

**UNIVERSITY OF NAIROBI**

**DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF SCIENCE AND TECHNOLOGY**

INVEX INVENTORY MANAGEMENT SYSTEM

SCS3/2265/2023

MR. ERIC MASIGA AYIENGA

**A Project Report Submitted in Partial** Fulfillment **of the Requirements for the Award of Bachelor of Science in Computer Science of the University of Nairobi**

December **202**4

# **DECLARATION**

I, Kitavi Justus Wambua (SCS3/2265/2023), hereby confirm that this programming project report, titled Invex Inventory Management System, is my own original work and has not been previously submitted for any academic qualification at this or any other institution.

This project has been carried out as part of the requirements for the Bachelor of Science in Computer Science degree at the University of Nairobi, under the guidance of Mr. Eric Masiga Ayienga. Any content derived from the work of others has been properly cited and referenced accordingly.

I acknowledge that acts of plagiarism and dishonesty are serious academic offenses and I certify that this work adheres to the ethical standards and academic integrity policies of the university.

**Kitavi Justus Wambua: \_\_\_\_\_\_\_\_\_\_\_\_  
Date: \_\_\_\_\_\_\_  
SCS3/2265/2023**

# ACKNOWLEDGEMENT

**I would like to extend my heartfelt appreciation to my supervisor, Dr. Samuel N. Ruhiu, for his invaluable guidance, support, and encouragement throughout the development of the Invex Inventory Management System. His insightful feedback and expertise have played a crucial role in shaping the quality and direction of this project.**

**I am also sincerely grateful to the University of Nairobi and the Department of Computing and Informatics for providing the necessary resources and a supportive learning environment that enabled me to successfully complete this project.**

**My gratitude also goes to everyone who contributed in any way towards the successful realization of this work.**

**Kitavi Justus Wambua  
SCS3/2265/2023**

# ABSTRACT

The Inventory Management System (INVEX) is a web-based application developed to enable small scale shop owners to manage their shops and products efficiently. Through Invex, it will be possible to run a small-scale shop, even on absentia. The owner’s job will simply be reduced to ensuring the shops never run out of stock.

The system automates tasks such as product stock tracking by sending emails notifications to the owner when the quantity of a given product reaches a given threshold, which will be set by the owner, in order to start working on restocking. On depletion of a given product, the owner is also notified for appropriate action from his side. The software also allows selling and restocking, employee creation as well as shop reports where the owner is presented with charts the overall revenue from each product as well as each product revenue, all for a specified time duration.

This project adopts Django as the backend framework and PostgreSQL for the database, using HTML, CSS, and JavaScript on the frontend. This report provides a comprehensive breakdown of system analysis, design, implementation, and the challenges encountered.

TABLE OF CONTENTS

[DECLARATION 2](#_Toc199315759)

[ACKNOWLEDGEMENT 3](#_Toc199315760)

[ABSTRACT 4](#_Toc199315761)

[LIST OF TABLES 8](#_Toc199315762)

[ABBREVIATIONS 9](#_Toc199315763)

[CHAPTER 1: INTRODUCTION 10](#_Toc199315764)

[1.1. Background 10](#_Toc199315765)

[1.2. Problem Statement 11](#_Toc199315766)

[1.3. Objectives 12](#_Toc199315767)

[1.4. Scope 13](#_Toc199315768)

[1.5. Significance 14](#_Toc199315769)

[CHAPTER 2: REVIEW OF SIMILAR SYSTEMS 15](#_Toc199315770)

[2.1. Odoo Inventory 15](#_Toc199315771)

[2.2. Zoho Inventory 15](#_Toc199315772)

[2.3. InFlow Inventory 15](#_Toc199315773)

[2.4. Local POS Systems 16](#_Toc199315774)

[2.5. The Proposed System 16](#_Toc199315775)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 17](#_Toc199315776)

[3.1. SYSTEMS ANALYSIS AND DESIGN METHODOLOGY 17](#_Toc199315777)

[3.2. SYSTEM ANALYSIS 18](#_Toc199315778)

[3.3. SYSTEM DESIGN 25](#_Toc199315779)

[3.3.1. Architectural Design 25](#_Toc199315780)

[3.3.2. Processes Design 26](#_Toc199315781)

[3.3.3. Database Design 27](#_Toc199315782)

[3.3.4. Data Flow Diagrams (DFDs) 28](#_Toc199315783)

[3.3.5. User Interface Design 31](#_Toc199315784)

[3.3.6. Technology Design 36](#_Toc199315785)

[CHAPTER 4: SYSTEM IMPLEMENTATION 37](#_Toc199315786)

[**4.1.** **Technology Stack** 37](#_Toc199315787)

[Frontend Technology 37](#_Toc199315788)

[HTML 37](#_Toc199315789)

[Bootstrap CSS 38](#_Toc199315790)

[JavaScript 38](#_Toc199315791)

[Backend Technologies 39](#_Toc199315792)

[Database 40](#_Toc199315793)

[**4.2.** **Testing** 41](#_Toc199315794)

[1. User Registration 41](#_Toc199315795)

[2. User Login 42](#_Toc199315796)

[3. Creating a Shop 43](#_Toc199315797)

[4. Creating a Product 43](#_Toc199315798)

[5. Selling a Product 44](#_Toc199315799)

[6. Adding Stock 45](#_Toc199315800)

[7. Creating an Employee 45](#_Toc199315801)

[CHAPTER 5: CONCLUSION AND RECOMMENDATION 47](#_Toc199315802)

[**Conclusion** 47](#_Toc199315803)

[**Recommendations** 47](#_Toc199315804)

[REFERENCES 48](#_Toc199315805)

[APPENDICES 1](#_Toc199315806)

[Appendix 1: User Manual 1](#_Toc199315807)

LIST OF FIGURES

 Fig 1: Agile methodology

 Fig 2: Architecture design

 Fig 3: Process design

 Fig 4: Entity relationship diagram

 Fig 5: Context diagram

 Fig 6: Level 1 DFD

 Fig 7: Use case diagrams

 Fig 8: Registration page

 Fig 9: Login page

 10: Home page

 Fig 11: Shops page

 Fig 12: Products Page

 Fig 13: Employees Page

 Fig 14: Profile page

 Fig 15: Sales Report page

 Fig 16: Technology design

# LIST OF TABLES

 Table 1: User registration test case

 Table 2: Login test case

 Table 3: Creating Shop test case

 Table 4: Creating product test case

 Table 5: Selling test case

 Table 6: Adding stock test case

 Table 7: Creating an employee test case

# ABBREVIATIONS

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form** |
| IMS | Inventory Management System |
| OTP | One-Time Password |
| ERP | Enterprise Resource Planning |
| CRM | Customer Relationship Management |
| UI | User Interface |
| POS | Point of Sale |
| DFDs | Data Flow Diagrams |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| JS | JavaScript |
| ORM | Object-Relational Mapping |
| SQL | Structured Query Language |
| XSS | Cross-Site Scripting |
| CSRF | Cross-Site Request Forgery |
| MVC | Model-View-Controller |
| IMS | Inventory Management System (duplicate) |
| DoB | Date of Birth |
|  |  |

# CHAPTER 1: INTRODUCTION

This chapter covers the background, problem statement, objectives, scope, and the significance of the project.

## Background

Inventory management plays a crucial role in the smooth operation of businesses, ensuring that stock levels are accurately monitored to prevent shortages or overstocking. Traditionally, businesses have relied on manual record-keeping methods such as spreadsheets and paper-based logs, which are prone to errors, inefficiencies, data loss, and also require special skilled labor to manage, which is often expensive for small businesses.

With advancements in technology, businesses are shifting towards automated inventory management systems that enhance accuracy, efficiency, and real-time tracking of stock. The ****Invex Inventory Management System**** is designed to provide an intuitive and efficient solution for tracking inventory, automating stock updates, generating reports, and streamlining the ordering process. By leveraging ****a centralized database, user authentication, and reporting features****, the system aims to minimize human errors and cost of labor and optimize business operations.

## Problem Statement

Many businesses, especially small and medium enterprises, struggle with inefficient inventory management due to manual record-keeping, inability to hire skilled labor and use of outdared systems. These challenges often result in:

* Stock inconsistencies due to inaccurate data entry.
* Loss of inventory records due to lack of a centralized storage system.
* Time-consuming stock tracking processes leading to delays in order fulfillment.
* Poor decision-making caused by the unavailability of real-time reports.

The Invex Inventory Management System seeks to address these challenges by providing an automated, easy-to-use, and efficient inventory management solution that ensures accuracy, improves stock control, and enhances business productivity while still utilizing the available cheap unskilled labor.

## Objectives

* + 1. Automate inventory tracking to reduce human errors and improve accuracy.
    2. Provide real-time stock updates to prevent overstocking and stockouts.
    3. Generate detailed reports to support decision-making and business analysis.
    4. Implement user authentication and role-based access to ensure data security and integrity.

## Scope

The Invex Inventory Management System will be designed for small to medium-sized businesses that require a centralized, efficient, and easy-to-use inventory management solution. The system will include:

* Stock management module for adding, updating, and deleting inventory records.
* Real-time notifications for low stock levels and order alerts.
* User authentication and role management to control access levels.
* Reporting and analytics module for generating inventory insights.
* Database storage for secure and efficient data management.

However, the system will not include advanced enterprise resource planning (ERP) features such as financial accounting, supplier management, or complex warehouse logistics.

## Significance

The Invex Inventory Management System will offer significant benefits to businesses by:

* Reducing human errors associated with manual record-keeping.
* Enhancing decision-making through real-time data and reports.
* Saving time and improving efficiency in stock management operations.
* Improving security by restricting access to inventory data based on user roles.
* Increasing profitability by preventing losses due to mismanaged stock.
* Enable entrepreneurs to own and run several branches in different locations successfully.

By implementing this system, businesses will experience improved operational efficiency, reduced costs, and enhanced customer satisfaction, making it a valuable investment in modern inventory management.

# CHAPTER 2: REVIEW OF SIMILAR SYSTEMS

This chapter covers the literature reviewed on appropriate technologies and similar systems.

## Odoo Inventory

**Odoo is an all-in-one business management software that offers a suite of integrated applications, including CRM, e-commerce, accounting, manufacturing, project management, and, crucially, inventory management. Its inventory module is designed to handle complex warehouse operations, including multi-warehouse management, drop-shipping, cross-docking, and advanced routing.**

**However, with all the impressive and desirable features, the system is quite complex for small to medium-sized businesses, and one with quite a steep learning curve in order to optimally use all the features available in the system. This leaves a niche, to be occupied by systems like Invex, which are simpler and light weight.**

## Zoho Inventory

**Zoho Inventory is a cloud-based inventory management software that helps businesses manage their sales and purchases, track inventory, and fulfill orders. It is part of the larger Zoho ecosystem, which includes CRM, accounting, and other business applications, allowing for extensive integrations. It's particularly popular among small and growing businesses due to its accessibility and relatively affordable pricing plans. While integrations are a strength, businesses heavily reliant on non-Zoho systems might find integrating with their existing tools challenging or less seamless.**

## InFlow Inventory

**InFlow Inventory is a desktop-based and cloud-based inventory software designed to help small to mid-sized businesses manage their stock, sales, and purchases. It offers features like barcode scanning, order management, reporting, and multi-location tracking. It is known for its user-friendly interface and robust reporting capabilities. The system lacks multi-shop support which may lead to operational inefficiencies.**

## Local POS Systems

**Local Point of Sale (POS) systems in Kenya typically refer to on-premise software solutions installed directly on a computer or server within a single business location. These systems often handle sales transactions, basic inventory tracking, and sometimes customer management. They can range from simple cash register software to more comprehensive retail management systems.**

**However, most of them lack internet access capability, which limit the ability to provide real time tracking of the stock quantities. The system is vulnerable to data breaches, due to poor security protocols. Scalability is also an issue for these systems.**

## The Proposed System

While the aforementioned systems are highly capable, they are primarily designed for large-scale operations and come at a high cost. This puts small to medium-sized businesses at a disadvantage. Invex aims to bridge this gap by providing an affordable and efficient inventory management solution tailored to the needs of small and medium enterprises.It is tailored for the local market, prioritizing simplicity, low cost, ease of use, and features critical to retail operations such as secure stock editing and real time reporting.

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

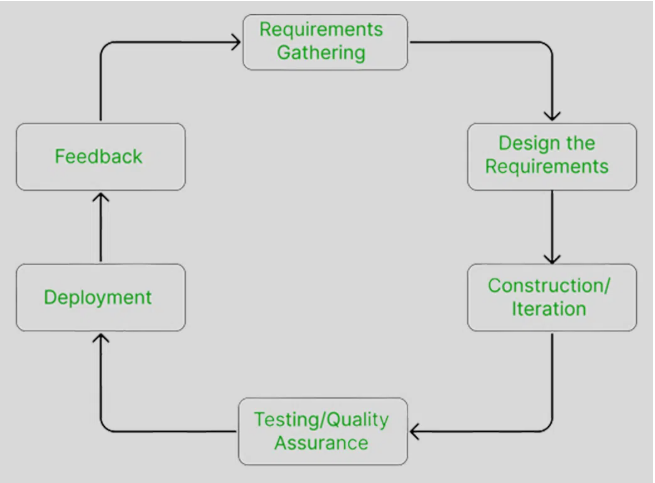
This chapter details the methodology, analysis and design used in this project to develop the target system.

## SYSTEMS DESIGN METHODOLOGY

I plan to adopt the Agile methodology for system development.

Justification:

* Iterative development and adaptability – Enable modifications to be incorporated seamlessly without affecting the entire project.
* Early and continuous delivery – Allows for early testing of a functional system, ensuring that key features like stock tracking, reporting, and notifications are implemented efficiently.
* Scalability and maintainability – Support future expansion and the integration of additional functionalities such as supplier management or AI-powered analytics.

*fig 1: Agile methodology*

## SYSTEM ANALYSIS

* + 1. Requirements Analysis
  1. Functional Requirements

**a. User Authentication and Authorization**

1. The system shall allow new users to register using valid personal details and verify their identity via OTP.
2. The system shall allow registered users to log in using their credentials.
3. The system shall allow authenticated users to log out securely.
4. The system shall restrict access to sensitive views (e.g., shop dashboard, product editing) based on password verification.
5. The system shall support password reset and change functionalities.

**b. Shop Management**

1. The system shall allow users to create new shops with unique identifiers.
2. The system shall allow users to view a list of all their created shops.
3. The system shall display full details for each shop, including name, type, industry, and location.
4. The system shall allow users to delete (shut down) a shop upon confirmation and password authentication.

**c. Product Management**

1. The system shall allow users to add new products to a shop, capturing product name, description, manufacturer, price, quantity, and threshold.
2. The system shall allow users to edit existing product details after password authentication.
3. The system shall allow users to delete products from the shop after confirming their password.
4. The system shall display all products associated with a shop in a tabular format.

**d. Stock Control**

1. The system shall allow users to increase stock quantity of a product through a dedicated form.
2. The system shall allow users to reduce (sell) stock of a product.
3. The system shall prevent stock transactions (add or sell) without password verification.
4. The system shall track and update product quantity after every stock adjustment.

**e. Sales Management**

1. The system shall record every sale transaction with a timestamp, product reference, quantity, and total price.
2. The system shall automatically calculate total revenue per sale.
3. The system shall maintain a history of sales for each product for reporting purposes.

**f. Reporting and Analytics**

1. The system shall provide general sales summaries per shop, displayed as bar charts using Chart.js.
2. The system shall allow users to view revenue trends for a specific product over the last 12 months duration.
3. The system shall visualize product sales as line graphs for analysis.
4. The system shall enable comparison of sales performance among products.

**g. Role Management**

1. The system should support creation of shop employees with limited privileges.

**h. Responsiveness and UI Behavior**

1. The system shall use modals for all password verification and form inputs to prevent page reloads.
2. The system shall alert users upon successful or failed operations.
3. The system shall dynamically load and inject content using JavaScript to provide a smooth experience.
   1. Non-functional requirements
4. The system shall respond to user actions (e.g., clicking buttons, submitting forms) within 2 seconds under normal network conditions.
5. The system shall load shop dashboards and product lists within 3 seconds for up to 1000 product entries.
6. AJAX-based form interactions (stock updates, product edits) shall be processed and rendered without reloading the page.
7. The system shall be able to support multiple shops and users concurrently without performance degradation.
8. The database schema shall support extensibility to accommodate more shops, users, and employee records.
9. The system shall support future integrations such as multi-store synchronization, barcode scanning, and external API data feeds.
10. The system shall use **Django’s CSRF protection** to secure all POST requests.
11. Passwords shall be stored securely using hashing algorithms provided by Django.
12. Access to shop-level data shall be restricted to authenticated users only, with further password prompts for sensitive operations (e.g., editing, deleting).
13. The system shall sanitize all user inputs to prevent SQL injection, XSS, and CSRF attacks.
14. OTP verification shall be enforced during registration and password resets to prevent unauthorized access.
15. The system shall provide an intuitive, responsive, and clean user interface using **Bootstrap 5**.
16. Modals shall be used for password entry and confirmation dialogs to maintain context and avoid disruptive navigation.
17. Success and error messages shall be clearly displayed after each operation (add, edit, delete, sell).
18. The system shall offer helpful tooltips and labels for all form fields.
19. The system shall be available 24/7 when deployed to production.
20. User data shall persist in the PostgreSQL database and remain consistent after every transaction.
21. Critical operations like selling and deleting products shall include confirmation dialogs to reduce errors.
22. The codebase shall follow Django’s recommended **MVC (Model-View-Controller)** pattern for clarity and maintainability.
23. JavaScript functions are modular and separated by functionality (shop, stock, sales).
24. CSS is externalized and follows class naming conventions for maintainability.
25. The system shall support future features like:
    1. Employee management (limited roles),
    2. Inventory alerts via email,
26. The system shall run on any modern browser (Chrome, Firefox, Edge, Safari).
27. The system shall be deployable on any server environment supporting Python 3.9+, PostgreSQL, and Django.
    * 1. **Use Cases**
    1. **User Registration**

Actor: New User  
Precondition: User is not already registered.

**Basic Flow:**

1. User accesses the registration page.
2. User fills in required information: name, email, national ID, DoB, gender, phone number, and password.
3. System sends a One-Time Password (OTP) to the user's email.
4. User enters the OTP.
5. System verifies OTP and creates the user account.
6. Registration successful message is displayed.

**Alternate Flows:**

* Invalid Email or National ID: System displays an error message.
* Incorrect OTP: System prompts for re-entry or to resend OTP.
* OTP Timeout: User must request a new OTP.
  1. **User Login**

Actor: Registered User  
Precondition: User account exists and is active.

**Basic Flow:**

1. User enters email and password.
2. System authenticates credentials.
3. User is redirected to their dashboard.

**Alternate Flows:**

* Invalid Credentials: Error message displayed.
* Inactive Account: System prompts to complete verification.
  1. **Creating a Shop**

Actor: Logged-in User  
Precondition: User must be logged in.

**Basic Flow:**

1. User navigates to “Create Shop”.
2. User fills in shop details (name, type, industry, location).
3. System validates input and assigns a unique shopID.
4. Shop is saved to the database and linked to the user.
5. User is redirected to shop detail page.

**Alternate Flows:**

* Duplicate Shop Name: System warns and prompts for a different name.
* Missing Required Fields: Form is not submitted; errors are shown.
  1. **Creating a Product**

Actor: Shop Owner  
Precondition: User has a shop.

**Basic Flow:**

1. User clicks “Add Product” in shop dashboard.
2. User enters product details (name, description, price, quantity, threshold).
3. System validates input and assigns productID.
4. Product is saved to the shop’s inventory.

**Alternate Flows:**

* Missing Fields or Invalid Data: Errors are shown and product is not created.

1. **Selling a Product**

Actor: Shop Owner or Employee  
Precondition: Product exists and quantity > 0.

**Basic Flow:**

1. User clicks “Sell” for a product.
2. User enters quantity to sell and confirms.
3. System checks stock availability.
4. If sufficient, stock is reduced and sale is recorded.
5. Success message shown.

**Alternate Flows:**

* Insufficient Stock: System displays warning.
* Invalid Input (e.g. negative quantity): Sale is not processed.

1. **Adding Stock**

Actor: Shop Owner  
Precondition: Product exists.

**Basic Flow:**

1. User clicks “Add” next to a product.
2. A modal appears asking for quantity and password.
3. User enters valid quantity and password.
4. System verifies credentials.
5. Quantity is added to product inventory.
6. Success message displayed.

**Alternate Flows:**

* Incorrect Password: Authentication fails.
* Invalid Quantity: Validation fails and action is aborted.

**7. Creating an Employee**

Actor: Shop Owner  
Precondition: Shop exists.

**Basic Flow:**

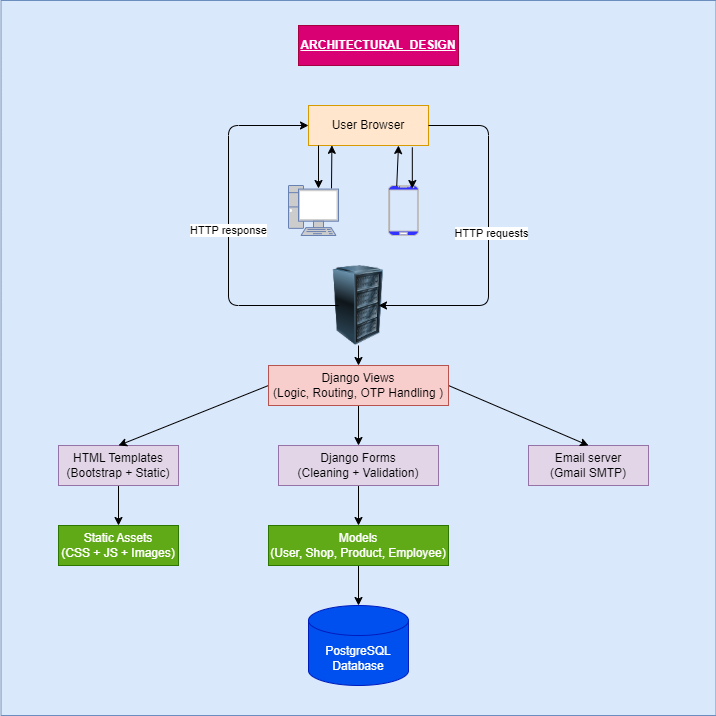
1. User navigates to “Add Employee” page.
2. Fills out employee details (name, DoB, national ID, gender).
3. System validates and assigns employeeID.
4. Employee is added to the shop’s records.

**Alternate Flows:**

* Duplicate National ID: Error displayed; no record saved.
* Missing Fields: Prompt user to complete required fields.

## SYSTEM DESIGN

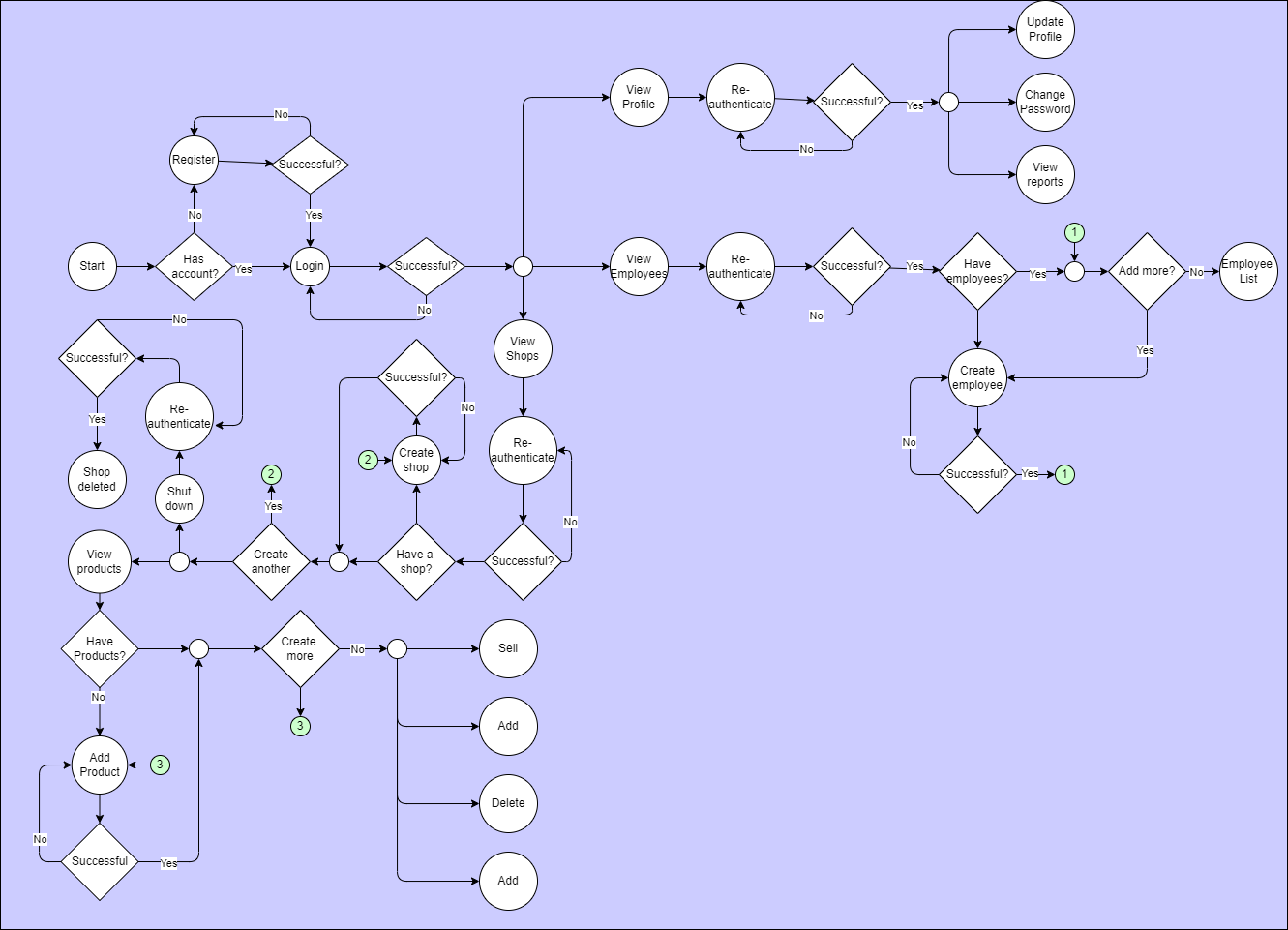
### Architectural Design



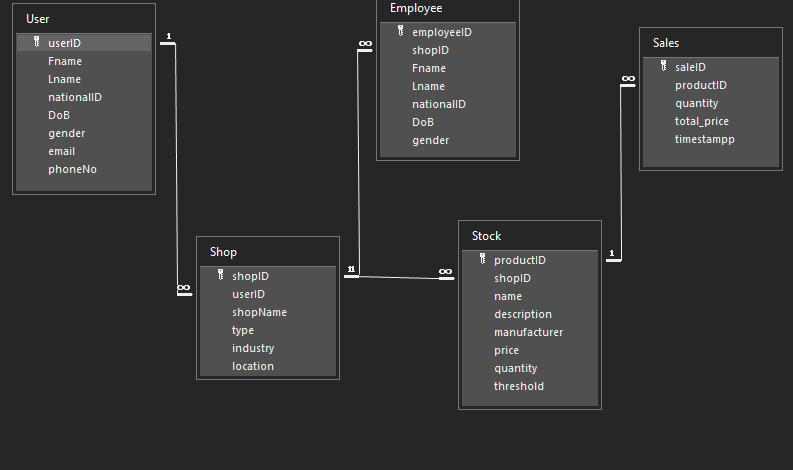
*Fig 2: Architecture design*

### Processes Design

*Fig 3: Process design*



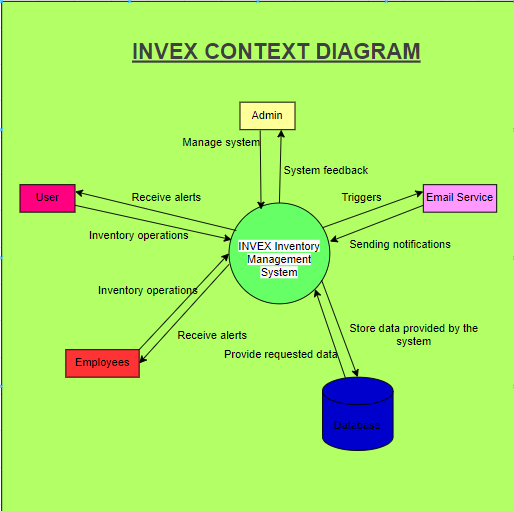
### Database Design



*Fig 4: Entity relationship diagram*

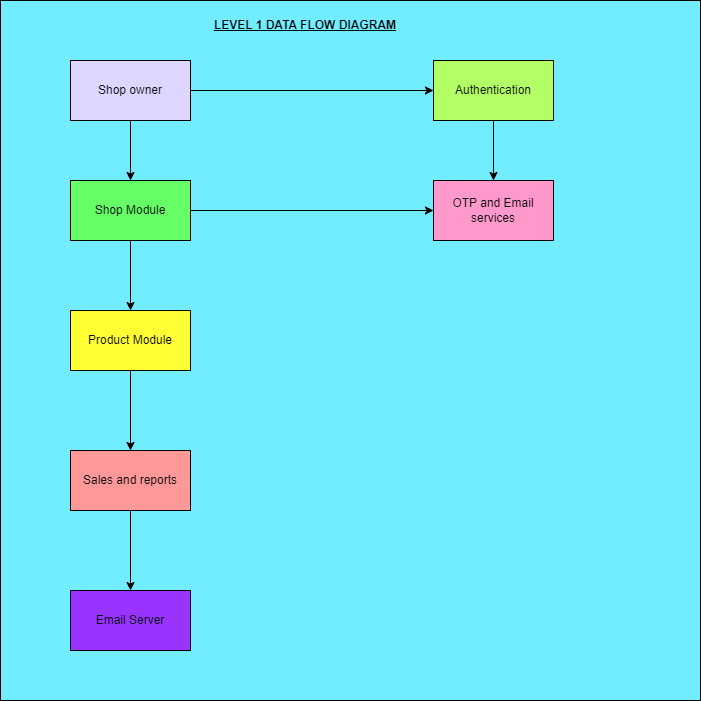
### Data Flow Diagrams (DFDs)

* + - 1. Level 0 DFD (Context Diagram)



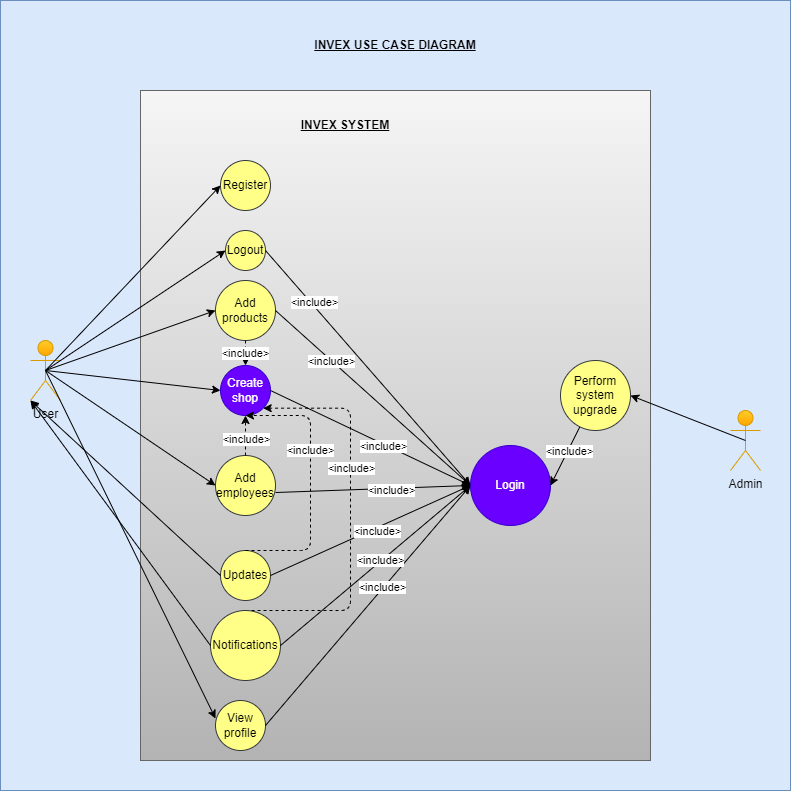
*Fig 5: Context diagram*

* + - 1. **Level 1 DFD**



*Fig 6: Level 1 DFD*

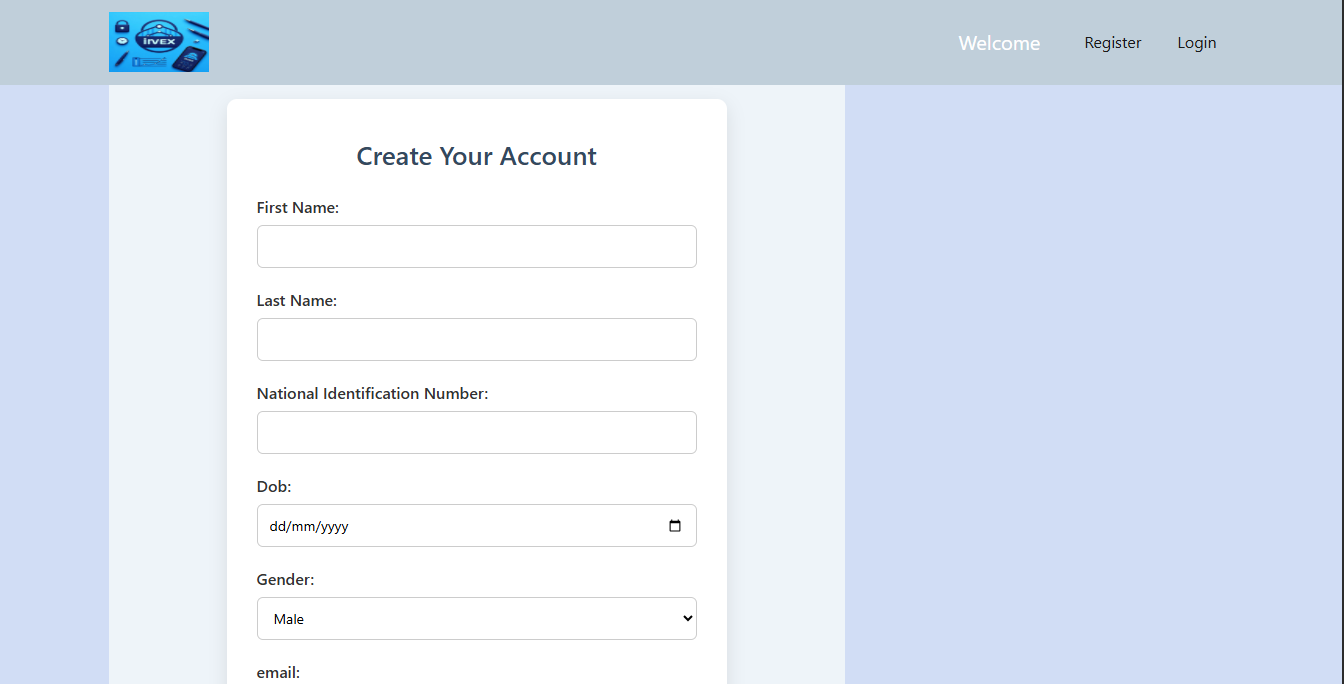
* + - 1. Use Case Diagram

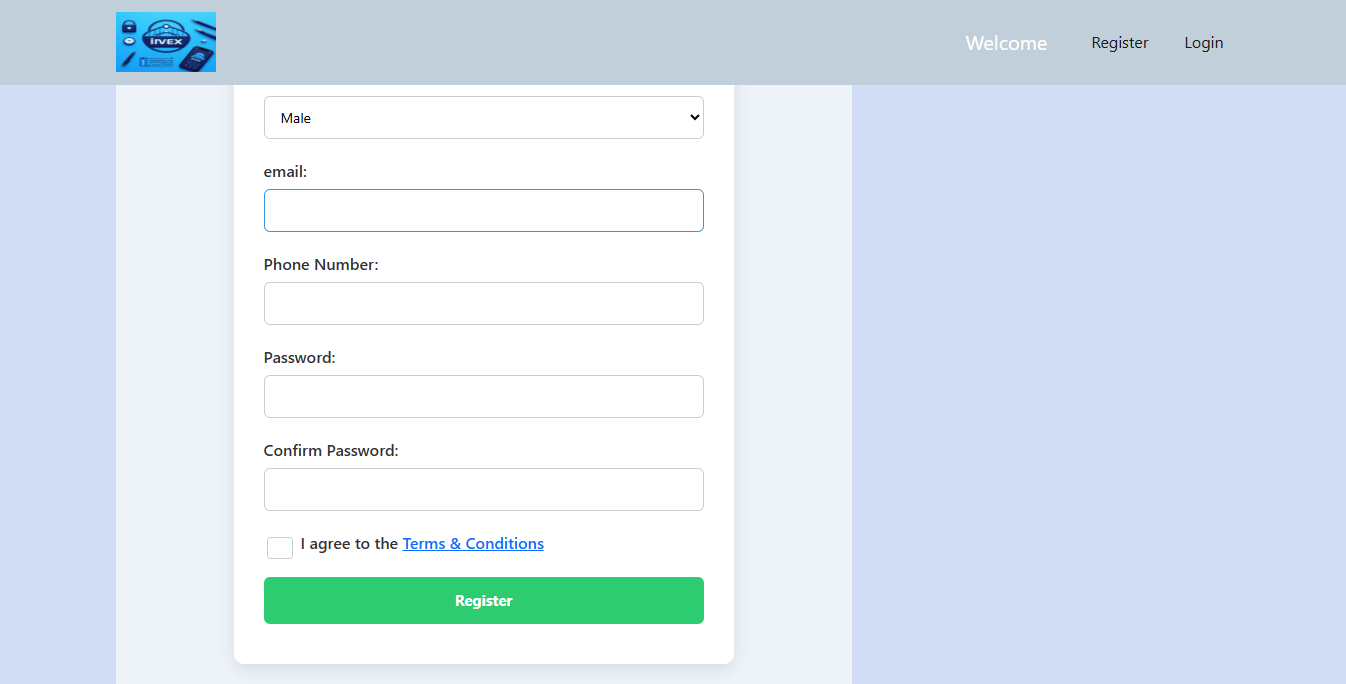


*Fig 7: Use case diagrams*

### User Interface Design

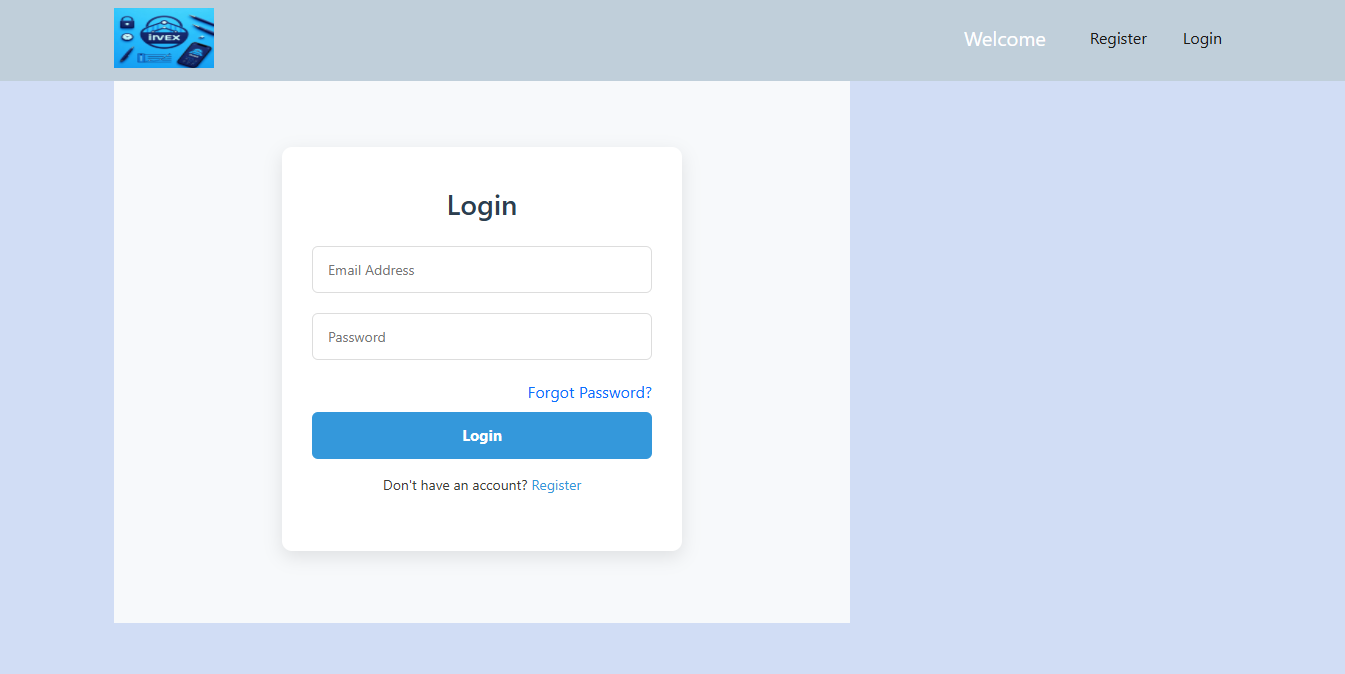
1. **Registration page**

****

****

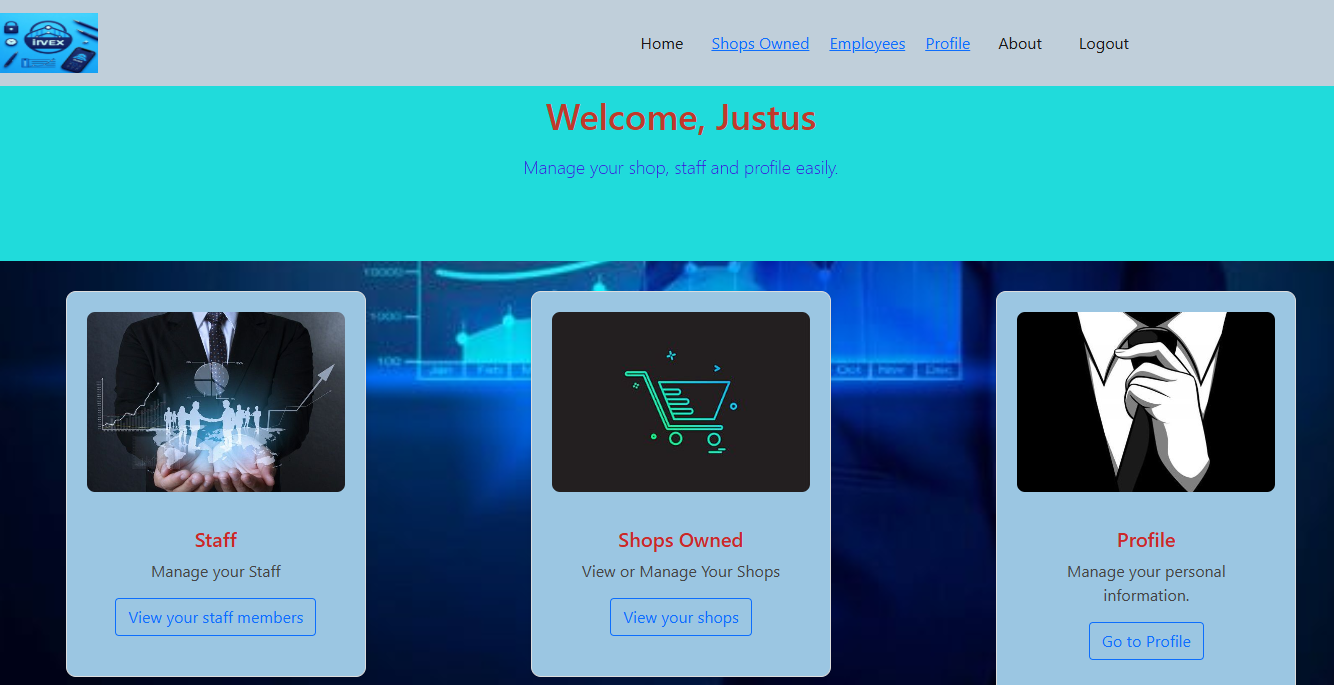
*Fig 8: Registration page*

1. **Login Page**

****

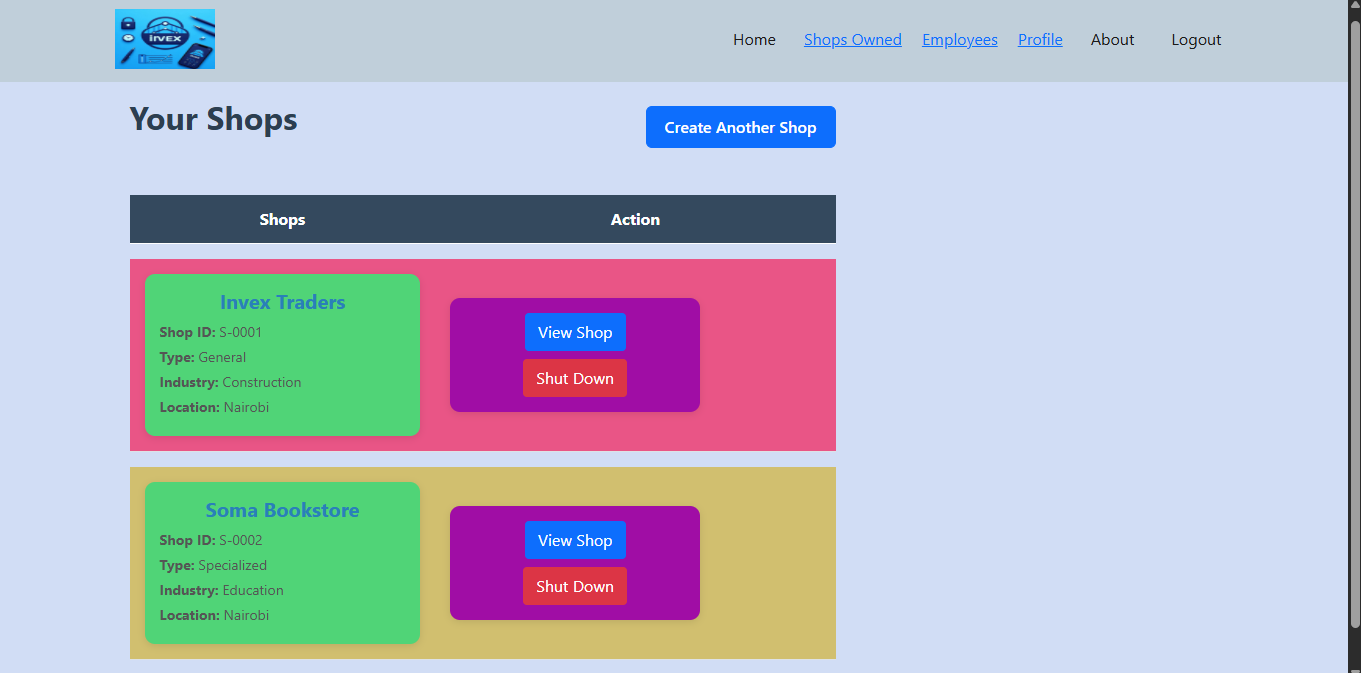
*Fig 9: Login page*

1. **Home page**

****

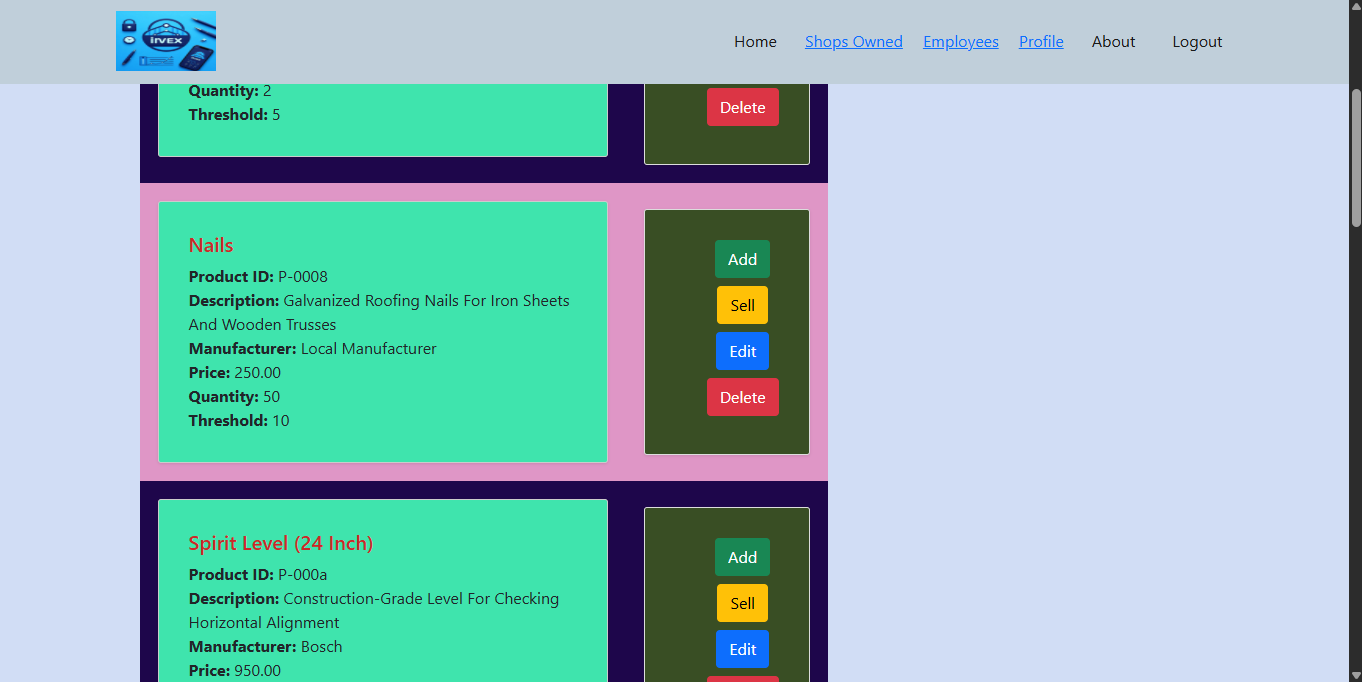
*Fig 10: Home page*

1. **Shops page**

****

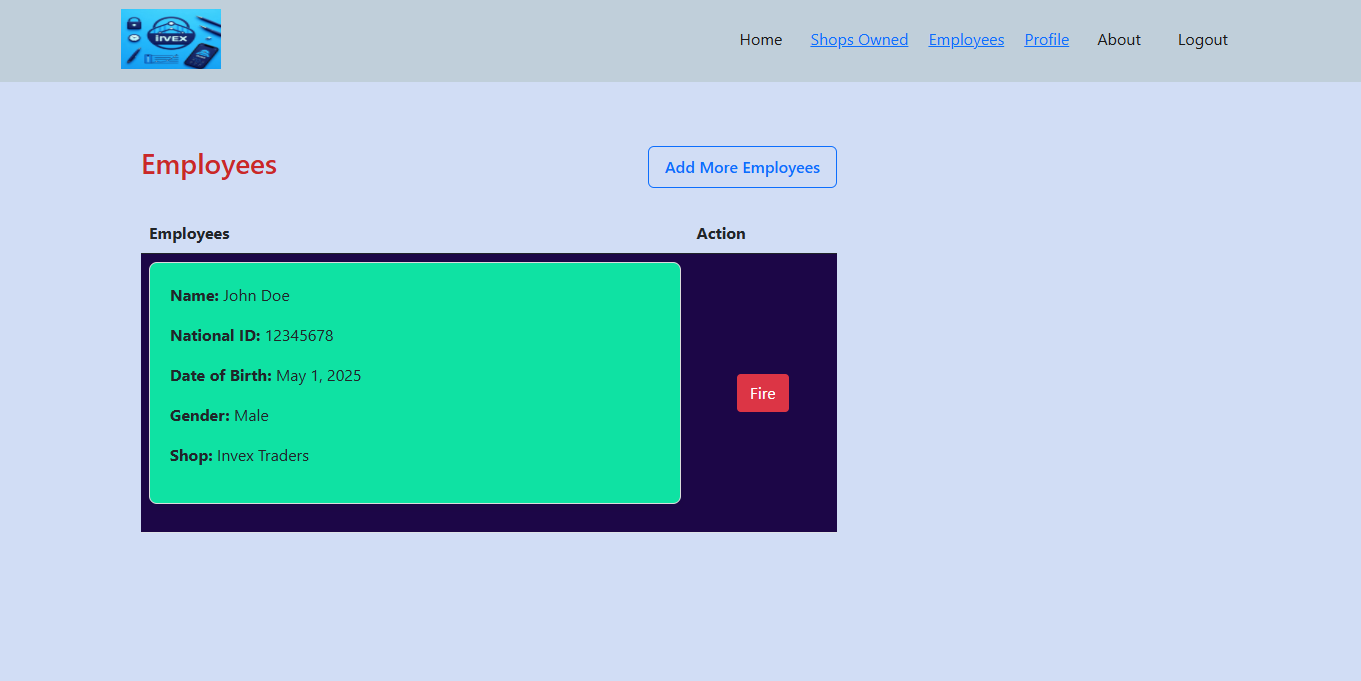
*Fig 11: Shops page*

1. **Products page**

****

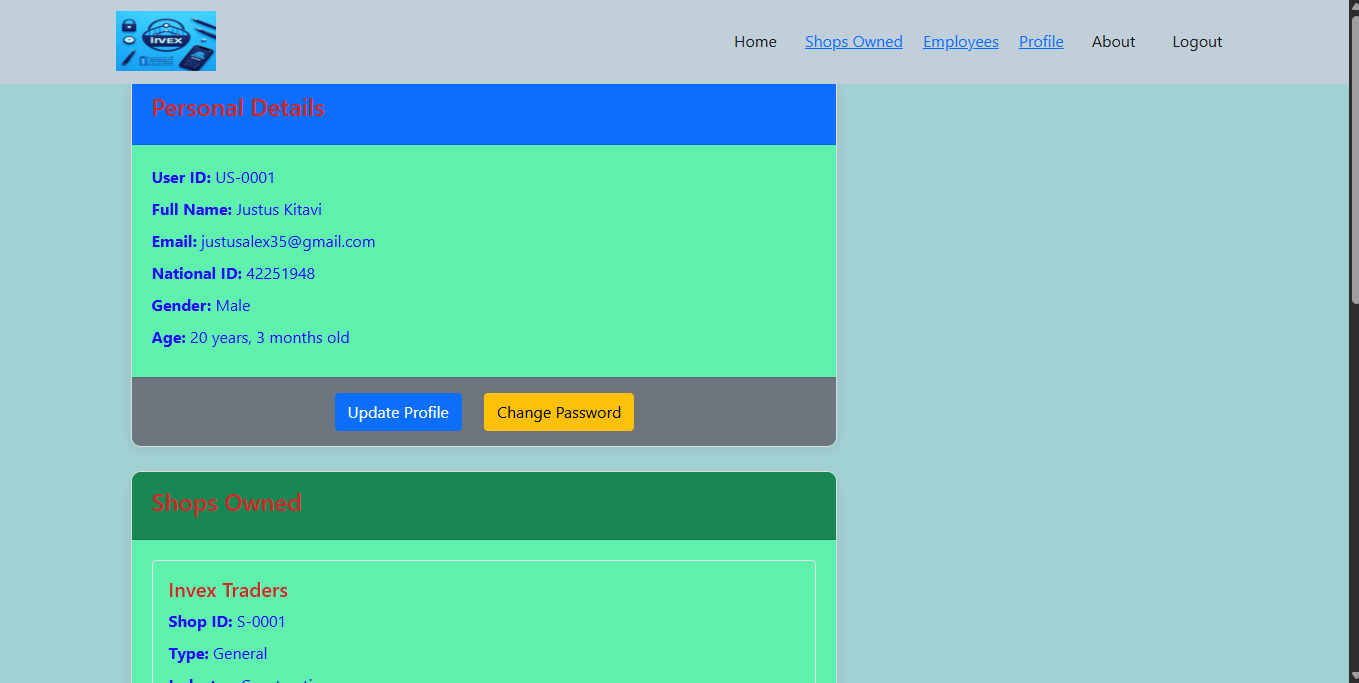
*Fig 12: Products Page*

1. **Employees page**

****

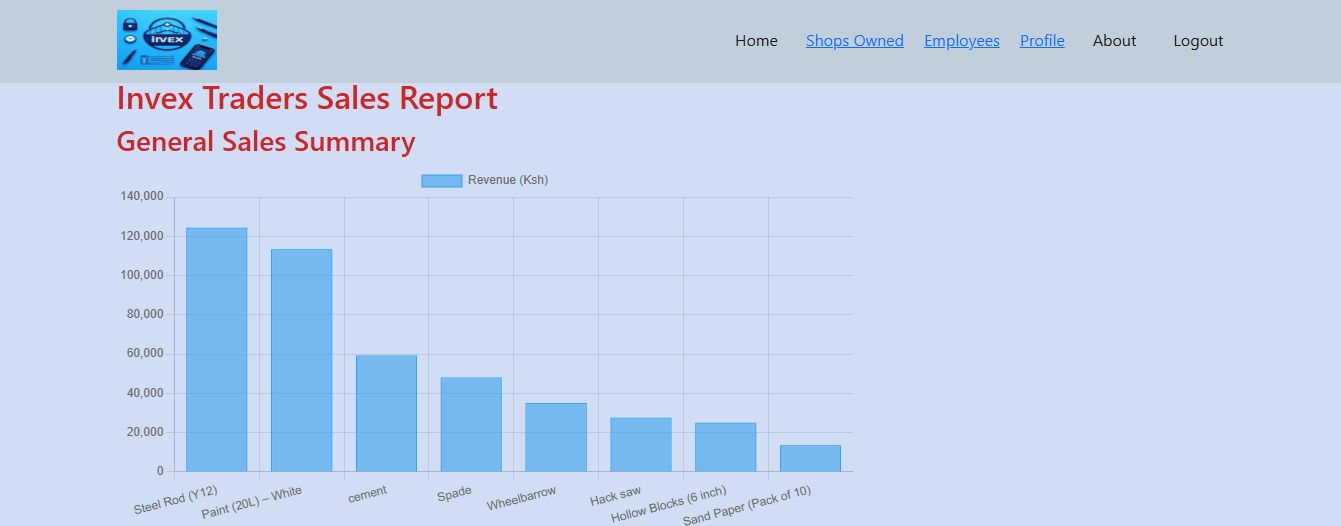
*Fig 13: Employees Page*

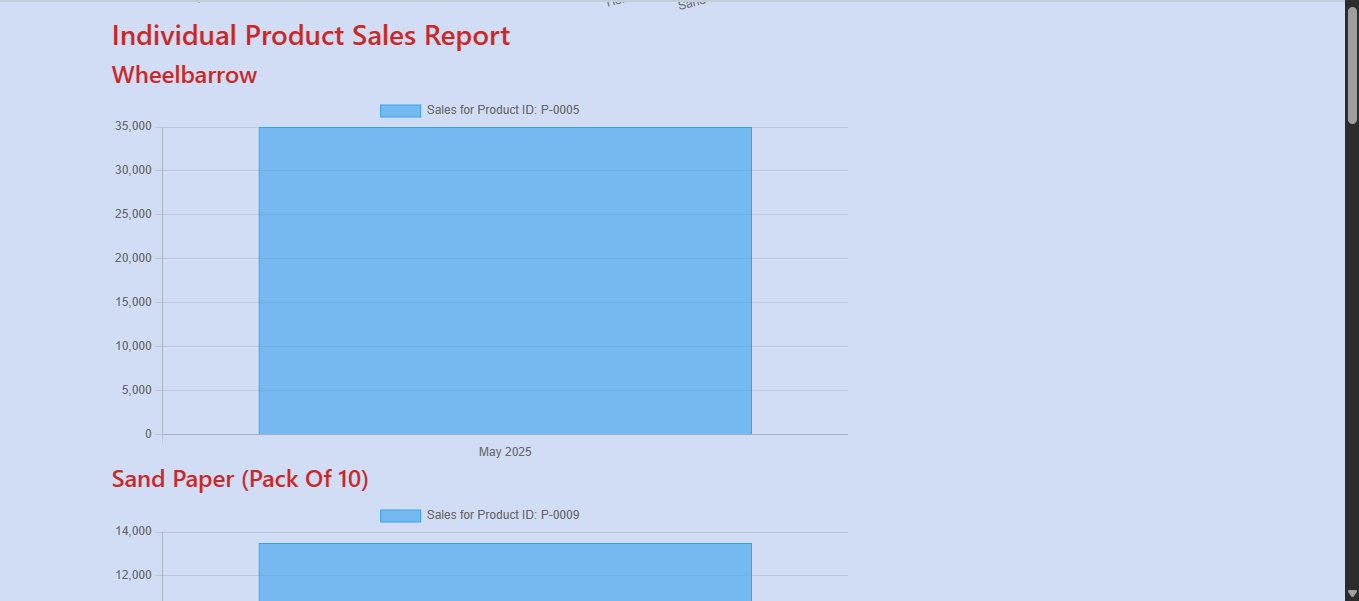
1. **Profile page**

****

*Fig 14: Profile page*

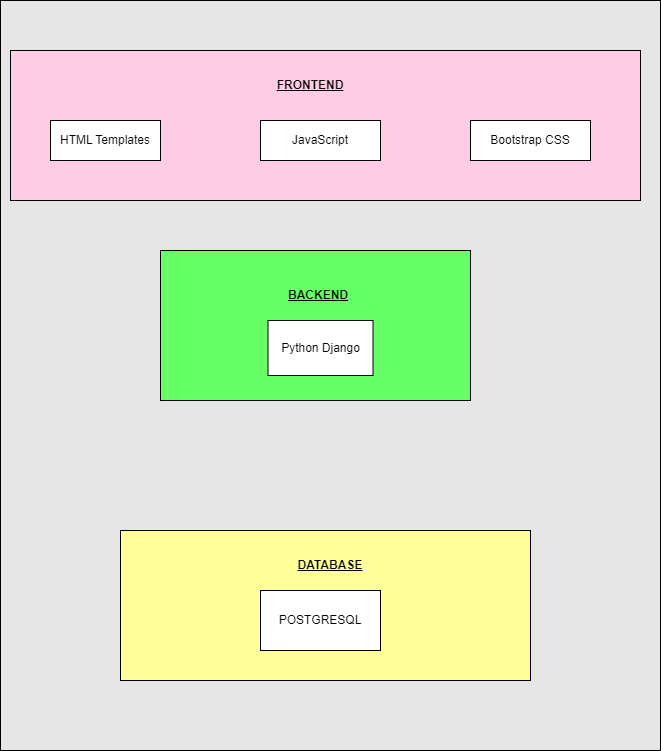
1. **Reports page**

****

****

*Fig 15: Sales Report page*

### Technology Design



*Fig 16: Technology design*

# CHAPTER 4: SYSTEM IMPLEMENTATION

This chapter details how the system was implemented

## **Technology Stack**

### Frontend Technology

Selecting a reliable and effective frontend technology stack was crucial for the development of the Invex IMS system to ensure a responsive, accessible, and user-friendly interface. For the frontend, HTML, Bootstrap CSS, and JavaScript were chosen as the primary tools. This combination was selected due to its simplicity, efficiency, and suitability for small to medium-sized web applications like this project. Below is an overview of each tool and the reasons for their inclusion in the development.

#### HTML

HTML serves as the fundamental layer of any web application. In the case of Invex IMS, an inventory management system, HTML was employed to structure various user interface components, such as inventory entry forms, stock tracking dashboards, order management pages, and user account interfaces.

One notable benefit of using HTML is its semantic richness. By incorporating semantic elements like <nav>, <section>, and <table>, the application's structure becomes more meaningful and easier to maintain. Additionally, semantic HTML enhances search engine optimization and accessibility, making the system more inclusive for users who rely on assistive technologies like screen readers.

HTML is supported across all modern web browsers, ensuring that Invex IMS's content displays consistently across different platforms and devices. A solid understanding of HTML provided a strong foundation for troubleshooting and customizing the system, making it an essential component of frontend development.

#### Bootstrap CSS

To achieve a responsive and visually appealing design, the Bootstrap CSS framework was incorporated into Invex IMS, an inventory management system. Bootstrap provides a comprehensive suite of pre-designed UI components such as buttons, forms, modals, and navigation bars, which greatly accelerate the development process and allow developers to focus on functionality without compromising the user interface's quality.

Bootstrap’s key strength lies in its grid system, enabling mobile-first, responsive design. This ensures that Invex IMS adjusts seamlessly to various screen sizes, offering an optimal user experience across all devices. Additionally, Bootstrap promotes consistency in design, ensuring that all components follow a unified styling language.

Bootstrap also offers a robust collection of utility classes that simplify layout adjustments, spacing, text alignment, and element visibility. These utility classes significantly reduce the need for custom CSS, resulting in cleaner and more manageable code. Many Bootstrap components come with built-in accessibility features such as aria attributes and keyboard navigation, supporting the development of an inclusive web application.

#### JavaScript

To handle the dynamic aspects of Invex IMS, JavaScript was used. This decision was based on the need for lightweight performance, complete control over the application logic, and the opportunity to reinforce core programming concepts.

JavaScript enabled direct interaction with the Document Object Model (DOM), which was crucial for implementing key features such as real-time form validation, dynamic stock level updates on dashboards, and providing immediate feedback to users through interactive elements. By avoiding the use of external frameworks, Invex IMS maintained a lightweight architecture, ensuring fast performance—an essential requirement for delivering timely updates and a responsive user experience.

The use of JavaScript also eliminated dependency risks associated with third-party libraries, such as version conflicts, ensuring better long-term maintainability. Moreover, JavaScript offers greater flexibility in how code is organized and implemented, allowing for the development of custom features without being constrained by the architectural conventions of external frameworks. Additionally, it served as a valuable learning tool, providing a deeper understanding of core web development principles such as event handling, asynchronous operations, and dynamic content rendering.

### Backend Technologies

In the development of Invex IMS, Python Django was used as the primary backend framework to handle server-side logic, data management, and application security. Django’s robust features and built-in capabilities made it an ideal choice for building a reliable and scalable inventory management system.

Django was utilized to implement key backend functionalities such as user authentication and authorization, ensuring secure access to sensitive inventory data. Its built-in user management system allowed for straightforward creation, management, and permission assignment for different user roles, such as admins and staff. This facilitated controlled access and maintained data integrity across the system.

For data management, Django’s Object-Relational Mapping (ORM) was employed to define models representing inventory items, orders, suppliers, and other relevant entities. This ORM simplified database interactions by allowing developers to work with Python objects instead of raw SQL queries, ensuring efficient data retrieval, updates, and deletions. Django’s migration system also enabled seamless database schema evolution without disrupting existing data.

The framework’s templating engine was used to dynamically generate HTML pages, integrating server-side data with frontend components. This facilitated real-time updates on dashboards, order processing views, and inventory reports, providing users with accurate and current information.

Django’s built-in security features, such as protection against common web vulnerabilities (e.g., cross-site scripting, CSRF attacks), ensured that the system maintained high security standards. Its modular architecture allowed for the development of RESTful APIs, which were necessary for asynchronous frontend interactions via JavaScript, enabling features like real-time stock level updates and order processing without full page reloads.

Django’s comprehensive documentation, community support, and extensive libraries made it sufficient for implementing all core backend functionalities required by Invex IMS. Its scalability and robustness ensured that the system could handle increasing data volume and user load, making Django an effective and appropriate choice for achieving the system's backend requirements.

### Database

In the Invex IMS project, PostgreSQL was chosen as the primary database management system due to its robustness, scalability, and advanced features, making it highly suitable for managing complex inventory data and supporting the application's dynamic requirements.

PostgreSQL's strong compliance with SQL standards and its support for advanced data types allowed for efficient and flexible data modeling. It was used to define and manage the database schema, including tables for inventory items, orders, suppliers, users, and transaction histories. Its relational structure enabled the establishment of meaningful relationships between entities, such as linking inventory items to suppliers or orders to users, which facilitated comprehensive data integrity and consistency.

The database supported complex queries essential for real-time features, such as fetching current stock levels, generating inventory reports, and tracking order statuses. PostgreSQL's powerful indexing capabilities improved query performance, ensuring quick data retrieval even as the data volume grew. Its support for advanced indexing methods, like B-tree and GIN indexes, optimized searches and filtering operations critical for the system’s responsiveness.

PostgreSQL also offered features like transactional integrity and concurrency control, which were vital for maintaining accurate inventory counts during simultaneous operations, such as multiple users updating stock levels or processing orders concurrently. These features ensured data reliability and prevented conflicts or inconsistencies.

Additionally, PostgreSQL’s extensibility allowed for custom functions and stored procedures to automate routine data operations or enforce complex business rules directly within the database layer, reducing application load and increasing efficiency.

Its compatibility with Django ORM made it straightforward to integrate PostgreSQL with the backend, enabling seamless data manipulation through Django models while leveraging PostgreSQL's advanced features under the hood.

Overall, PostgreSQL was suitable because it provided a reliable, scalable, and feature-rich database solution capable of supporting the complex data management needs of Invex IMS, ensuring data integrity, fast performance, and flexibility for future growth and feature expansion.

## **Testing**

### 1. User Registration

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-UR-01 | Valid registration | User not registered | 1. Access registration page  2. Fill in valid details  3. Submit form | OTP sent to email; registration success message displayed | N/A | Ensure email is valid and unique |
| TC-UR-02 | Invalid email or ID | User not registered | 1. Access registration page 2. Fill in invalid email or ID 3. Submit form | Error message indicating invalid email or ID | N/A | Test with malformed email or duplicate ID |
| TC-UR-03 | Incorrect OTP entry | OTP sent to email | 1. Enter correct details 2. Enter wrong OTP 3. Submit | System prompts for re-entry or resend OTP | User re-enters OTP correctly | Test OTP expiry and resend options |
| TC-UR-04 | OTP timeout | OTP sent to email | 1. Wait beyond OTP validity period 2. Enter OTP | System prompts to request new OTP | User requests new OTP and completes registration | Ensure timeout handling |

*Table 1: User registration test case*

### 2. User Login

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-UL-01 | Valid login | User account exists | 1. Enter email and password 2. Submit | Redirected to dashboard | N/A | Verify successful login |
| TC-UL-02 | Invalid credentials | User account exists | 1. Enter wrong email/password 2. Submit | Error message: invalid credentials | N/A | Test with incorrect combinations |
| TC-UL-03 | Inactive account | User account is inactive | 1. Enter email/password 2. Submit | Prompt to complete verification or activation | N/A | Ensure inactive accounts are restricted |

*Table 2: Login test case*

### 3. Creating a Shop

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-CS-01 | Successful shop creation | User logged in | 1. Navigate to “Create Shop” 2. Fill in valid details 3. Submit | Shop saved, redirected to shop detail page | N/A | Confirm unique shop name |
| TC-CS-02 | Duplicate shop name | User logged in | 1. Enter existing shop name 2. Submit | Warning: Name already exists | N/A | Validate name uniqueness |
| TC-CS-03 | Missing required fields | User logged in | 1. Leave required fields empty 2. Submit | Errors displayed; shop not created | N/A | Ensure form validation |

*Table 3: Creating Shop test case*

### 4. Creating a Product

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-CP-01 | Successful product creation | User has a shop | 1. Click “Add Product” 2. Fill in details 3. Submit | Product added to inventory | N/A | Check data validation |
| TC-CP-02 | Missing or invalid data | User has a shop | 1. Leave fields empty or invalid 2. Submit | Errors shown, product not created | N/A | Validate input fields |

*Table 4: Creating product test case*

### 5. Selling a Product

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-SP-01 | Successful sale | Product exists, quantity > 0 | 1. Click “Sell” 2. Enter quantity 3. Confirm | Stock reduced; sale recorded | N/A | Verify stock update |
| TC-SP-02 | Insufficient stock | Product with low stock | 1. Attempt to sell more than available | Warning: Insufficient stock | N/A | Ensure stock check logic works |
| TC-SP-03 | Invalid input | Product exists | 1. Enter negative quantity 2. Attempt to sell | Sale is blocked, error message shown | N/A | Validate input constraints |

*Table 5: Selling test case*

### 6. Adding Stock

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-AS-01 | Successful stock addition | Product exists | 1. Click “Add” 2. Enter quantity and password 3. Submit | Quantity added, success message | N/A | Verify password validation |
| TC-AS-02 | Incorrect password | Product exists | 1. Enter wrong password 2. Submit | Authentication fails; stock not updated | N/A | Ensure security check |
| TC-AS-03 | Invalid quantity | Product exists | 1. Enter invalid quantity (e.g., negative) 2. Submit | Validation error, action aborted | N/A | Validate quantity input |

*Table 6: Adding stock test case*

### 7. Creating an Employee

| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** | **Alternate Flows** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| TC-CE-01 | Successful employee creation | Shop exists | 1. Navigate to “Add Employee” 2. Fill details 3. Submit | Employee added to records | N/A | Check for duplicate national ID |
| TC-CE-02 | Duplicate national ID | Shop exists | 1. Enter existing national ID 2. Submit | Error message: duplicate ID | N/A | Validate uniqueness |
| TC-CE-03 | Missing required fields | Shop exists | 1. Leave fields empty 2. Submit | Prompt to complete required fields | N/A | Ensure form validation |

*Table 7: Creating an employee test case*

# CHAPTER 5: CONCLUSION AND RECOMMENDATION

## **Conclusion**

Invex IMS effectively addresses the core needs of small-scale shop owners by providing a user-friendly platform to manage shop operations remotely. The system streamlines essential tasks such as creating and managing shop profiles, adding and editing products, and handling sales transactions. Its features for stock updates, product deletion, and employee management enable shop owners to maintain comprehensive control over their inventory and personnel. Overall, Invex IMS demonstrates that a lightweight, accessible inventory management solution can significantly enhance operational efficiency and remote oversight for small businesses, reducing reliance on manual record-keeping and physical supervision.

## **Recommendations**

To further enhance Invex IMS’s utility and user experience, the **Enable Mobile Compatibility:** Optimize the system for mobile devices to facilitate on-the-go management, considering many shop owners may rely on smartphones.

1. **Provide Data Backup and Export Options:** Enable easy data backup and reports exporting capabilities to safeguard information and facilitate offline record-keeping.
2. **Incorporate Barcode Functionality:** Future updates should include barcode integration for products, allowing for quick scanning during sales, stock updates, and inventory checks—improving efficiency and reducing manual errors.
3. **Gather User Feedback for Continuous Improvement:** Regularly collect feedback from users to identify pain points and prioritize new features or improvements.

# 

# REFERENCES

** Django Software Foundation. (2024). *Django Documentation*. Retrieved from** [**https://docs.djangoproject.com/**](https://docs.djangoproject.com/)

 PostgreSQL Global Development Group. (2024). *PostgreSQL: The World's Most Advanced Open-Source Relational Database*. Retrieved from <https://www.postgresql.org/>

 Bootstrap Team. (2024). *Bootstrap 5 Documentation*. Retrieved from https://getbootstrap.com/docs/5.3/getting-started/introduction/

 Chart.js Contributors. (2024). *Chart.js: Simple yet flexible JavaScript charting*. Retrieved from https://www.chartjs.org/docs/latest/

 GitHub. (2024). *GitHub Documentation: Version Control with Git*. Retrieved from <https://docs.github.com/>

 Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

 Pressman, R. S. & Maxim, B. R. (2014). *Software Engineering: A Practitioner’s Approach* (8th ed.). McGraw-Hill Education.

 ISO/IEC 25010:2011. (2011). *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. International Organization for Standardization.

 W3C. (2024). *Web Accessibility Guidelines (WCAG) 2.1*. Retrieved from <https://www.w3.org/WAI/standards-guidelines/wcag/>

# APPENDICES

## Appendix 1: User Manual