**1.PROJECT PROPOSAL**

**1.1 Background to the Problem**

In Bangladesh, many small and medium retail shops still rely heavily on manual processes for their daily operations. Administrators and employees commonly use traditional methods such as writing product stock in notebooks, maintaining sales records manually, checking product availability by verbal confirmation, and tracking employee performance without any digital support. In many cases, multiple separate tools are used for inventory, billing, and user accounts, which remain disconnected and inefficient. These outdated practices generate several operational challenges: High error rate: Manual stock updates and handwritten invoices often lead to incorrect sales entries, stock mismatches, or missing items. Slow sales processing: Employees spend unnecessary time finding product details and preparing invoices manually, which increases customer waiting time. Weak security: Sensitive data like passwords, sales records, and financial information often lack proper protection. Poor monitoring: Admins struggle to track daily sales, best-selling products, and employee contributions due to lack of real-time reports. Scalability issues: As shops grow, manual systems fail to handle larger inventories, increased invoices, and more complex data. To address these issues, a unified digital solution is required—one that centralizes product management, sales processing, invoicing, user authentication, and reporting. The proposed Shop Management System aims to eliminate manual errors, automate key tasks, improve security, and support long-term scalability for retail businesses. It provides a structured, efficient, and secure environment where both admins and employees can operate smoothly.

**Existing Software Solutions**

Some software solutions currently used by shops and small retail businesses include:

**• Excel / Google Sheets** – Used for sales and stock records, but no security, role-based access, or real-time updates.

• **Manual Billing Systems** – Basic invoicing only, no integrated inventory or reporting.

• **Advanced POS Software** – Feature-rich but expensive and complex for small shops.

None of these provide a simple, affordable, and integrated shop management solution suitable for small retail businesses.

**1.2 Selection of Process Model**

**1.Selected Software Development Process Model: Scrum**

We selected the Scrum model for developing the Shop Management System because it supports incremental delivery, continuous feedback, and role-based collaboration, which perfectly matches the evolving needs of retail management systems.

Scrum divides the project into short iterations called Sprints, allowing frequent inspection and adaptation.

**2. Why Scrum is Suitable for This Project**

• Requirements may evolve based on admin and employee feedback

• Product catalog, sales, and security features can be developed incrementally

• Scrum supports frequent testing and stakeholder review

• Works well with small-to-medium development teams

• Ensures transparency, accountability, and continuous improvement

**3. Project Environment & Requirements**

**Stable Requirements**

• Secure login and authentication

• Role-based access control (Admin & Employee)

• Product catalog management

• Sales invoicing and stock tracking

Changing Requirements

• Report formats

• UI improvements

• Notification types

• Sales analytics depth

Scrum handles this stable + changing requirement balance efficiently.

**4. Scrum Roles and Responsibilities**

**4.1 Product Owner**

• Represents business needs (Shop Admin)

• Defines and prioritizes Product Backlog

• Accepts or rejects completed features

**4.2 Scrum Master**

• Ensures Scrum rules are followed

• Removes team obstacles

• Facilitates sprint meetings

**4.3 Development Team**

• Backend Developer

• Frontend Developer

• UI/UX Designer

• QA/Test Engineer

Team is cross-functional and self-organizing.

**5. Product Backlog**

|  |  |
| --- | --- |
| **Backlog Item** | **Priority** |
| Secure login & authentication | High |
| Role-based access (Admin/Employee) | High |
| User profile management | High |
| Product catalog management | High |
| Sales invoice generation | High |
| Real-time stock update | Medium |
| Sales history & reporting | Medium |
| Error handling & notifications | Medium |
| UI enhancements | Low |
| Advanced analytics | Low |

**6. Sprint Planning**

Each Sprint duration: 2 weeks

**Sprint 1: Core Security & Authentication**

• Login system

• Password encryption

• Role-based authorization

**Sprint 2: Product & Inventory Management**

• Add/edit products

• Stock tracking

• Search by name, size, gender, ID

**Sprint 3: Sales & Invoicing**

• Invoice creation

• Quantity control

• Real-time stock feedback

**Sprint 4: Reports & User Experience**

• Sales history

• Top-selling products

• Error messages & notifications

**7. Scrum Events**

**Sprint Planning**

• Team selects backlog items

• Defines sprint goal

**Daily Scrum**

• 15-minute stand-up

• What I did / What I’ll do / Blockers

**Sprint Review**

• Demo to Admin & Employees

• Feedback collection

**Sprint Retrospective**

• What went well

• What needs improvement

• Action items for next sprint

**8. Incremental Delivery (MVP Approach)**

• First MVP includes login, product management, and invoicing

• Each sprint adds usable features

• System remains deployable at all times

**9. Risk Management in Scrum**

• Requirement risk → Managed via backlog refinement

• Security risk → Regular testing each sprint

• Schedule risk → Velocity tracking

• Quality risk → Continuous integration & testing

**10. Flexibility of Scrum**

• New features can be added to backlog

• Priority can change anytime

• Continuous customer involvement ensures correctness

**11. Creative & Practical Solution**

• Centralized shop management

• Secure and role-based operations

• Real-time data for decision-making

• Simple and efficient UI for employees

**12. Target Users and Benefits**

**Shop Administrator**

• Full control over users, products, and reports

• Secure system management

**Employees**

• Simple sales process

• Easy product search & invoicing

**13. Justification of Scrum Selection**

• Waterfall → Too rigid

• XP → More developer-centric

• Scrum → Best balance of control, flexibility, and collaboration

Scrum ensures high-quality delivery, transparency, and continuous improvement, making it ideal for this system.

**1.3 Project Role Identification and Responsibilities**

(Scrum-based – For Shop Management System)

**1.3.1 Main Roles**

Product Owner – Represents shop business needs, prioritizes the Product Backlog, and validates features from Admin perspective.

Scrum Master – Ensures Scrum practices are followed, removes team blockers, and facilitates Scrum events.

System Analyst – Gathers requirements from shop admin and employees and refines user stories.

UI/UX Designer – Designs simple and user-friendly interfaces for dashboards, product, and sales flows.

Backend Developer – Develops server-side logic, APIs, database management, and business rules.

Frontend Developer – Builds web/mobile interfaces for login, product management, and invoicing.

QA/Test Engineer – Tests each sprint increment, ensures functionality, security, and reliability.

**1.3.2 Responsibilities at Key Stages**

• Requirements Gathering – Product Owner and System Analyst work with shop admin and employees.

• Design – UI/UX Designer prepares mockups; developers review technical feasibility.

• Implementation – Development team implements backlog items during sprints.

• Testing – QA Engineer and developers perform unit, integration, and sprint testing.

• Deployment – Scrum Master coordinates releases; developers ensure system stability.

**1.3.3 Responsibility Distribution & Justification**

• Product Owner manages priorities to ensure business value is delivered first.

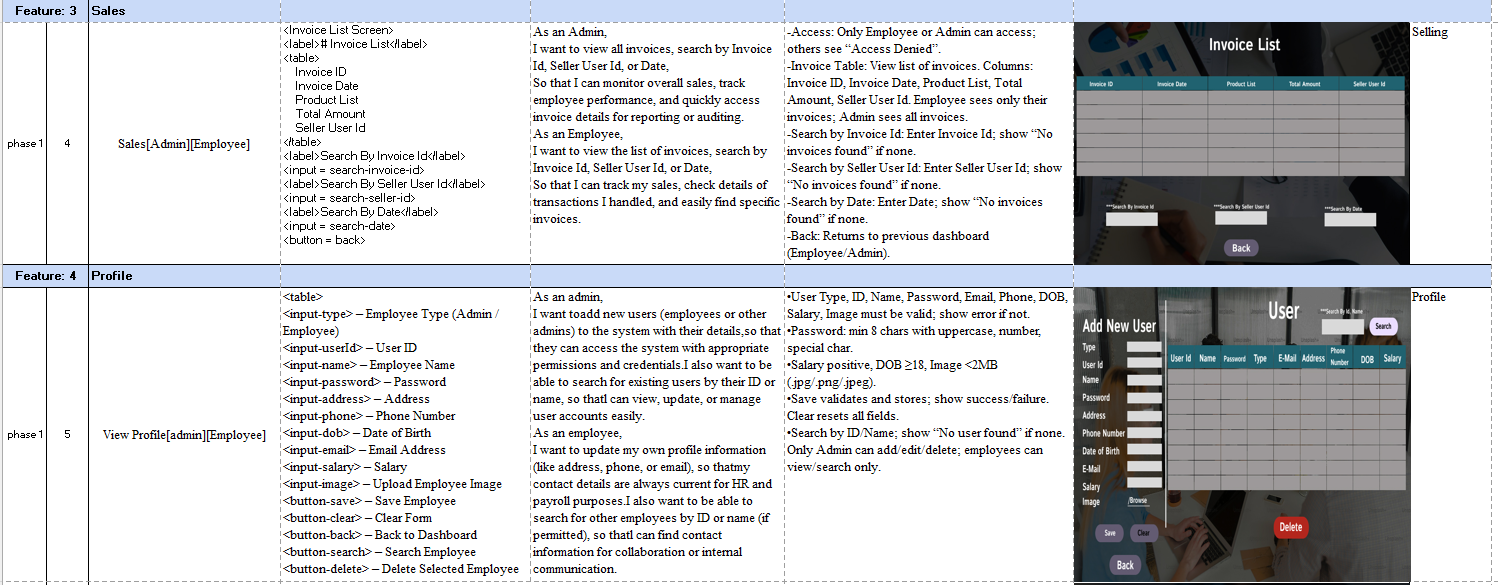
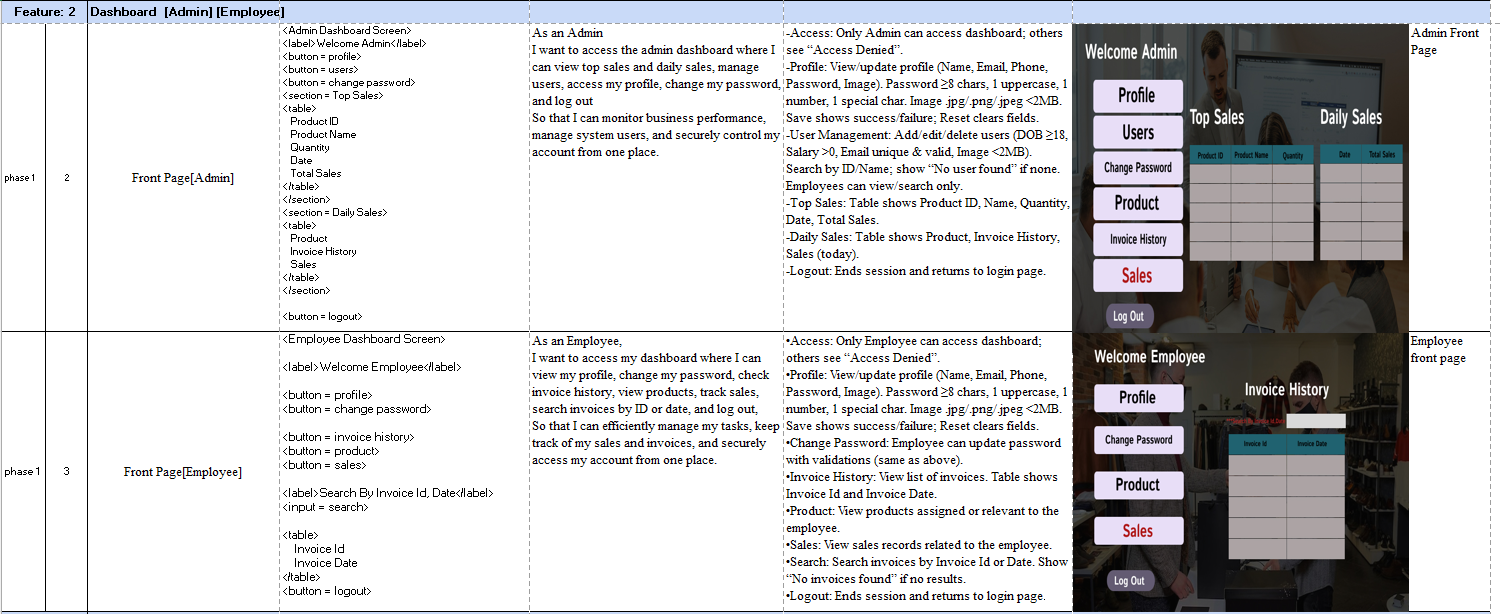
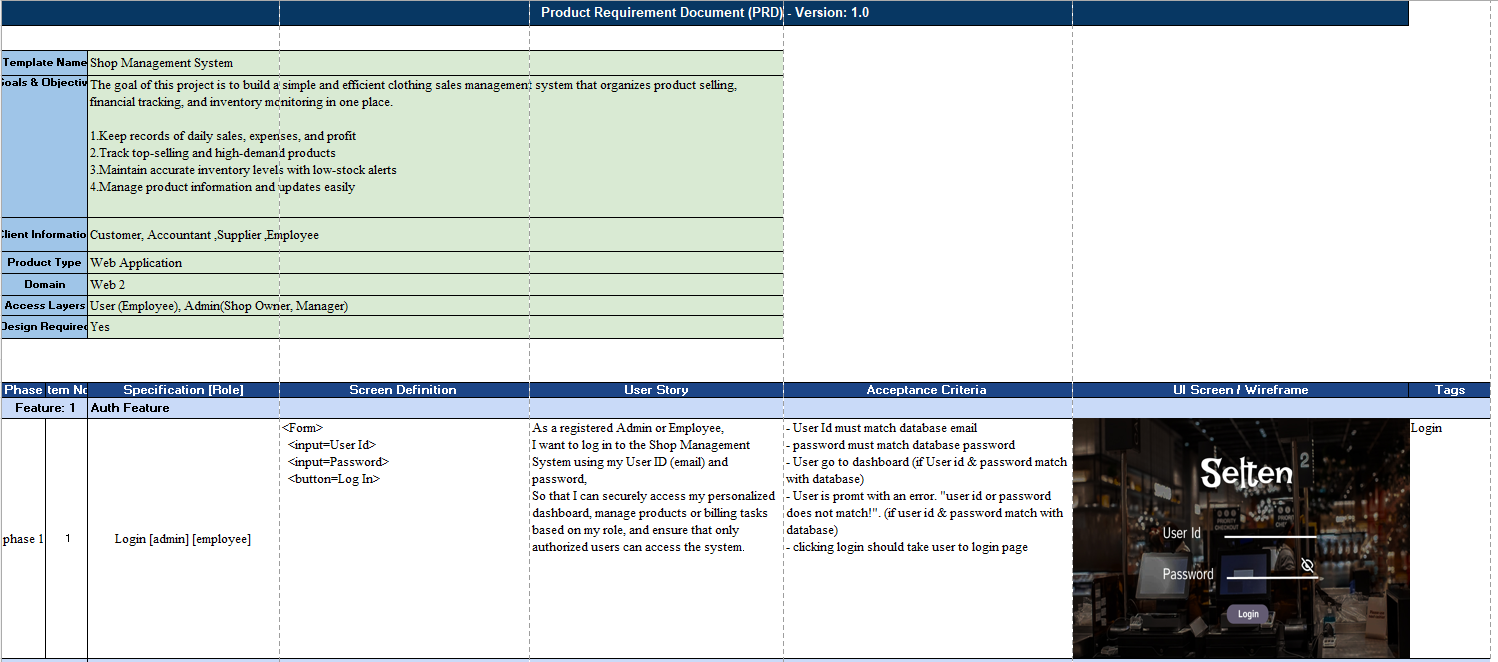
• Scrum Master ensures smooth sprint execution and team productivity.

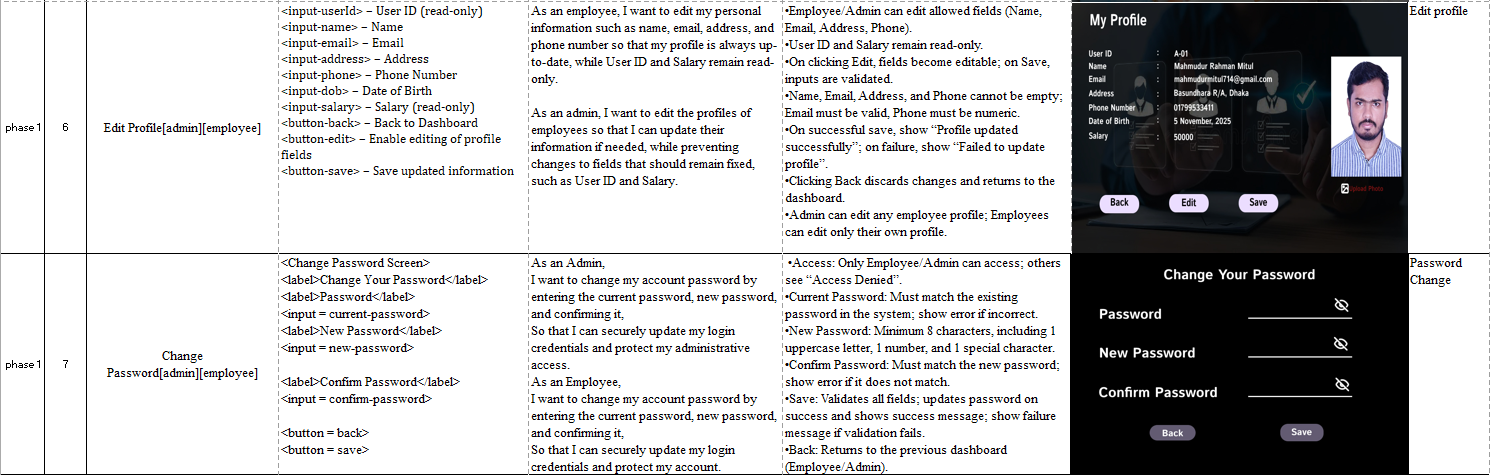
• Developers share implementation tasks to improve maintainability and knowledge transfer.

• QA focuses on system correctness, security, and data accuracy.

This role distribution ensures efficient collaboration, clear accountability, and continuous delivery, which aligns well with the Scrum-based development approach.

**2. SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) / PRODUCT REQUIREMENTS DOCUMENT (PRD)**

****

****

**2.1 Project Scope and Features**

The proposed Shop Management System covers all essential operations needed by both admins and employees:

**1. Authentication & Role Management**: Secure login with strong passwords. Automatic generation of unique user IDs. Role-based access for Admin and Employee. Protected profile and password management.

**2. User Profile Management:** Admins can view and update employee data (contact info, salary, account status). Employees can update their own profile and password. Prevents unauthorized edits and ensures privacy.

**3. Product Catalog & Inventory Management:** Add, edit, delete, and view product details (name, category, size, price, stock). Real-time stock tracking to prevent overselling. Organized catalog with search and filter options.

**4. Sales & Invoice Management**: Product search by ID, name, gender, or size. Add items to invoice with quantity validation. Automatic calculation of totals. Auto-generated unique invoice IDs. Access to past sales records for employees.

**5. Reporting & Analytics:** Daily, weekly, and monthly sales summaries. Identification of top-selling products. Low-stock notifications. Employee sales performance overview.

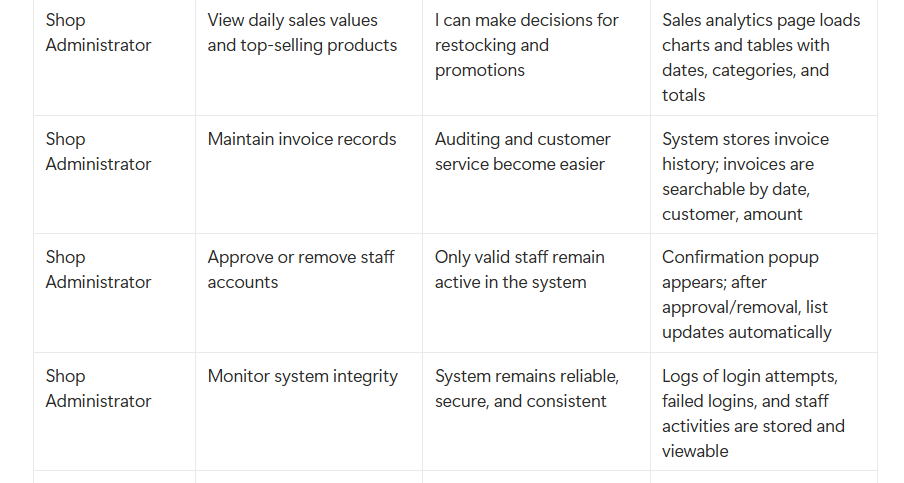
**6. Notifications & System Feedback:** Clear error messages for invalid input. Success messages for updates and sales. Smooth transitions to avoid workflow interruptions.

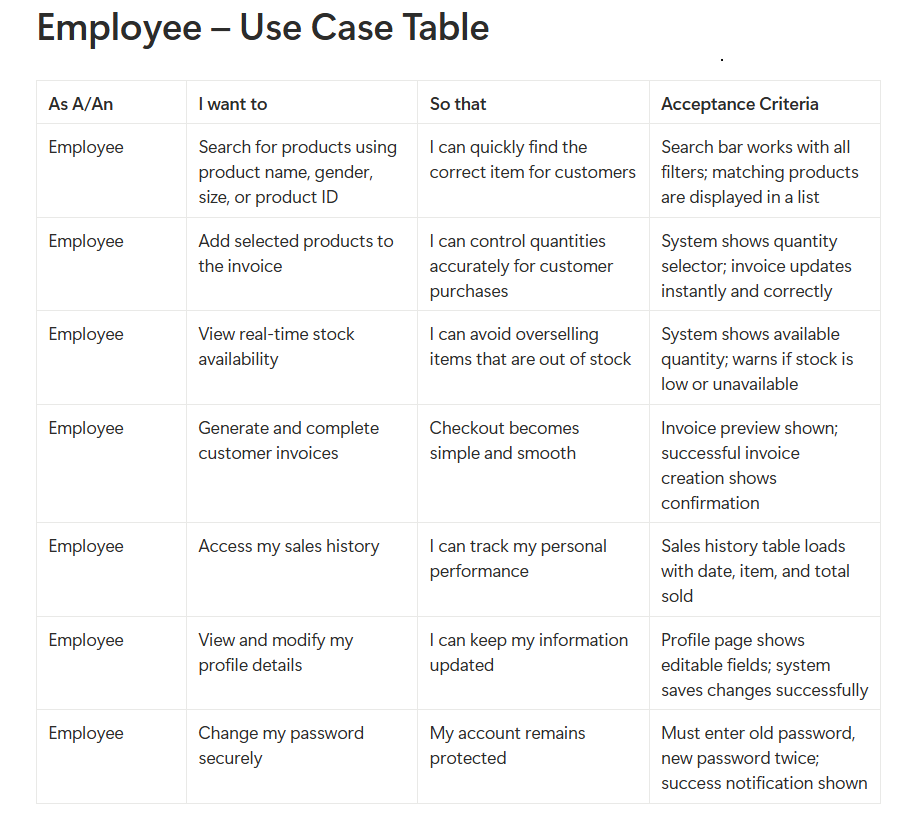
**7. Security Features:** Strong password enforcement. Protection of business data and user credentials. Unique IDs for users, products, and invoices to prevent conflicts.

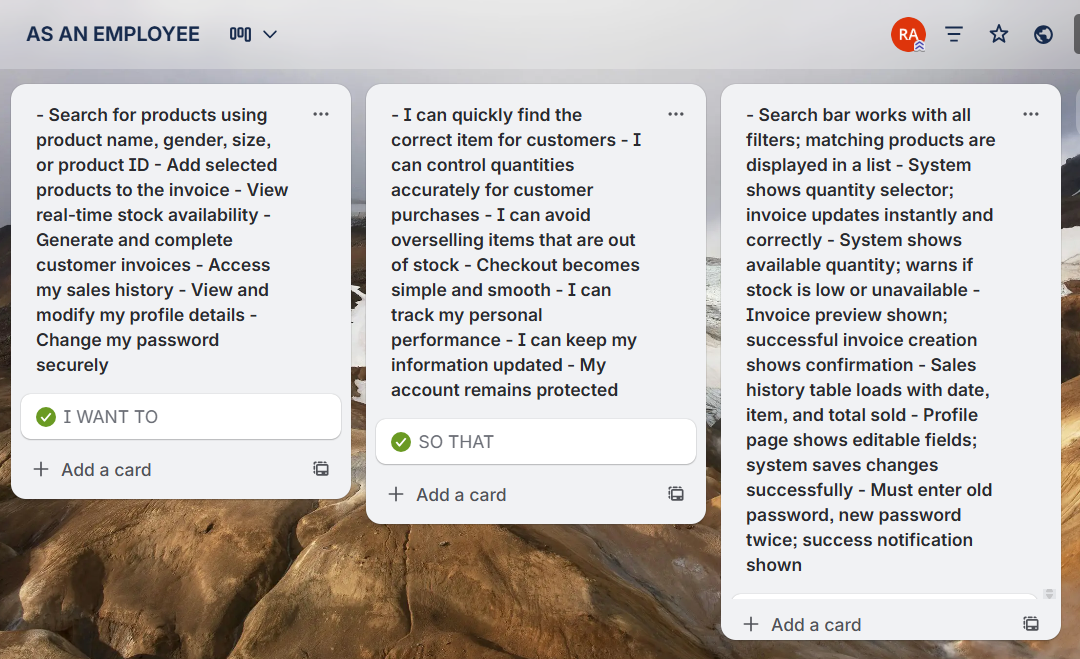
**2.2 User Story Table**

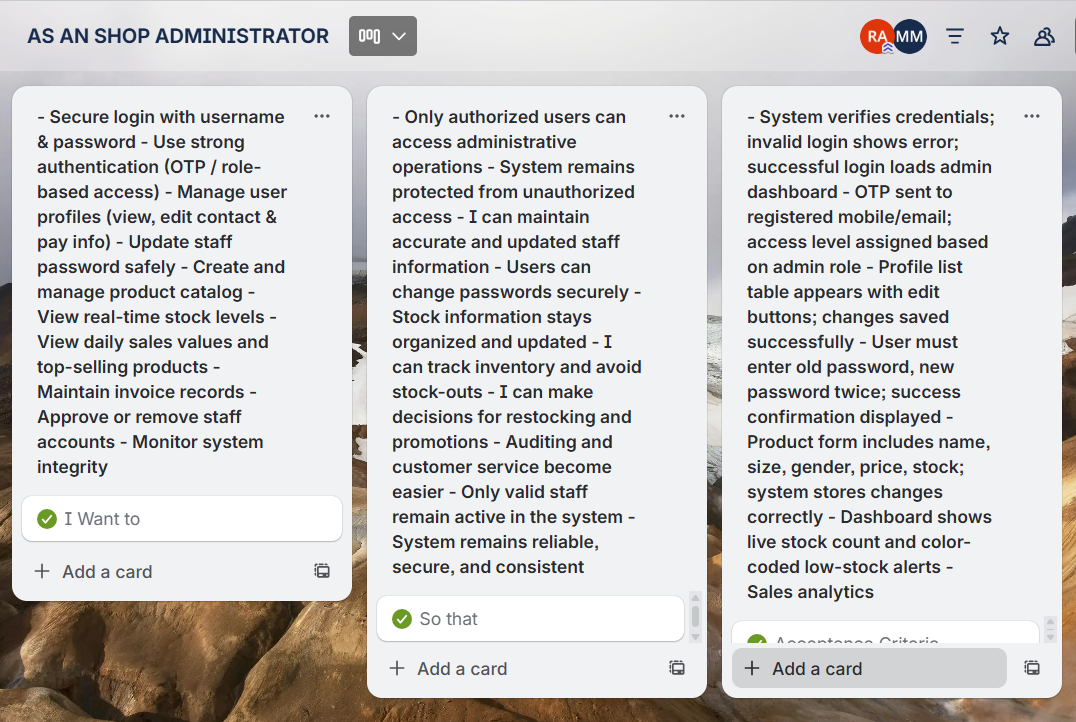
Here is the expanded explanation of user needs:





****

****

****

**2.3.1 Functional Requirements**

**1. Description of Major Functionalities**

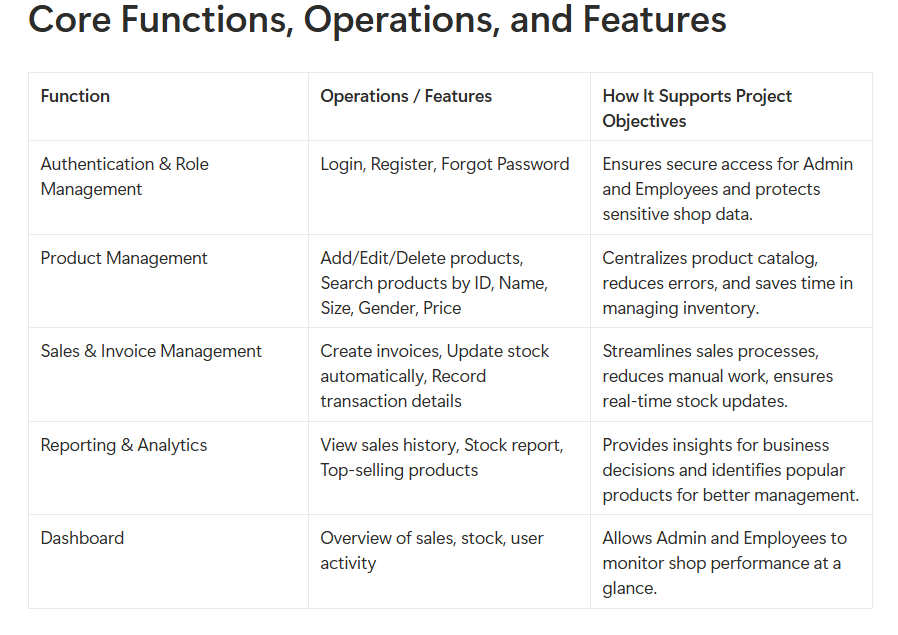
The Shop Management System is an integrated platform for small and medium retail businesses to simplify daily operations. The system’s major functionalities focus on secure, efficient, and centralized management of shop tasks, reducing manual work and improving accuracy.

The platform includes:

* Authentication & Role Management – Secure login and access control for Admin and Employees.
* Dashboard – Overview of products, sales, and stock status.
* Product Management – Add, edit, delete, and search products.
* Sales & Invoice Management – Create sales invoices and update stock automatically.
* Reporting & Analytics – View sales history, stock levels, and top-selling products.

The primary objective is to replace manual record-keeping and fragmented tools (Excel, manual billing, multiple apps) with a single, organized system.

**2.Core Functions ,Operations, and Features**

****

**3. How Functions Support Project Objectives**

Each module of the Shop Management System addresses specific real-world shop challenges:

**1.Authentication & Role Management**: Prevents unauthorized access and protects sensitive sales and stock data.

**2.Product Management:** Centralizes product records, avoids duplication, and simplifies updates.

**3.Sales & Invoice Management:** Automates stock updates and billing, reducing manual errors and saving time.

**4.Reporting & Analytics:** Provides transparency, improves decision-making, and reduces the risk of stock-outs.

**5.Dashboard:** Combines all key operations in one interface, reducing reliance on multiple tools and simplifying management.

**4. Workflow Description**

Workflow for Adding a New Product:

1.Admin logs into the system.

2.Navigates to the Product Management section.

3.Clicks “Add Product” and fills in details: Product Name, ID, Size, Gender, Quantity, Price.

4.Clicks “Save” to register the product.

5.The new product appears in the product catalog, available for sale and reporting.

**5. Acceptance Criteria**

* User Registration & Login: Admin or Employee must successfully log in after registration. System validates credentials and provides secure access.
* Product Addition: New products must appear immediately in the catalog with correct details.
* Sales Invoice Creation: Invoice creation updates stock automatically and records transaction details with date and time.
* Reporting: Sales and stock reports must reflect accurate data in real-time.
* Dashboard: Displays all key metrics correctly for Admin and Employees.

**2.3.2 Non-Functional Requirements**

**1. Quality Attributes**

The Shop Management System prioritizes usability, reliability, security, and scalability, ensuring it provides a professional, dependable, and growing platform.

**2. Performance**

Pages like Product Dashboard and Sales Invoice should load within 2–3 seconds.

Stock updates and invoice generation should be near-instantaneous for smooth operation.

**3. Reliability**

High availability with minimal downtime.

Secure backend infrastructure ensures stable data storage and synchronization.

**4. Integrity / Security**

Encrypted storage and secure login protect sensitive shop data (product, stock, sales records).

Role-based access ensures only authorized users can perform restricted operations.

Financial transaction data is stored securely to prevent tampering.

**5. Usability**

Intuitive, simple UI for Admin and Employees, even for non-technical users.

Clear navigation, accessible design, and minimal clicks for key operations.

**6. Maintainability**

Built with modular design for easy updates and future feature additions.

Use of open-source frameworks allows quick bug fixes and scalability.

**7. Scalability**

Can handle increasing number of products, sales transactions, and users.

Supports incremental growth as the shop expands its operations.

**3. Software Design**

**3.1 System Design**

**Use Case Design**

****

**Use Case Diagram – Shop Management System**

To prepare the Use Case Diagram, the system users (actors) and their roles are first identified. Their interactions with the system are represented through use cases placed inside the Shop Management System boundary. Relationships such as <> and <> are used where required.

**1. Actors and Roles**

**Admin:**

The Admin is the main user with full control over the system. The Admin manages employees, products, sales, and reports.

**Employee:**

The Employee works under the Admin and performs sales-related tasks with limited access.

**2. System Boundary**

All system functionalities are enclosed within the Shop Management System boundary. Actors remain outside and interact through defined use cases.

**3. Use Cases**

Common Use Cases (Admin & Employee):

* Login
* Show Profile
* Update Profile
* Sales Product
* Show Invoice History

**Admin Use Cases:**

* Manage Employee Profile
* Add Employee
* Remove Employee
* Update Product
* Show Top-Selling Products
* View Daily Sales Report
* Employee Use Cases:
* Show Own Sales History

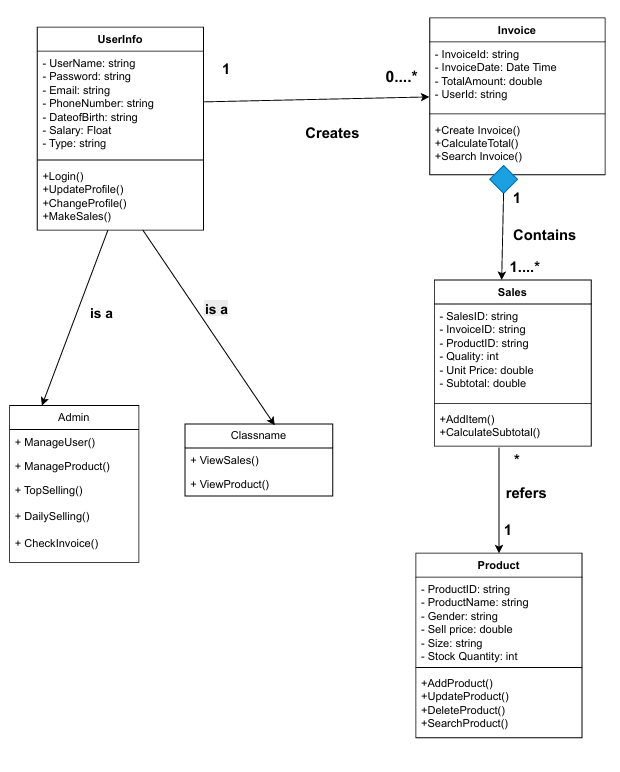
**4. Use Case Relationships**

* Manage Employee Profile <> Add Employee
* Manage Employee Profile <> Remove Employee
* Sales Product <> Show Invoice History
* Daily Sales Report <> Show Top-Selling Products

**5. Conclusion**

The Use Case Diagram clearly represents interactions between Admin and Employee within the Shop Management System. It highlights access control, system functionality, and the use of include and extend relationships for mandatory and optional behaviors.

Class Diagram



**1. UserInfo (Base Class)**

**• Attributes:**

userName: string

password: string

email: string

phoneNumber: string

dateOfBirth: string

salary: float

type: string

**• Methods:**

login(): void

updateProfile(): void

changeProfile(): void

makeSales(): void

**2. Admin (Inherits UserInfo)**

**• Methods:**

manageUser(): void

manageProduct(): void

topSelling(): void

dailySelling(): void

checkInvoice(): void

**3. Employee (Inherits UserInfo)**

**• Methods:**

viewSales(): void

viewProduct(): void

**4. Invoice**

**• Attributes:**

invoiceId: string

invoiceDate: DateTime

totalAmount: double

userId: string

**• Methods:**

createInvoice(): void

calculateTotal(): double

searchInvoice(): Invoice

**5. Sales**

**• Attributes:**

salesId: string

invoiceId: string

productId: string

quantity: integer

unitPrice: double

subTotal: double

**• Methods:**

addItem(): void

calculateSubtotal(): double

**6. Product**

**• Attributes:**

productId: string

productName: string

gender: string

sellPrice: double

size: string

stockQuantity: integer

**• Methods:**

addProduct(): void

updateProduct(): void

deleteProduct(): void

searchProduct(): Product

**7. Relationships (Implied in Diagram)**

**• Inheritance:**

UserInfo → Admin

UserInfo → Employee

**• Associations:**

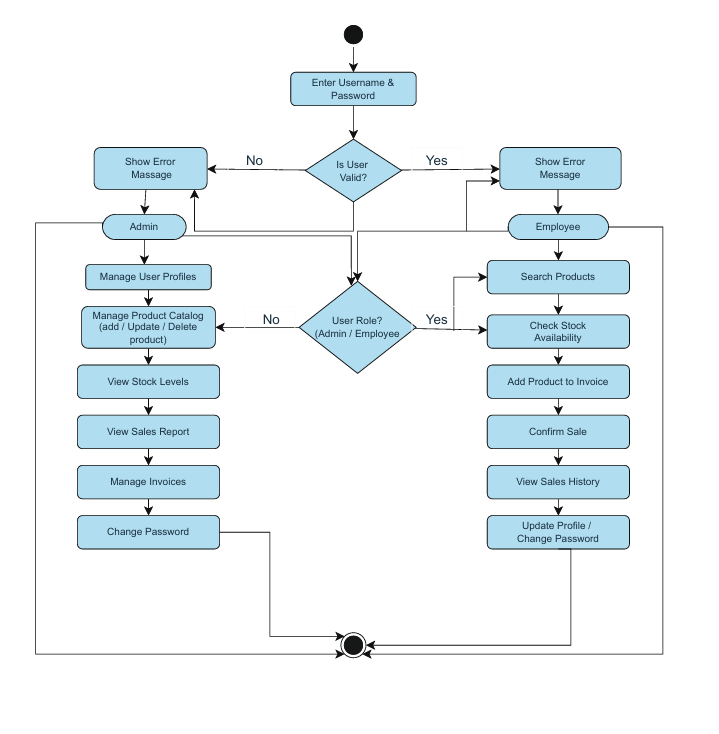
UserInfo creates Invoice

Sales is associated with Product

**• Composition:**

Invoice contains Sales

**Activity Diagram**

****

**Activity Diagram – Shop Management System**

* The Activity Diagram represents the flow of activities in the Shop Management System from start to end.
* The process begins when a user enters login credentials into the system.
* The system validates the credentials and checks whether the user is valid or not.
* If the user is invalid, an error message is displayed and the user is redirected to the login page.
* If the user is valid, the system identifies the user role (Admin or Employee).
* The Admin can manage users, manage products, view stock levels, generate sales reports, manage invoices, and change password.
* The Employee can search products, check stock availability, add products to invoices, confirm sales, view sales history, and update profile information.
* After completing the required actions, the system reaches the final state when the user logs out.

**Key Points**

* Shows dynamic workflow of the system
* Includes decision nodes for validation and role identification
* Separate activity flows for Admin and Employee
* Clearly defined start and end points

**3.2 UI Design Using Figma**

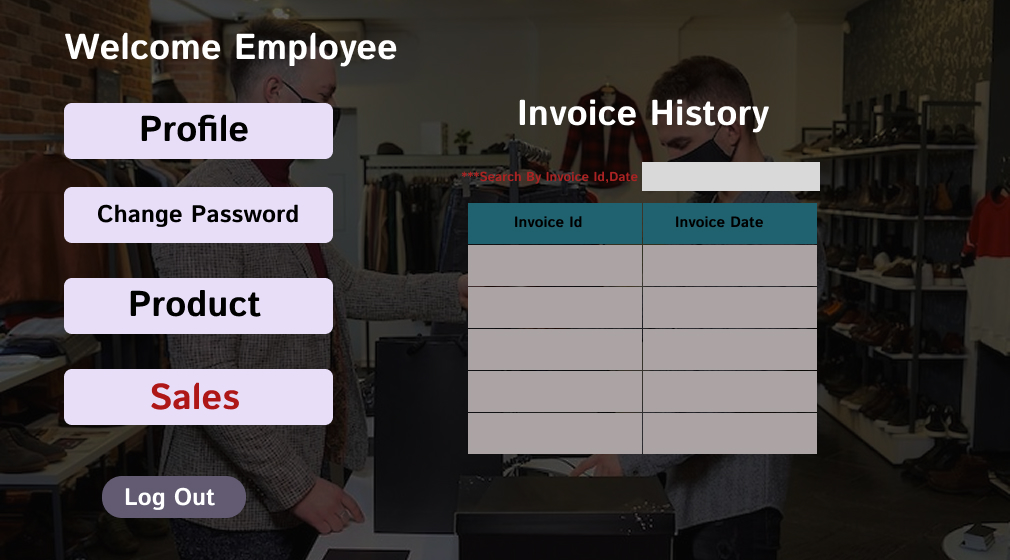
**LOGIN**



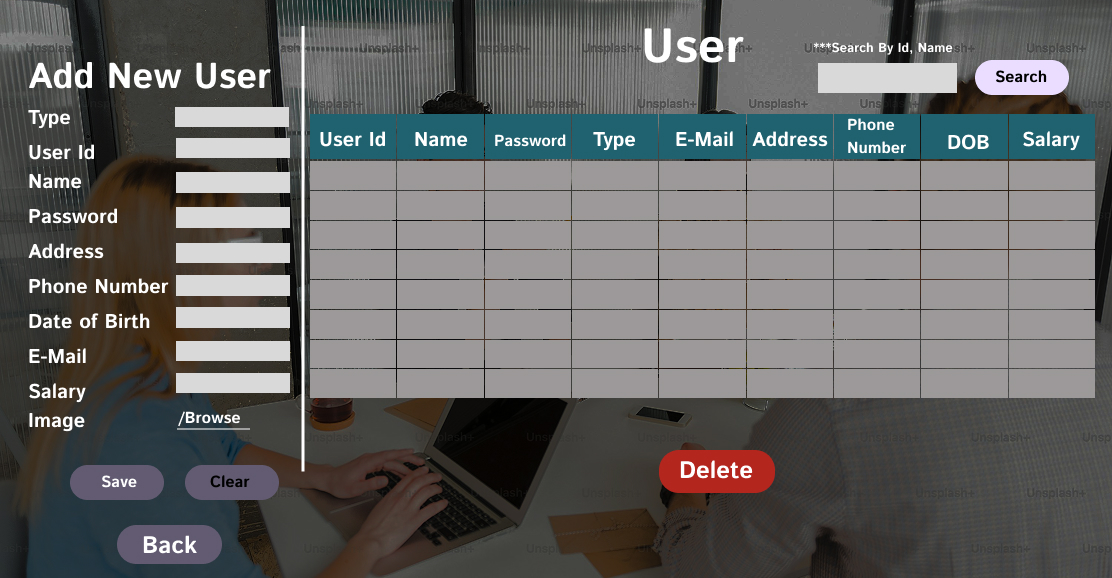
**ADMIN FRONT PAGE**



**EMPLOYEE FRONT PAGE**



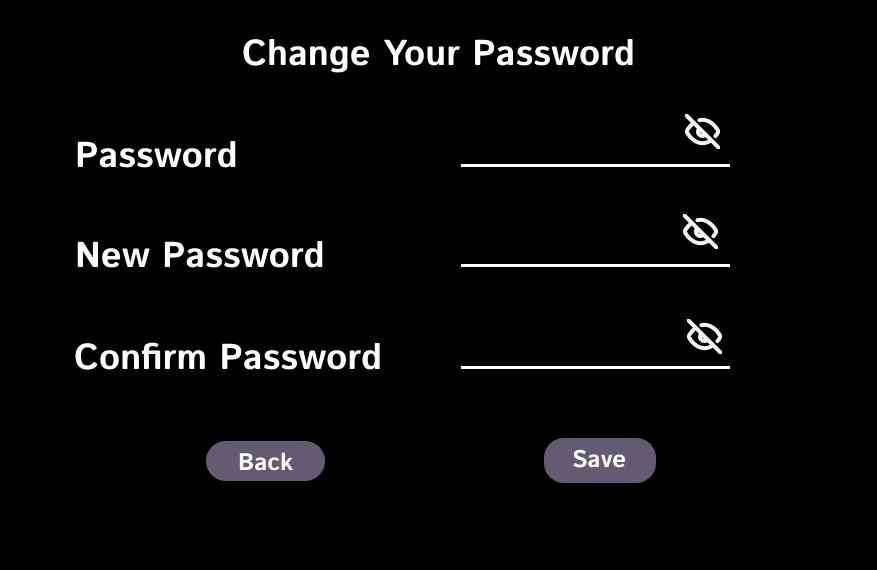
**USER**



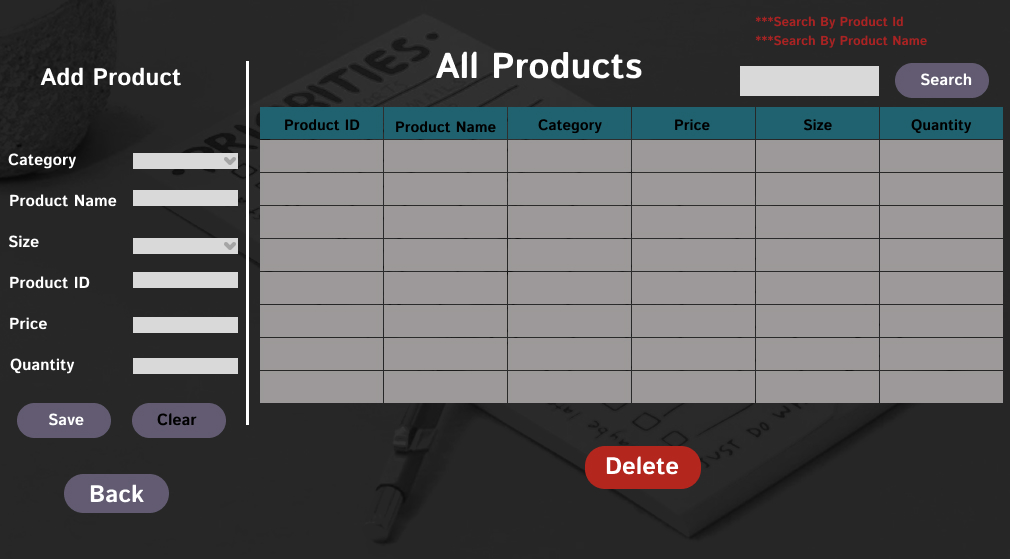
**PROFILE INFO**



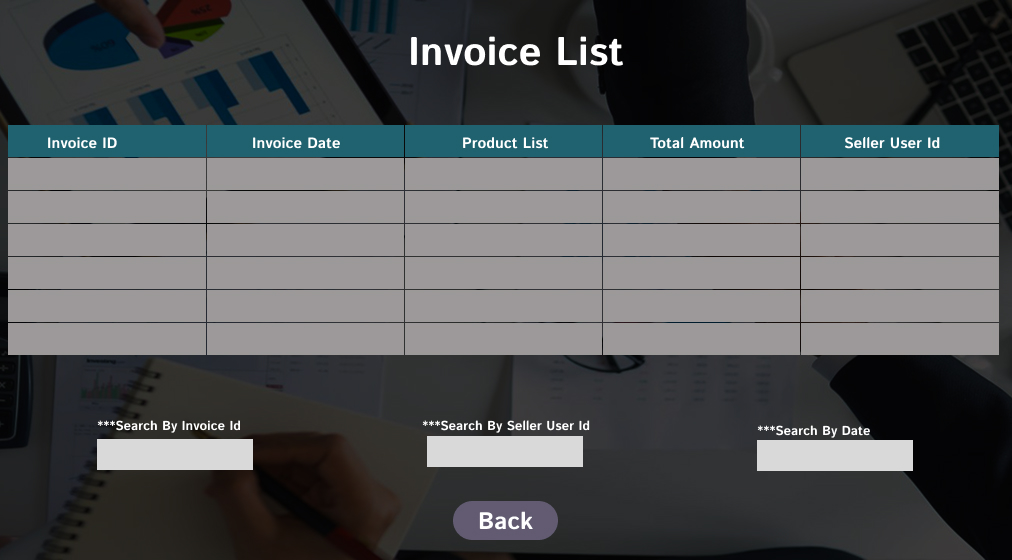
**PASSWORD**



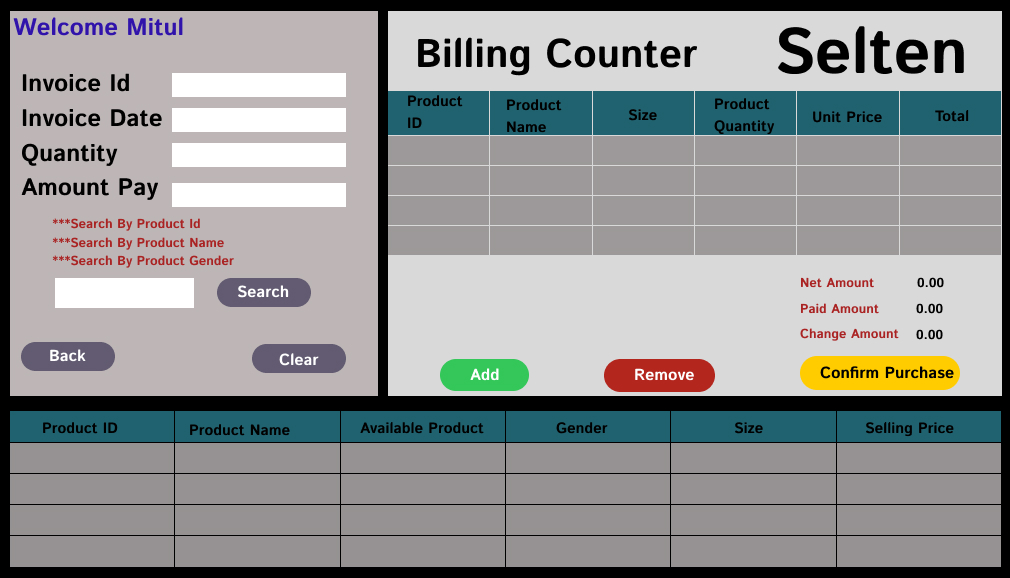
**PRODUCT**



**SALES HISTORY**



**BILLING COUNTER**

****

**6 CONCLUSION**

The Shop Management System project provides an effective solution to the operational challenges faced by small and medium retail shops in Bangladesh. Traditional management methods are often error-prone, time-consuming, and lack real-time insights. This system centralizes key operations, including product management, sales tracking, billing, user management, and employee monitoring, streamlining daily activities and improving efficiency.

Developed using the Scrum model, the project benefited from iterative development, frequent feedback, and adaptability to evolving requirements from shop owners and employees. Comprehensive system design, documented through Use-Case, Class, and Activity diagrams, ensured a strong architectural foundation. Implementation followed a structured version control workflow, promoting collaboration, accountability, and maintainability. Software product metrics were applied to assess functionality, performance, and usability, confirming the system’s reliability and quality.

Ultimately, the system is a practical, scalable, and user-friendly tool that reduces manual errors, saves time, and provides clear insights into sales and inventory. It empowers shop owners and employees to make informed decisions, optimize resources, and enhance customer service, making it a valuable asset for modern retail operations.

**6.1 FUTURE WORK**

* Future enhancements could further increase the system’s value and usability:
* Introduce advanced sales analytics for trend analysis and reporting.
* Integrate digital payment systems like bKash, Nagad, or bank APIs.
* Implement inventory forecasting to automate stock replenishment.
* Develop a mobile application for on-the-go management.
* Extend multi-user roles and permissions for better security and control.