented in upcoming sprints during agile development.

BI (Business Intelligence) Technologies and practices for analyzing business data and presenting actionable insights through dashboards and reports.

Cart System The interface logic that manages product selection, quantity, and totals during a transaction in the POS.

Dashboard A visual interface that displays key performance indicators, statistics, and analytics for decision-making.

JWT JSON Web Token – A compact and secure format for transmitting identity and session information between parties.

Next.js A full-stack React framework used in the project to handle frontend rendering and backend API routing.

PDF Invoice A downloadable document generated after a sale, containing transaction details and a QR code for tracking.

PlantUML A text-based diagram tool used to create UML diagrams such as class, sequence, and use case diagrams.

POS (Point of Sale) The system used to complete sales transactions, manage inventory, and generate invoices in retail environments.

PostgreSQL A relational database system used to store structured data such as users, products, stock levels, and transactions.

Power BI A Microsoft business analytics tool used in this project to visualize and embed real-time sales and performance dashboards.

QR Code A machine-readable code used in the invoices for referencing the transaction or verifying it later.

React A JavaScript library for building user interfaces using reusable components.

Scrum An agile framework used to manage project development in iterative sprints with a focus on collaboration and incremental delivery.

Sprint A short, time-boxed period (typically one week) in Scrum used to deliver a specific, functional increment of the project.

Tailwind CSS A utility-first CSS framework used for styling the POS interface with speed and consistency.

UML Unified Modeling Language – A set of standardized diagrams used to model the structure and behavior of a system.

VS Code Visual Studio Code – The integrated development environment used for writing and debugging the application code.

Bibliography

[1] Microsoft Docs. Embed power bi in web applications. https://learn.microsoft.com/en-us/power-bi/developer/embedding/, 2024.

[2] Auth0 Documentation. Secure authentication for web applications. https://auth0.com/docs, 2024.

[3] Tailwind Labs. Tailwind css – rapid utility-first styling. https://tailwindcss.com/docs, 2024.

[4] Vercel. Next.js documentation. https://nextjs.org/docs, 2024.

Annexes

This annex includes supporting materials related to the implementation and testing of the POS system.

Annex A – System Screenshots

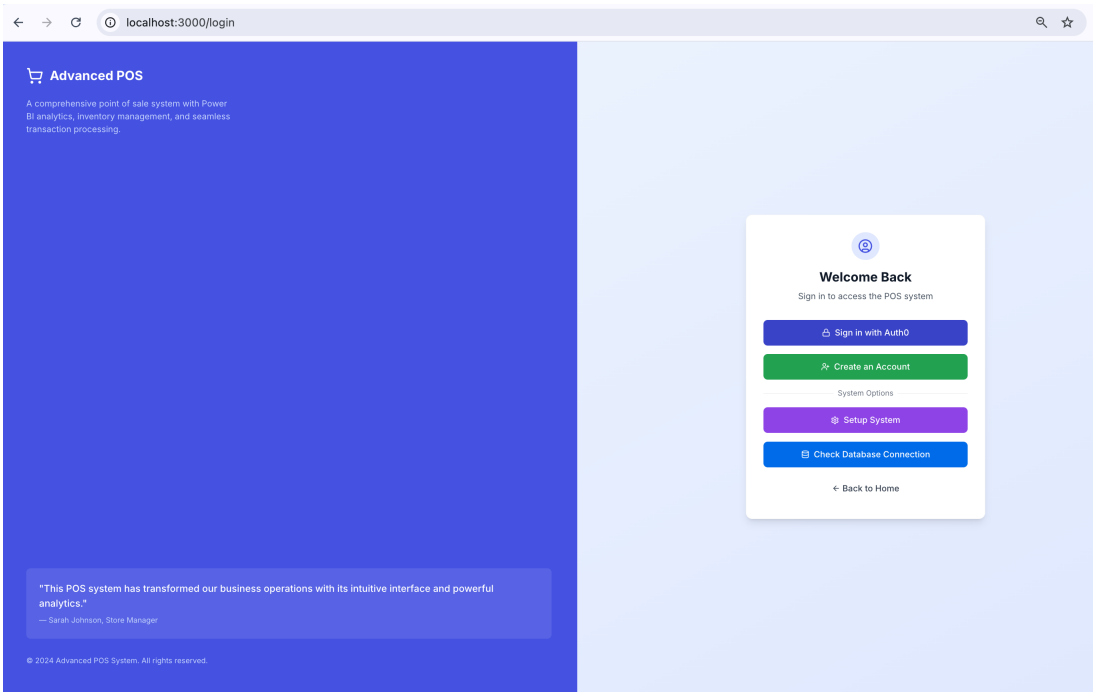


Figure 5: Login Interface – Auth0 Integration

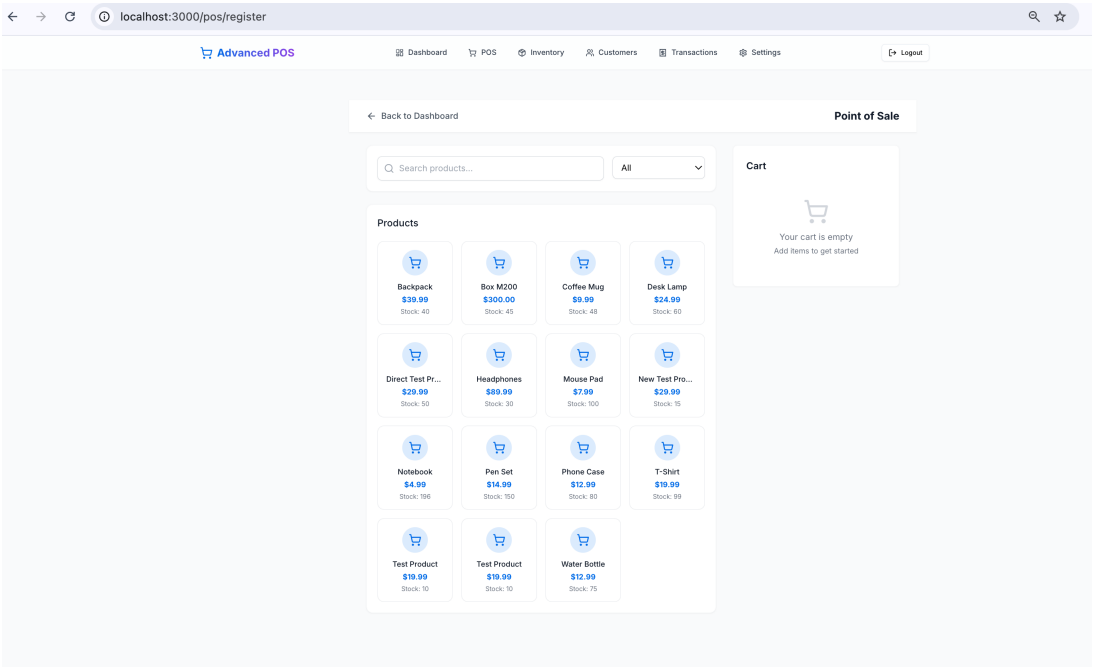


Figure 6: Cashier POS Interface – Product Cart and Grid

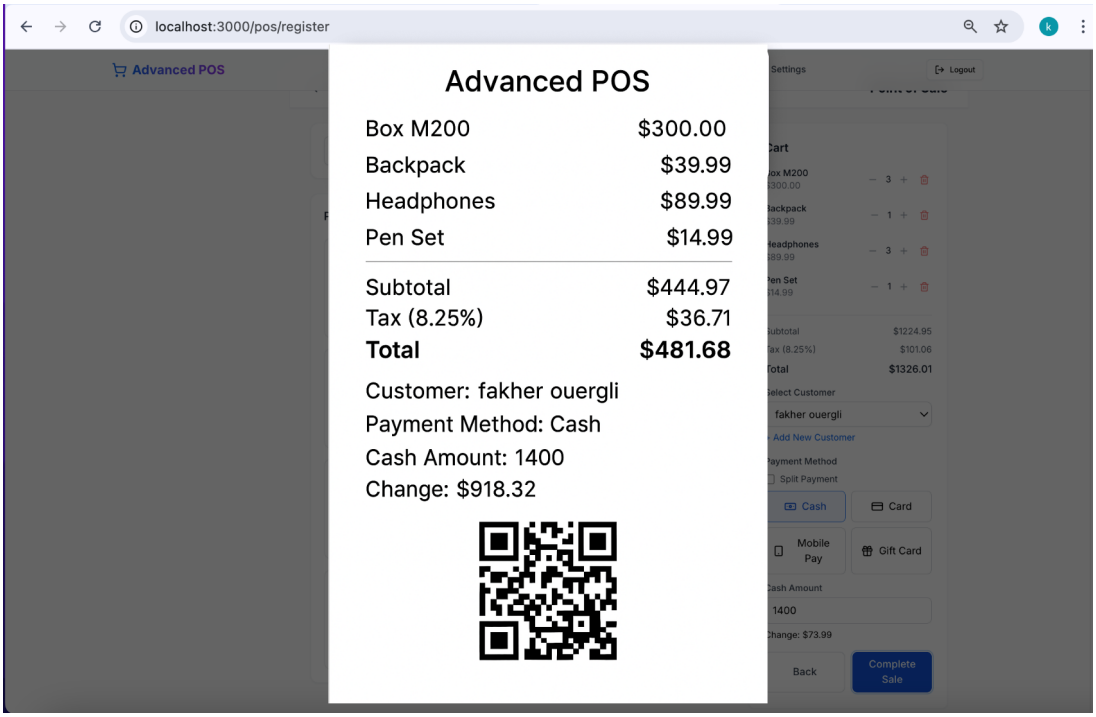


Figure 7: Generated Invoice with QR Code

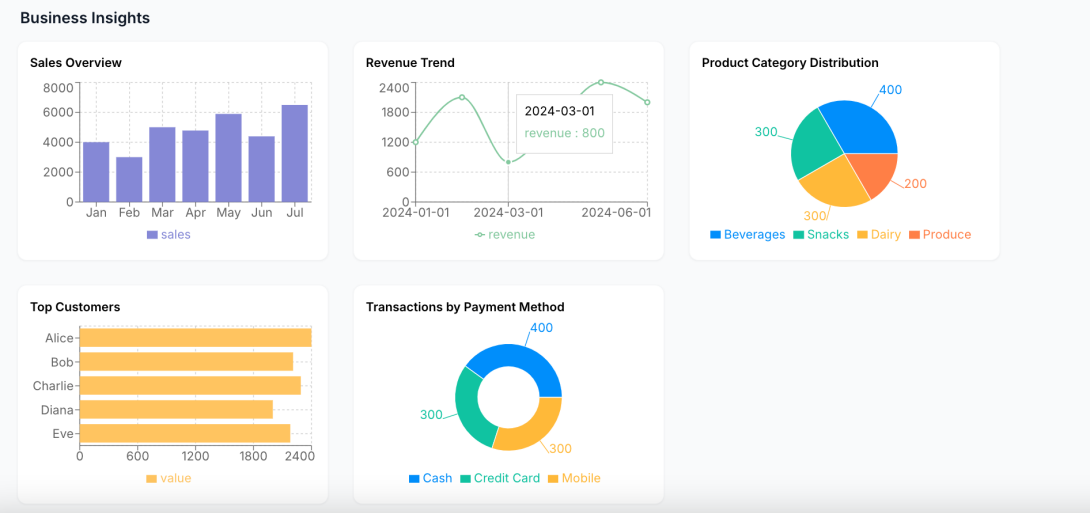


Figure 8: Embedded Power BI Dashboard in Admin Panel

Annex B – Environment Configuration (.env)

```
AUTH0_SECRET=***************
AUTH0_BASE_URL=http://localhost:3000
AUTH0_ISSUER_BASE_URL=https://dev-xxxx.eu.auth0.com
AUTH0_CLIENT_ID=**************
AUTH0_CLIENT_SECRET=**************
DATABASE_URL=postgres://postgres:postgres@localhost:5432/pos_db

NEXT_PUBLIC_POWERBI_WORKSPACE_ID=xxxxxxxxxxxxxxxxxxxx
NEXT_PUBLIC_POWERBI_ACCESS_TOKEN=xxxxxxxxxxxxxxxxxxxx
NEXT_PUBLIC_POWER_BI_REPORT_EMBED_URL=https://app.powerbi.com/view?r=...

NEXT_PUBLIC_APP_URL=http://localhost:3000
```

Annex C – Project Folder Structure

```
src/
 app/
    admin/
    pos/
    api/
 components/
    ui/
```

```
   shared/
lib/
   auth/
   db/
public/
   figures/
      logos/
      screenshots/
styles/
utils/
```

Annex D – Testing Evidence

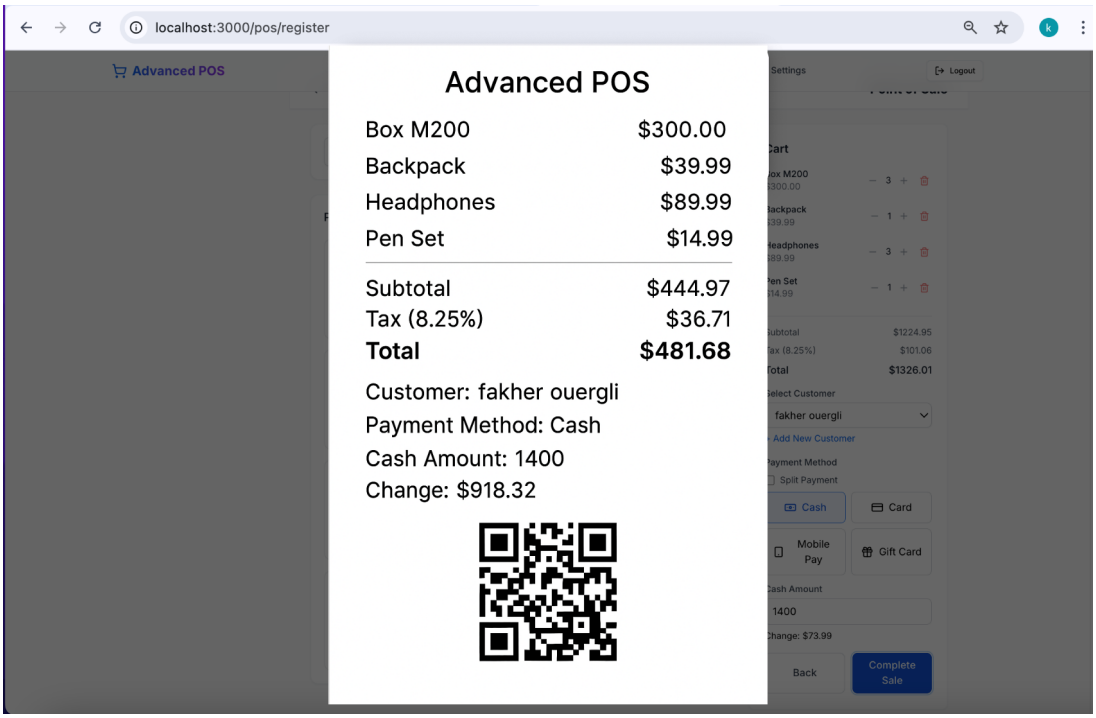


Figure 9: Test Passed – Successful Invoice and Stock Update

Annex E – Additional Admin Screens

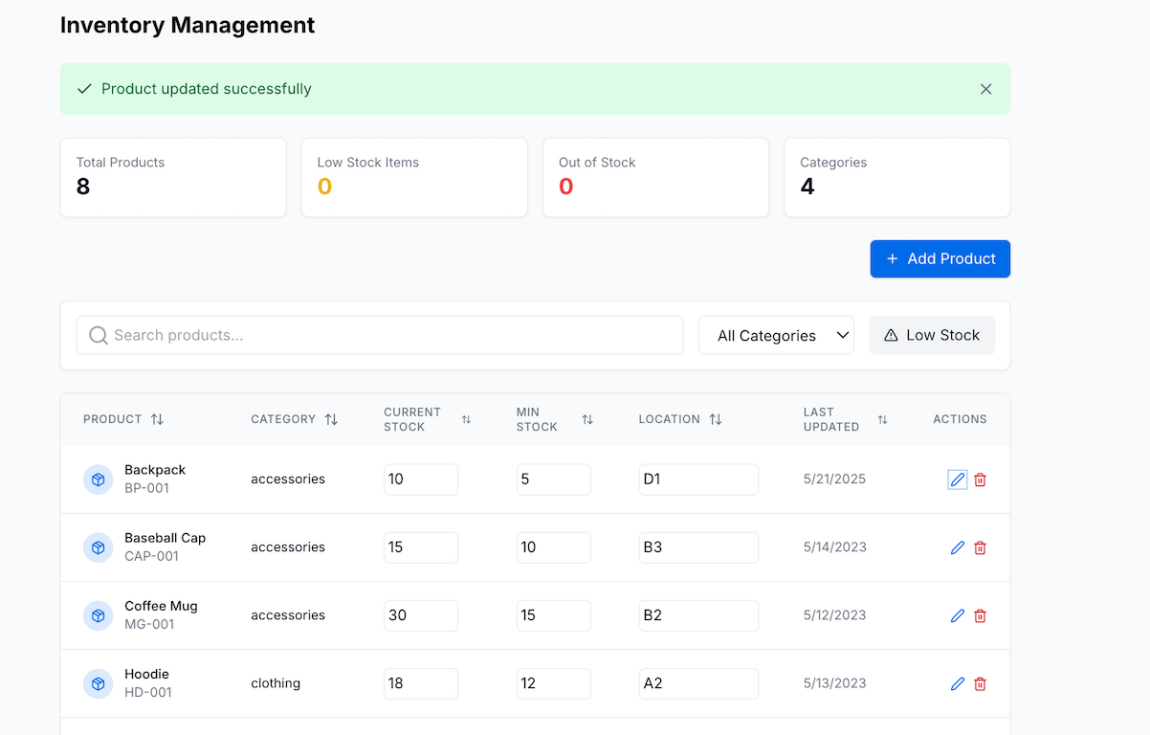


Figure 10: Admin Panel – Product CRUD Interface

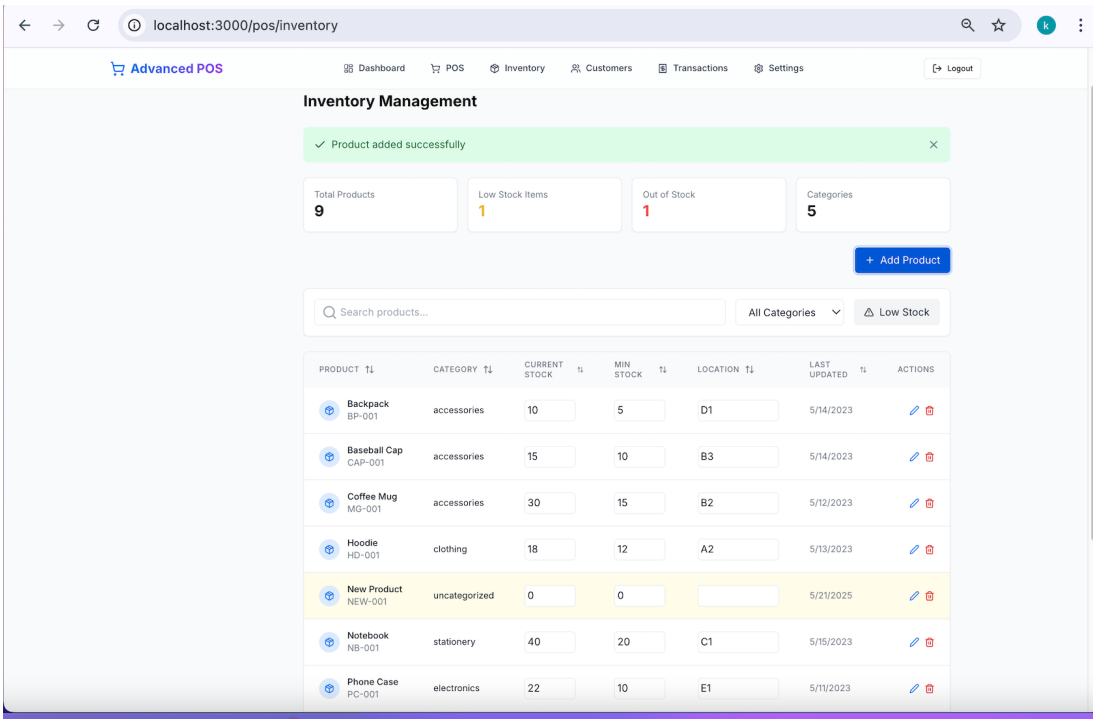


Figure 11: Stock Management – Update Quantity in Admin View

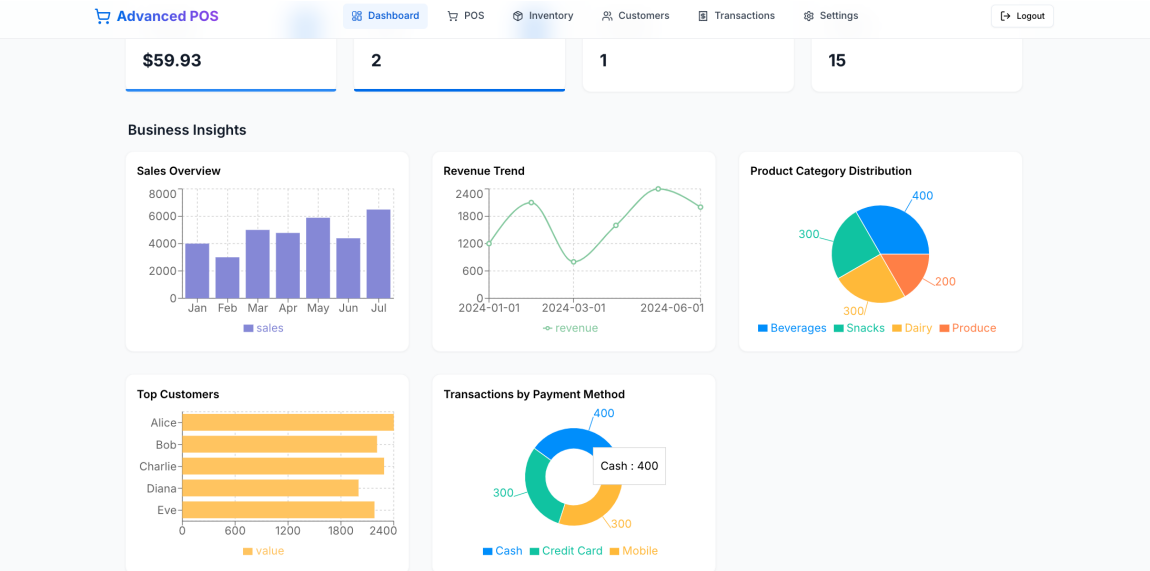


Figure 12: Power BI Admin Dashboard – Filtered Sales Visualization

Annex F – Validation Summary

- Role-based routing was tested for both admin and cashier flows.
- Invoice QR codes were verified with third-party readers.
- Power BI dashboard was validated using live database data.