 **PES UNIVERSITY**

**BENGALURU**

**Department of Computer Science and Engineering**

**Deliverable 4: Final Report**

**B. TECH. (CSE) V SEMESTER**

**Aug. – Dec. 2024**

**UE22CS341A – SOFTWARE ENGINEERING PROJECT REPORT**

**ON**

**BILLING MANEGMENT SYSTEM**

**SUBMITTED BY**

**1 . Vignesh N - PES1UG23CS836**

**2 . Sai Ganesh - PES1UG22CS546**

**Engineering Case Study**

**Title : Billing System**

|  |  |
| --- | --- |
| Sl. No. | TOPIC |
| 1. | Proposal Of The Project |
| 2. | Software Requirements Specification |
| 3. | Project Plan |
| 4. | Design Diagrams |
| 5. | Test Cases |
| 6. | Results (Screen Shots) |

Software Requirements Specification (SRS) for Billing System

# **Introduction**

* 1. **Purpose**  
     The purpose of this document is to specify the requirements for the Billing System in a department store. This system is designed to automate the billing process, manage product inventory, track customer transactions, and generate reports. The document outlines the system’s functionalities, performance, interface, and other critical factors necessary for development.
  2. **Scope**  
     The Billing System will be used in a department store to handle day-to-day sales operations, including product management, customer management, billing, and inventory control. The system will also generate various reports for sales and inventory analysis. It will be accessible by store staff with varying levels of permissions based on their roles (e.g., Admin, Staff).

**1.3 Definitions, Acronyms, and Abbreviations**

* **SRS**: Software Requirements Specification
* **GUI**: Graphical User Interface
* **DBMS**: Database Management System
* **CRUD**: Create, Read, Update, Delete

**1.4 References**

* Java Documentation: [Oracle Java Documentation](https://docs.oracle.com/en/java/)
* MySQL Documentation: [MySQL Documentation](https://dev.mysql.com/doc/)
* Software Engineering by Ian Sommerville, 10th Edition

**1.5 Overview**  
This document is structured to provide a comprehensive overview of the system, including detailed functional and non-functional requirements, system models, and interface descriptions.

# **Overall Description**

**2.1 Product Perspective**  
The Billing System is a standalone application that will integrate with the department store’s existing IT infrastructure. It is designed to replace manual billing processes, thereby reducing errors and improving efficiency. The system will interact with a backend database to store and retrieve data.

**2.2 Product Functions**

* **User Authentication**: Secure login and role-based access control.
* **Product Management**: Add, update, delete, and view product details.
* **Customer Management**: Manage customer information and purchase history.
* **Billing**: Generate and print bills for customers, apply discounts, and calculate totals.
* **Inventory Management**: Track stock levels and generate alerts for low stock.
* **Report Generation**: Produce sales and inventory reports for analysis.

**2.3 User Classes and Characteristics**

* **Admin**: Has full access to the system, including user management, product management, and report generation.
* **Staff**: Has limited access, primarily to the billing and inventory management modules.

**2.4 Operating Environment**

* **Operating System**: Windows/Linux/MacOS
* **Database**: MySQL or SQLite
* **Programming Language**: Java
* **Hardware**: Standard PC with at least 4 GB RAM, 100 GB HDD

**2.5 Design and Implementation Constraints**

* The system must be developed using Java for the front-end and SQL for the back-end.
* The system should be scalable to handle large amounts of data without performance degradation.
* The application must ensure data security, particularly for customer and transaction data.

**2.6 Assumptions and Dependencies**

* The users have basic knowledge of using a computer and can navigate a GUI.
* The system depends on a stable database connection for real-time operations.
* The store’s network infrastructure must support client-server communication.

# **System Features**

**3.1 User Authentication**  
**3.1.1 Description**  
The system will require users to log in with a username and password. User roles (Admin or Staff) will determine the level of access to system features.

**3.1.2 Functional Requirements**

* **REQ-1.1**: The system shall validate user credentials against the database.
* **REQ-1.2**: The system shall allow Admin users to manage user accounts.
* **REQ-1.3**: The system shall log user activities for audit purposes.

**3.2 Product Management**  
**3.2.1 Description**  
The system will allow users to add, update, delete, and view product information, including product name, category, price, and stock quantity.

**3.2.2 Functional Requirements**

* **REQ-2.1**: The system shall allow Admin users to add new products to the inventory.
* **REQ-2.2**: The system shall allow Admin and Staff users to update existing product details.
* **REQ-2.3**: The system shall allow Admin users to delete products from the inventory.
* **REQ-2.4**: The system shall display a list of products with search and filter options.

**3.3 Billing**  
**3.3.1 Description**  
The billing module will generate bills for customers, apply discounts, and calculate totals based on selected products.

**3.3.2 Functional Requirements**

* **REQ-3.1**: The system shall allow users to select products and quantity for billing.
* **REQ-3.2**: The system shall calculate the total bill amount, including applicable taxes and discounts.
* **REQ-3.3**: The system shall generate a printable bill receipt.

**3.4 Inventory Management**  
**3.4.1 Description**  
The inventory management module will track stock levels and provide alerts when stock is low.

**3.4.2 Functional Requirements**

* **REQ-4.1**: The system shall update stock levels automatically after each transaction.
* **REQ-4.2**: The system shall generate alerts for products with stock levels below a predefined threshold.

**3.5 Report Generation**  
**3.5.1 Description**  
The report generation module will provide sales and inventory reports for analysis.

**3.5.2 Functional Requirements**

* **REQ-5.1**: The system shall generate daily, weekly, and monthly sales reports.
* **REQ-5.2**: The system shall generate inventory status reports.
* **REQ-5.3**: The system shall allow reports to be exported in PDF and Excel formats.

# **External Interface Requirements**

**4.1 User Interfaces**

* The system will have a GUI developed using Java Swing or JavaFX.
* The main interface will include menus for navigating between different modules (e.g., Billing, Product Management, Reports).
* Forms will be used for data entry, with validation for mandatory fields.

**4.2 Hardware Interfaces**

* The system will interface with standard input devices (keyboard, mouse) and output devices (monitor, printer).

**4.3 Software Interfaces**

* The system will interface with a MySQL database for data storage and retrieval.
* The system will generate reports that can be exported to third-party applications like Excel and PDF viewers.

**4.4 Communications Interfaces**

* The system will communicate with the database over TCP/IP if deployed in a client-server environment.

# **Other Non-Functional Requirements**

**5.1 Performance Requirements**

* The system should be able to handle up to 10,000 products and 1,000 transactions per day without performance degradation.
* The system should load the product list within 3 seconds for up to 1,000 items.

**5.2 Safety Requirements**

* The system should ensure data integrity by using transactions to prevent data loss in case of a failure during data operations.
* Regular backups should be scheduled to prevent data loss.

**5.3 Security Requirements**

* User passwords should be stored in the database using encryption.
* The system should enforce role-based access control to restrict unauthorized access to certain features.

**5.4 Software Quality Attributes**

* **Usability**: The system should have an intuitive interface, making it easy for users to perform tasks with minimal training.
* **Reliability**: The system should have an uptime of 99.9% and handle errors gracefully without crashing.
* **Maintainability**: The system’s codebase should be modular and well-documented to facilitate easy updates and maintenance.

**5.5 Business Rules**

* All products must have a unique identifier (Product ID).
* Only Admin users can delete products or modify user roles.
* Discounts applied during billing must not exceed 50% of the total bill.

# **Other Requirements**

**6.1 Database Backup**

* The system should automatically back up the database daily and store the backup in a secure location.

**6.2 System Documentation**

* The system should include a user manual and technical documentation covering installation, configuration, and troubleshooting.

**6.3 Training**

* Training sessions should be provided for store staff to familiarize them with the system's features

# **Requirements Traceability Matrix (RTM)**

The RTM ensures that all requirements are covered by design, development, and testing activities. Below is a simplified RTM example for the Billing system:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Requirement ID | Requirement Description | Source | Design Document | Development | Test Case ID | Status | Comments |
| R1 | The system must support multiple payment methods. | Business Analyst | Design Doc V1 | Dev Team A | TC01 | In Progress |  |
| R2 | The system should calculate and apply discounts. | Business Analyst | Design Doc V1 | Dev Team B | TC02 | Completed |  |
| R3 | The system must generate detailed receipts. | Business Analyst | Design Doc V2 | Dev Team C | TC03 | Not Started |  |
| R4 | The system should update inventory in real-time. | Business Analyst | Design Doc V3 | Dev Team A | TC04 | In Progress |  |
| R5 | The system must provide sales reporting. | Business Analyst | Design Doc V2 | Dev Team B |  |  |  |

## **3. Project Plan**

### **Understanding the Project**

 **Objective:** Develop a robust and scalable billing system for retail environments.

The system will manage customer registrations, item inventories, generate PDF bills, and ensure secure transactions.

 **Stakeholders:** Retail store owners, cashiers, customers, system administrators, and the development team.

###  Project Goals:

* Streamline the billing process.
* Enhance inventory and transaction management.
* Provide secure, real-time updates on customer data and inventory.
* Ensure scalable and multi-user access with robust authentication.

 **Scope:** Includes customer management, item management, real-time inventory tracking, billing, report generation, and secure data handling with encryption.

### **Tools Used for this Project**

* Java Swing: For front-end GUI.
* SQL (MySQL): Backend for managing customer and inventory data.
* PDF Libraries: For generating bills.
* Agile Tools: JIRA or Trello for sprint planning and task tracking.  Git: Version control system.

###  Develop:

* Custom modules for customer, item, billing, and reporting functionalities.
* Real-time inventory updates.
* Secure authentication and PDF bill generation.

###  Reuse:

* Pre-existing libraries for PDF generation.
* Standard authentication libraries.

 **Buy**:

* Optional third-party tools for analytics if custom reporting is not sufficient.

**Work Breakdown Structure**

###  Team Structure:

* Product Owner: Responsible for requirement prioritization and stakeholder communication.
* Scrum Master/Project Manager: Facilitates the Agile process, tracks progress.

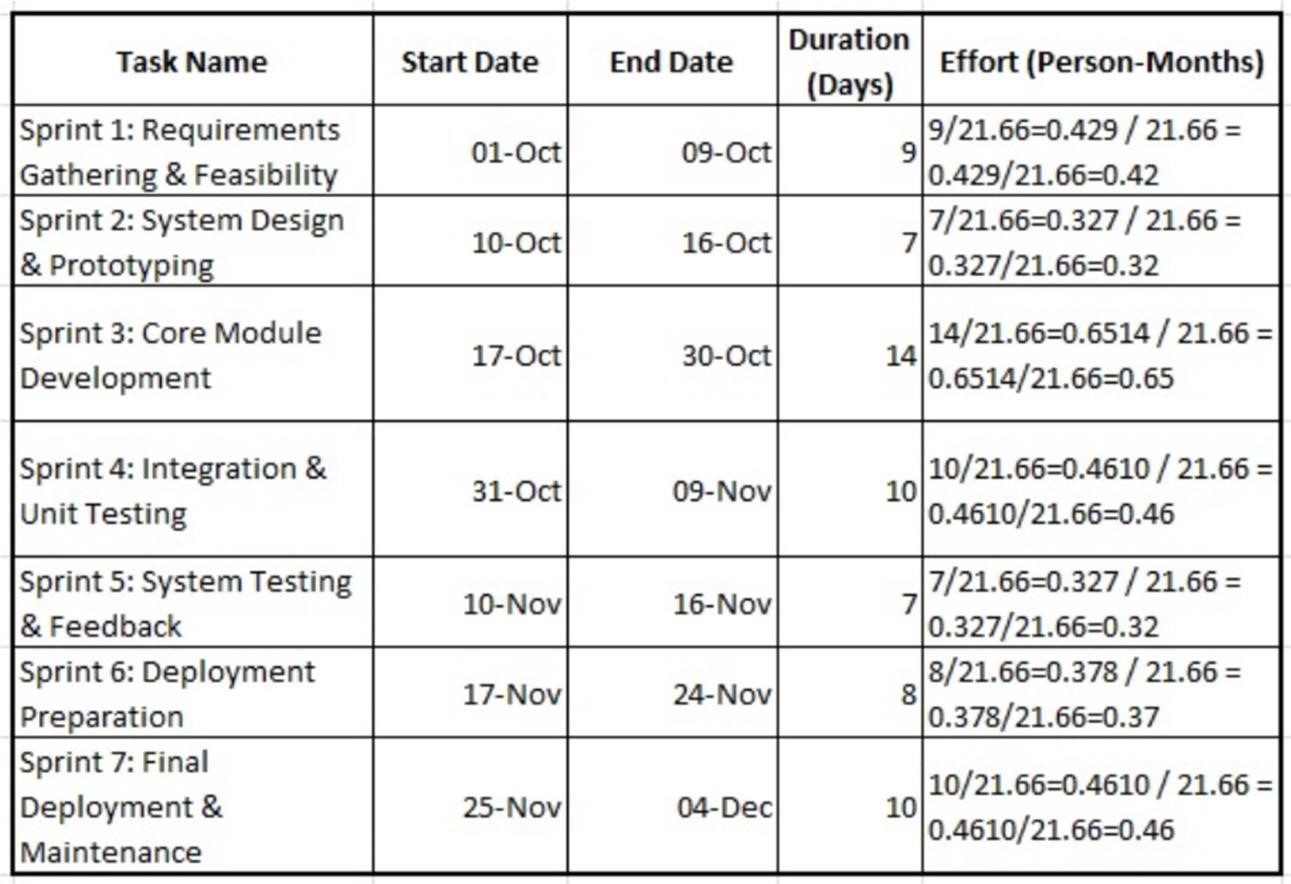
###  Development Team:

* Front-end developer (Java Swing).
* Back-end developer (SQL, PDF generation).
* Database Administrator (DBA).
* Testers for unit, integration, and system testing.
* Business Analyst: Collects and refines requirements.

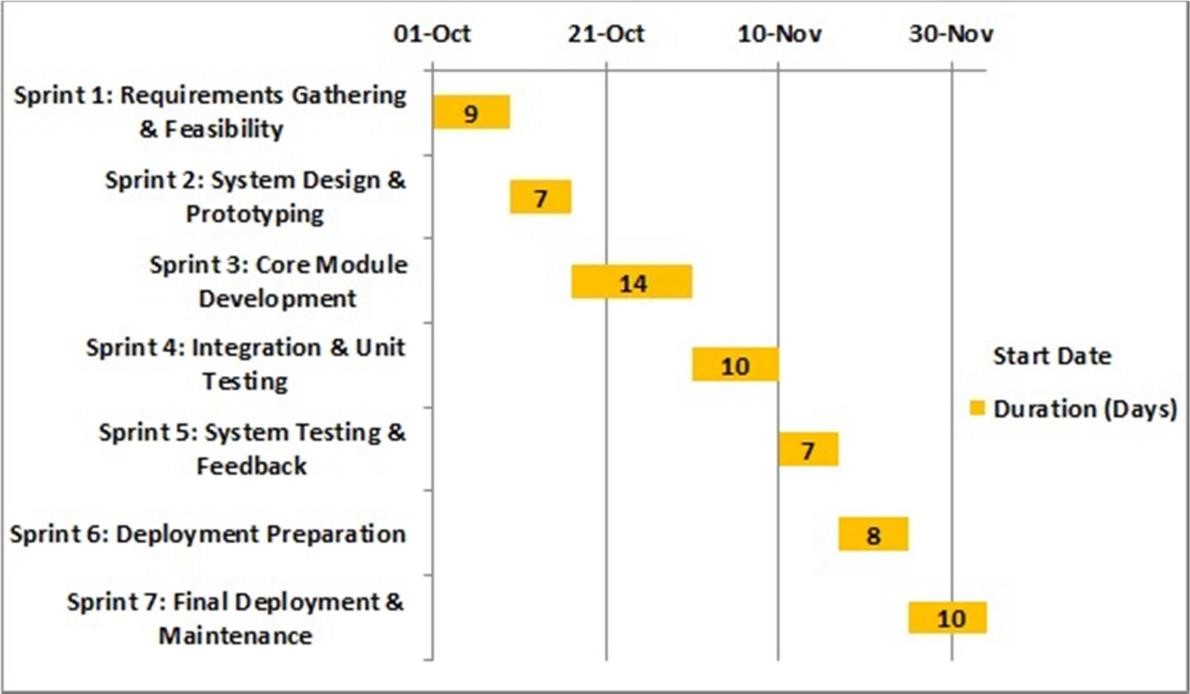
###  Responsibilities:

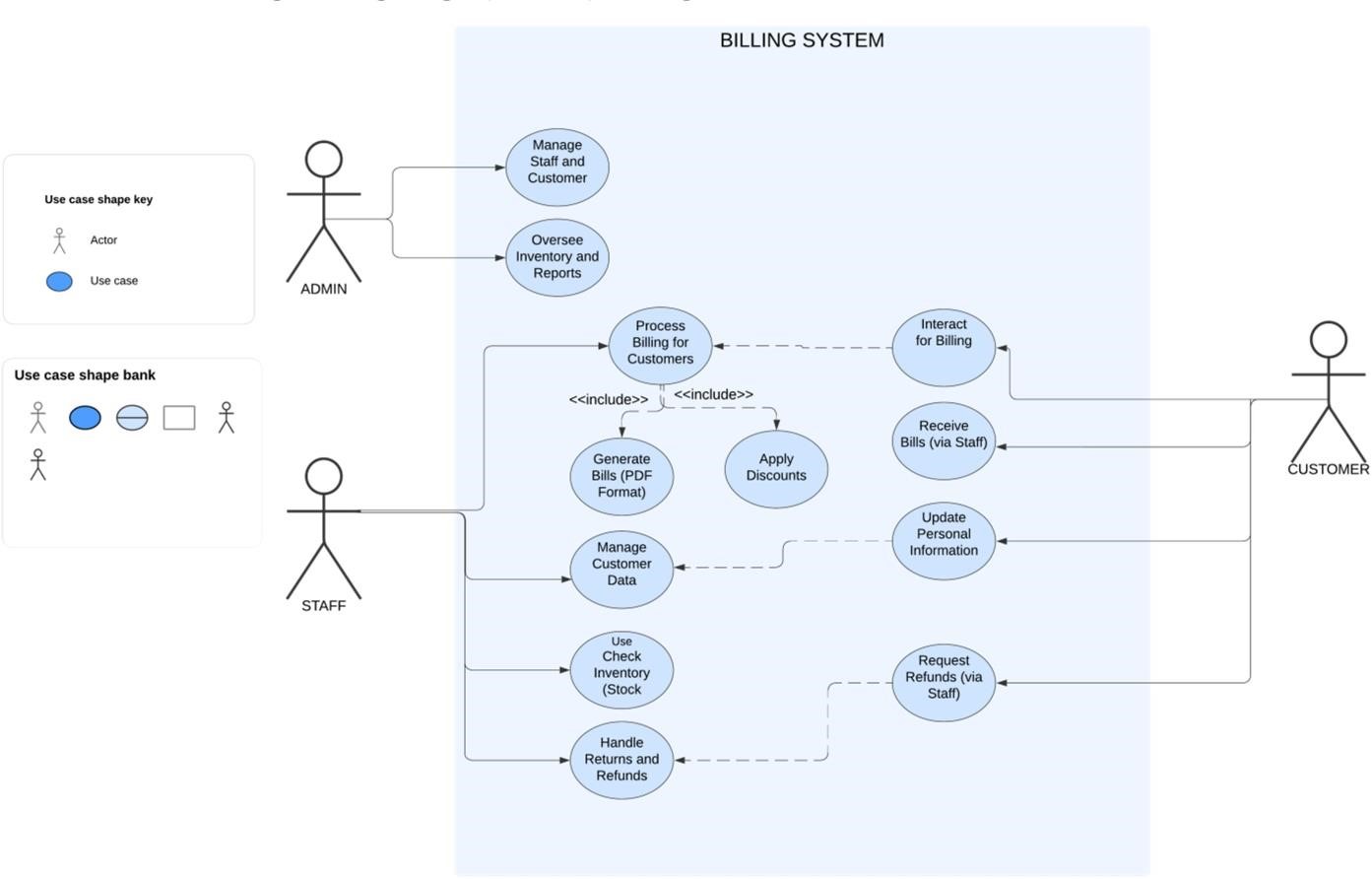
* Admin: Access to all modules (customer, inventory, billing, reports).
* Staff: Access to customer registration, billing, and stock check.
* Developers: Responsible for system development and integration.

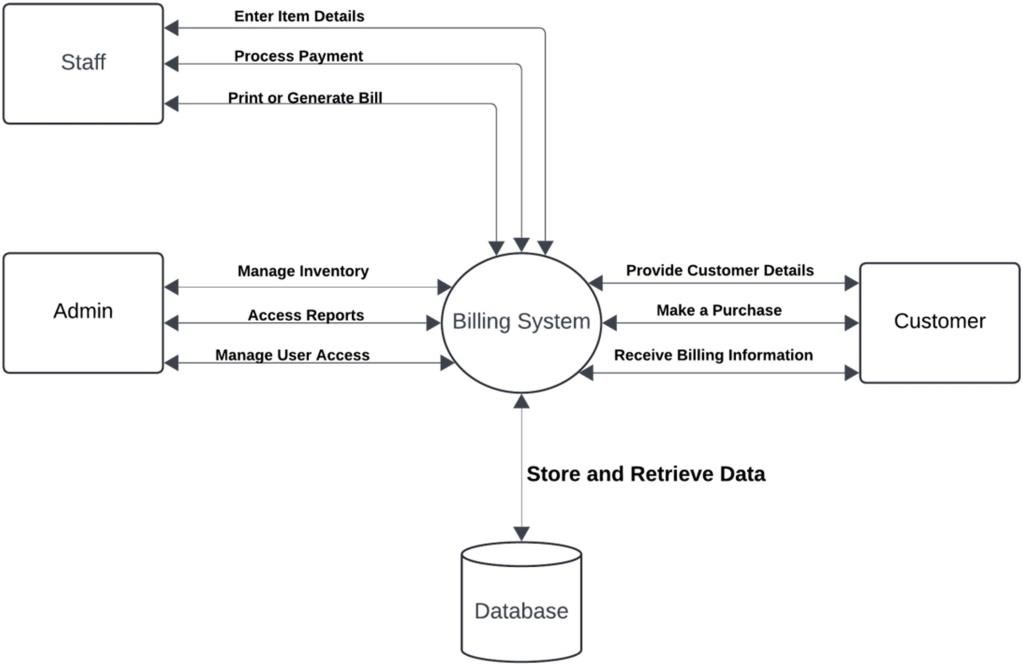
###  Effort Estimation:

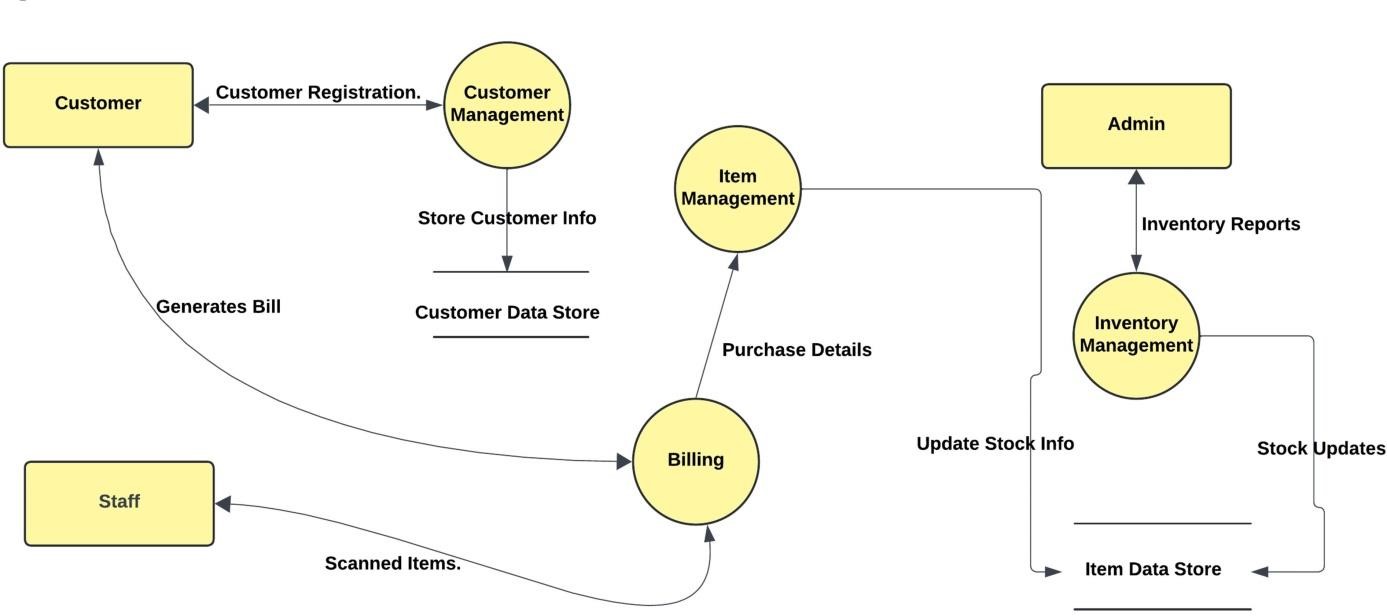
* Use Agile story points for task effort estimation.
* Assign resources based on expertise and task complexity

### **Gantt Chart**

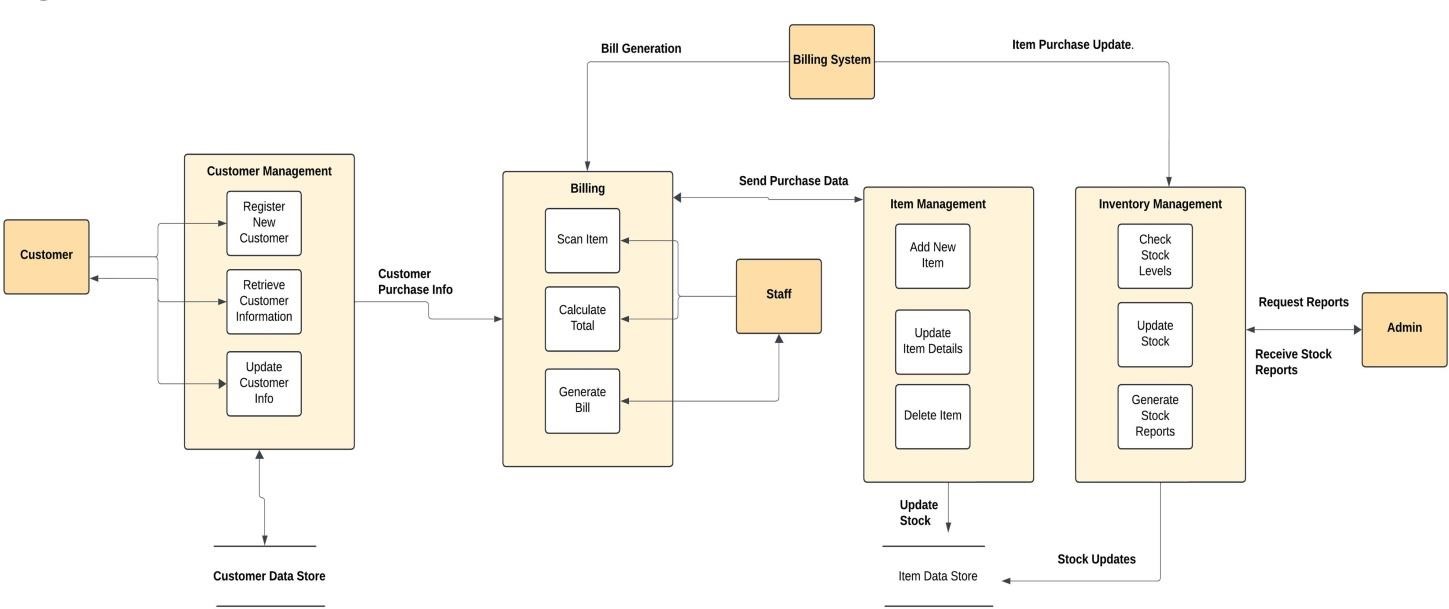


**Data Flow Diagram Logic**





### Logic 2



**TEST PLAN**

1. **Introduction**

**Project**: Billing System for Department Store

**Purpose**: The purpose of this testing document is to ensure that the Billing System for the department store meets all specified functional requirements, providing an accurate and efficient solution for billing, inventory management, and user operations. By conducting a series of structured manual tests, this document aims to identify and address potential issues early in the implementation phase, reducing errors and ensuring smooth operational performance. The ultimate goal is to enhance customer satisfaction and streamline daily processes for store staff.

**Scope**: This testing plan covers all critical modules and workflows within the billing system, ensuring seamless integration and robust functionality across the following areas:

* **Billing Module**: Includes functionalities like item scanning, cart management, discount application, tax calculations, payment processing, and receipt generation.
* **Inventory Management Module**: Manages inventory levels, ensures real-time updates for stock quantities post-purchase, and provides alerts for items with low stock.
* **User Management Module**: Manages customer and staff accounts, supports role-based access control, and ensures secure handling of customer data.

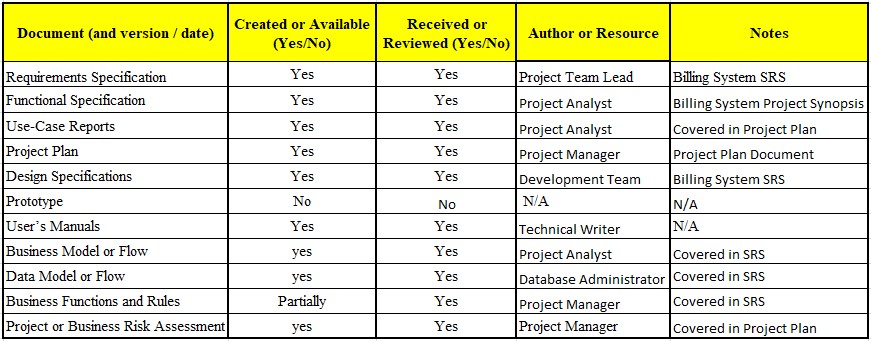
**Each module will be tested through:**

* **Unit Testing**: To validate individual functionalities (e.g., scanning items, applying discounts, calculating tax) and ensure they perform as expected under isolated conditions.
* **Integration Testing**: To test the interactions between different modules, such as linking the cart, inventory, and checkout processes to ensure smooth transitions across workflows.
* **System Testing**: To verify the entire billing workflow, ensuring that all components work together from start to finish, including edge cases and error-handling scenarios.

**Additional Objectives:**

* **Data Accuracy**: Ensures the correct calculation of item prices, taxes, and discounts, and verifies accurate updates to inventory and financial records after transactions.
* **Usability**: Confirms that the interface is user-friendly for both customers and store staff, with clear prompts, minimal errors, and ease of navigation through the billing and checkout process.
* **Security and Compliance**: Validates that the system complies with data security standards, protects customer data, and restricts access based on user roles.
* **Reliability and Performance**: Assesses system stability under varying loads, especially during peak hours, and ensures quick response times for scanning and processing payments.

**Expected Outcome**: By the end of this testing process, the billing system should operate reliably and efficiently, providing an intuitive user experience with accurate transaction handling, secure data management, and smooth integration across all modules. This testing process is designed to support the successful deployment of the billing system in a real-world department store environment, enhancing both customer service and operational efficiency.



**2. Test Case Template**

**1. Test Case: UT\_01**

|  |  |
| --- | --- |
| Test Case ID | UT\_01 |
| Name of the Module | Billing |
| Test Case Description | Verify the calculation of the total bill with multiple items. |
| Pre-conditions | System is connected to the inventory database. |
| Test Steps | 1. Add Item A (Price: $50, Quantity: 2). 2. Add Item B (Price: $30, Quantity: 1). 3. Click 'Calculate Total'. |
| Test Data | Item A, Item B, Quantities. |
| Expected Results | Total amount should be $130. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**2. Test Case: UT\_02**

|  |  |
| --- | --- |
| Test Case ID | UT\_02 |
| Name of the Module | User Management |
| Test Case Description | Verify login functionality with valid credentials. |
| Pre-conditions | User account is active. |
| Test Steps | 1. Navigate to the login page. 2. Enter valid username and password. 3. Click 'Login'. |
| Test Data | Username: cashier1, Password: pass123. |
| Expected Results | User should be logged in and redirected to the dashboard. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**3. Test Case: IT\_01**

|  |  |
| --- | --- |
| Test Case ID | IT\_01 |
| Name of the Module | Inventory Management |
| Test Case Description | Verify stock update after a sale. |
| Pre-conditions | Product has available stock. |
| Test Steps | 1. Sell 5 units of Product X. 2. Verify stock level in database. |
| Test Data | Product X, Initial Stock: 20 units, Quantity Sold: 5 units. |
| Expected Results | Stock should decrease to 15 units. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**4.** **Test Case: UT\_03**

|  |  |
| --- | --- |
| Test Case ID | UT\_03 |
| Name of the Module | Billing |
| Test Case Description | Verify behaviour when an invalid discount code is applied. |
| Pre-conditions | Discount feature is enabled. |
| Test Steps | 1. Add items to cart. 2. Apply discount code 'INVALIDCODE'. 3. Click 'Apply'. |
| Test Data | Discount Code: INVALIDCODE. |
| Expected Results | Error message "Invalid discount code" should be displayed. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**5.** **Test Case: ST\_01**

|  |  |
| --- | --- |
| Test Case ID | ST\_01 |
| Name of the Module | Reports |
| Test Case Description | Verify daily sales report generation. |
| Pre-conditions | Sales data is available for the day. |
| Test Steps | 1. Go to 'Reports' section. 2. Select 'Daily Sales Report'. 3. Click 'Generate'. |
| Test Data | Date: 2024-10-19. |
| Expected Results | Report should include all transactions for the selected date. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**6.** **Test Case: IT\_02**

|  |  |
| --- | --- |
| Test Case ID | IT\_02 |
| Name of the Module | Payment Processing |
| Test Case Description | Verify successful transaction with valid card details. |
| Pre-conditions | Payment gateway is accessible. |
| Test Steps | 1. Add items to cart. 2. Select 'Card Payment'. 3. Enter valid card details and submit. |
| Test Data | Card Number: 1234 5678 9123 4567, Expiry: 12/25, CVV: 123. |
| Expected Results | Transaction should be successful and receipt generated. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**7.** **Test Case: UT\_04**

|  |  |
| --- | --- |
| Test Case ID | UT\_04 |
| Name of the Module | User Management |
| Test Case Description | Verify failed login with invalid credentials. |
| Pre-conditions | User account exists. |
| Test Steps | 1. Navigate to login page. 2. Enter invalid username/password. 3. Click 'Login'. |
| Test Data | Username: wrongUser, Password: wrongPass. |
| Expected Results | Error message "Invalid username or password" should be displayed. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**8.** **Test Case: ST\_02**

|  |  |
| --- | --- |
| Test Case ID | ST\_02 |
| Name of the Module | Billing |
| Test Case Description | Verify invoice generation with detailed breakdown. |
| Pre-conditions | System is operational, Items are in cart. |
| Test Steps | 1. Add items to cart. 2. Click 'Generate Invoice'. 3. Check invoice details. |
| Test Data | Item A, Item B. |
| Expected Results | Invoice should include item details, quantity, price, discounts, and total amount. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**10.** **Test Case: UT\_05**

|  |  |
| --- | --- |
| Test Case ID | UT\_05 |
| Name of the Module | Inventory Management |
| Test Case Description | Verify alert for low stock level. |
| Pre-conditions | Stock threshold is set. |
| Test Steps | 1. Reduce stock of Product Y below threshold. 2. Check for alert. |
| Test Data | Product Y, Threshold: 5 units. |
| Expected Results | Alert "Low stock for Product Y" should be displayed. |
| Actual Results | (To be filled after testing) |
| Test Result | (Pass/Fail) |

**11. Test Case: IT\_03**

|  |  |
| --- | --- |
| Test Case ID | IT\_03 |
| Name of the Module | **Payment Processing** |
| Test Case Description | **Verify transaction failure with expired card.** |
| Pre-conditions | **Payment gateway is accessible.** |
| Test Steps | **1. Add items to cart. 2. Select 'Card Payment'. 3. Enter expired card details and submit.** |
| Test Data | **Card Number: 9876 5432 1098 7654, Expiry: 12/22, CVV: 321.** |
| Expected Results | **Transaction should be declined with message "Card expired."** |
| Actual Results | **(To be filled after testing)** |
| Test Result | **(Pass/Fail)** |

**Conclusion**

The testing phase for the Billing System for the department store is crucial in verifying that all functionalities work as intended and that the system meets the required standards for performance, accuracy, and usability. Through unit, integration, and system testing, we have systematically validated each core module, including Billing, Inventory Management, and User Management. This testing ensures that all components work cohesively to provide a seamless, secure, and efficient experience for users.

In summary, this testing process has significantly enhanced the system’s quality and readiness for use, ultimately contributing to improved operational efficiency, customer satisfaction, and overall business performance. Future maintenance and periodic testing are recommended to adapt the system to evolving needs and to ensure continued optimal performance.

**Test Deliverables :**

**The following deliverables will be produced as part of the testing process:**

1. **Software Requirements Specification (SRS)**: The document outlining the functional and non-functional requirements of the Billing System, which will serve as the baseline for the tests.
2. **Project Synopsis**: A summary of the project's objectives, scope, and expected outcomes, used to align testing with the overall project goals.
3. **Project Plan and Architecture Design**: Documentation detailing the project timeline, roles, responsibilities, and the system’s high-level design. This will be used to validate the system’s architecture against its intended design.
4. **Test Cases**: Documentation of all test scenarios, covering both functional and nonfunctional requirements (such as performance), and including detailed steps for executing each test.
5. **Test Summary Report**: A comprehensive report summarizing the results of all tests performed, identifying any issues, and providing recommendations for further action.

**Environmental Needs**

* 1. **Hardware:**

|  |  |
| --- | --- |
| **Machine** | **Configuration** |
| Server | 4 CPU, 16GB RAM, 500GB HDD |
| Client | POS Terminal, 8GB RAM |

* 1. ***Software:***

|  |  |  |
| --- | --- | --- |
| **Machine** | **OS** | **Software** |
| Server | Linux/Windows | MySQL, Java Runtime Environment |
| Client | Windows | Java Swing application, Printer |

* 1. ***Tools:***
* **Version Control**: Git
* **Task Tracking**: Jira or Trello

1. **Staffing**

***5.1 Responsibilities:***

* **Test Plan**: Your Name
* **Test Case**: Test Lead
* **Manual Testing**: Test Team
* **Automated Testing**: Automation Engineer

***5.2 Training:***

* Training will be provided on test automation tools and specific use cases for testing the billing system.

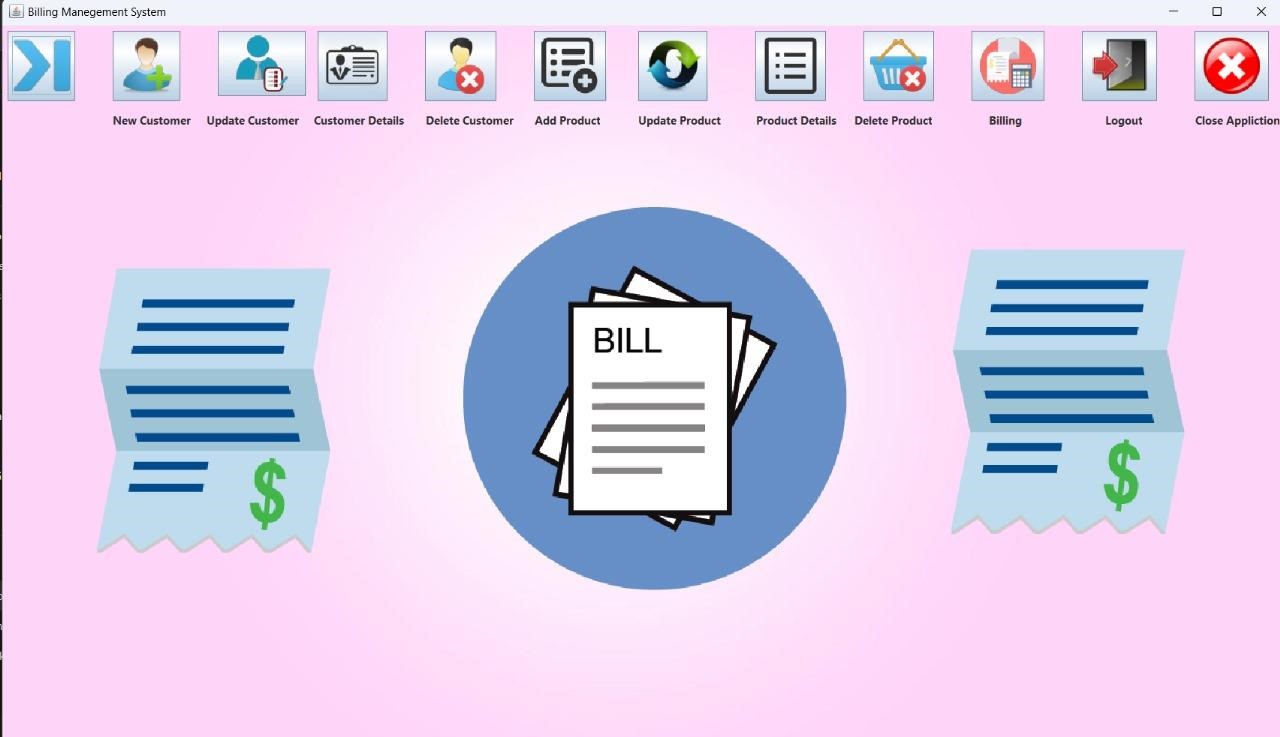
**6. Dependencies/Risks**

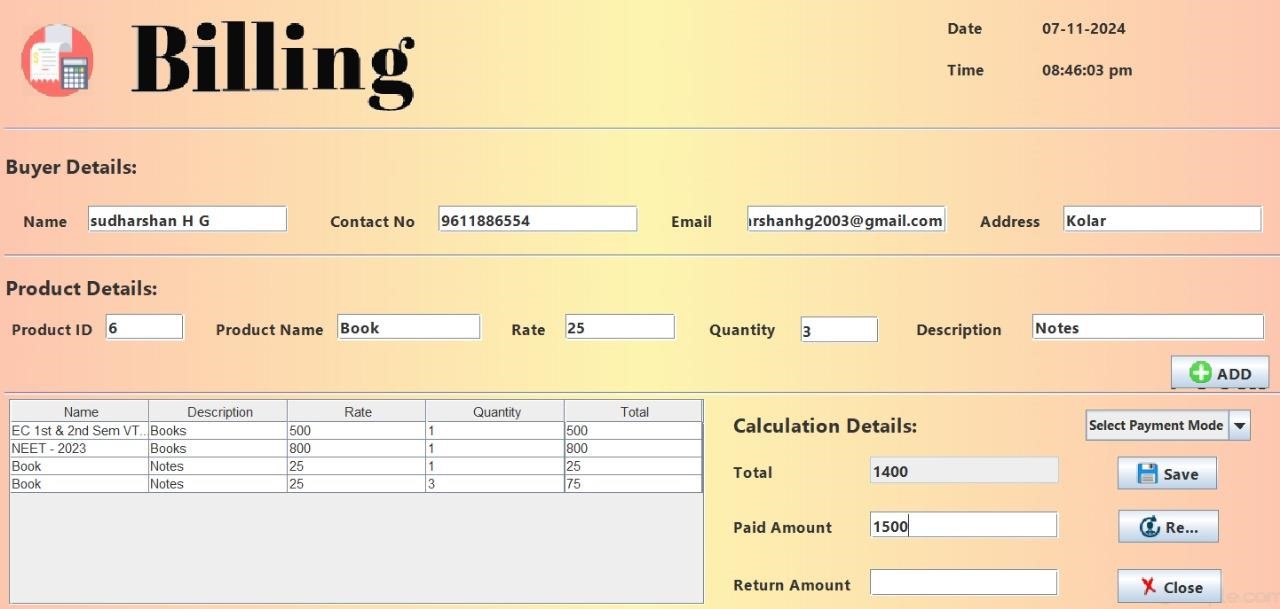
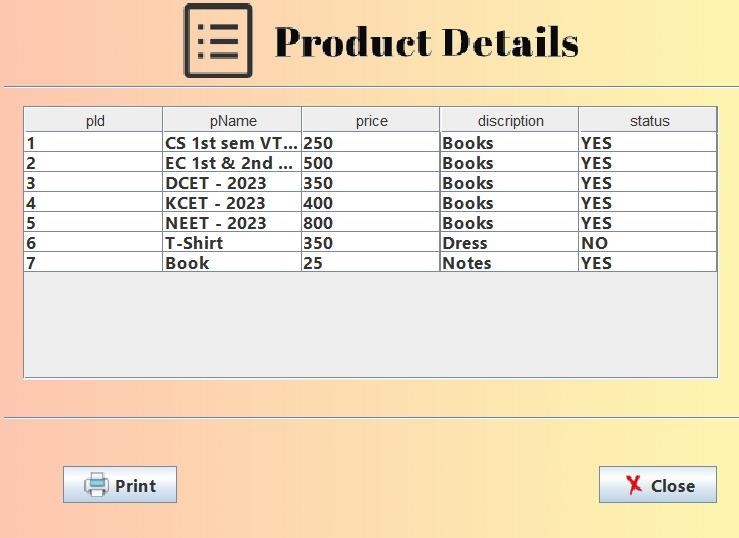
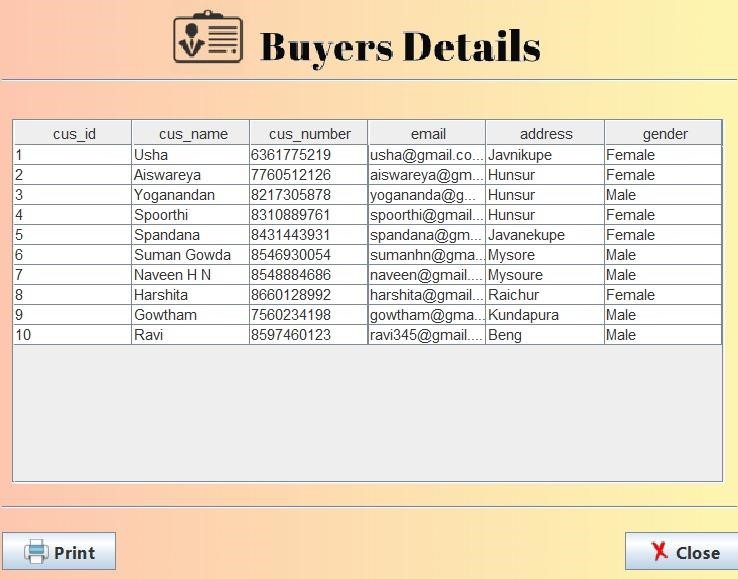
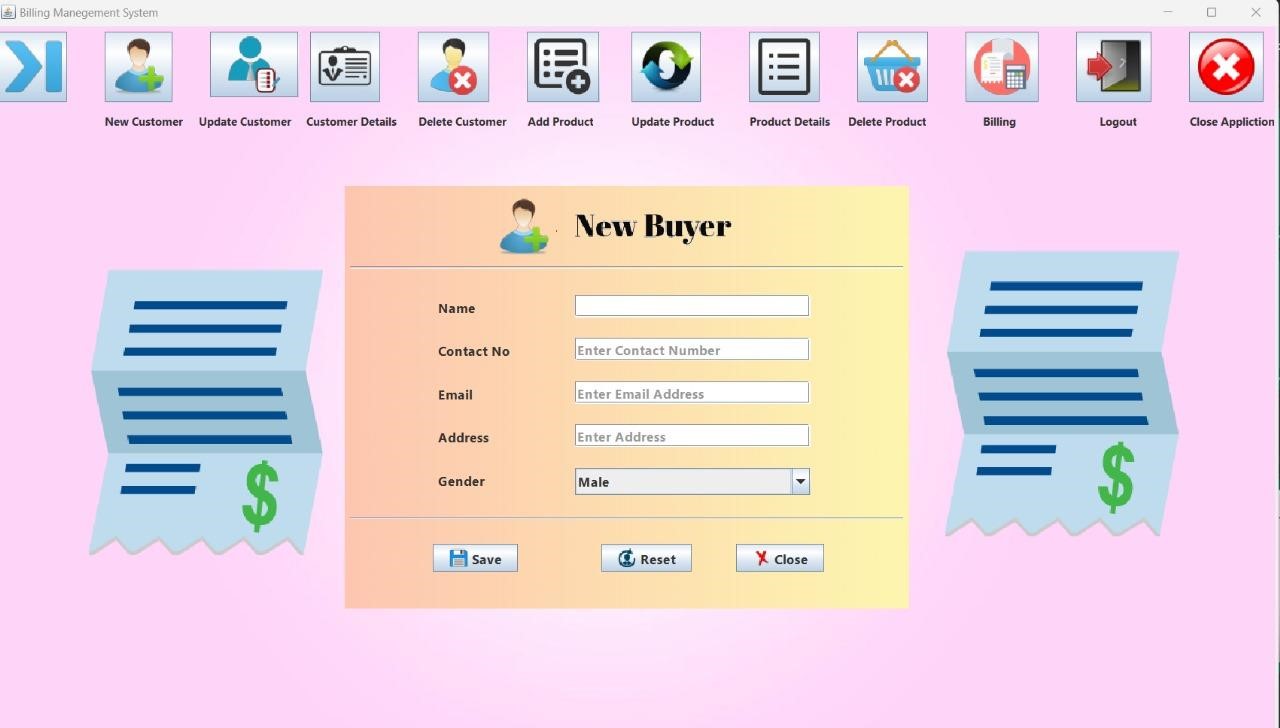
* 1. **Dependencies**:
* System components must be integrated on time for testing.
* Database backups must be scheduled regularly.
  1. **Risks**:
* Performance issues with high loads.
* Security vulnerabilities in data handling.

### **7. Schedule and Milestones**

|  |  |
| --- | --- |
| **Testing Type** | **Schedule Date** |
| Unit Testing | October 10, 2024 |
| Integration Testing | October 15, 2024 |
| System Testing | October 20, 2024 |
| Acceptance Testing | October 27, 2024 |

**SCREENSHOT OF OUTPUT**





**DATABASE OUTPUT SCREENSHOT**

