University of Regina | Go far, together.

ENSE 375 – Software Testing and Validation

Inventory Management

Drashti Soni (200502122)

Vishva Shah (200495565)

Premal Patel (200499806)

Table of Contents

1 Introduction 6
2 Design Problem 7
2.1 Problem Definition 7
2.2 Design Requirements 7
2.2.1 Functions 7
2.2.2 Objectives 7
2.2.3 Constraints 7
3 Solution 9
3.1 Solution 1 9
3.2 Solution 2 9
3.3 Final Solution 9
3.3.1 Components 14
3.3.2 Features 15
3.3.3 Environmental, Societal, Safety, and Economic Considerations 16
3.3.4 Test Cases and results 17
3.3.5 Limitations 17
4 Team Work 18
4.1 Meeting 1 18
4.2 Meeting 2 18
4.3 Meeting 3 19
4.4 Meeting 4 22
5 Project Management 23
6 Conclusion and Future Work 24
7 References 25

List of Figures

Figure 1 Block Diagram..... 14

List of Tables

Table 1 Table for Comparison 10
Table 2 Features 15
Table 3 Meeting 1 18
Table 4 Meeting 2 18
Table 5 Meeting 3 19
Table 6 Meeting 4 19
Table 7 Meeting 5 19
Table 8 Meeting 6 20
Table 9 Meeting 7 20
Table 10 Meeting 8 21
Table 11 Meeting 9 21
Table 12 Meeting 10 21
Table 13 Meeting 11 22
Table 14 Gantt Chart 23

1 Introduction

In the contemporary business landscape, efficient inventory management is paramount for the success and sustainability of enterprises across various sectors. The complexity of tracking item transactions, including additions, sales, and insertions, necessitates a robust system that supports Create, Read, Update, and Delete (CRUD) operations. This project aims to develop an advanced inventory management system that not only tracks these transactions but also provides comprehensive features for item listing, filtering, and viewing transaction history for selected dates.[1]

The project is structured around clear design requirements, focusing on functionalities, objectives, and constraints. The system is designed to be user-friendly, scalable, reliable, maintainable, and efficient, addressing the needs of businesses of all sizes. Economic factors, reliability, sustainability, and ethical considerations are integrated into the design process, ensuring a holistic approach to inventory management.

2 Design Problem

2.1 Problem Definition

This project involves developing an inventory management system to track item transactions including adding, selling, and inserting items. It supports CRUD operations for managing items with details such as Item ID, Name, Cost, Price, and Quantity. The system features item listing and filtering, as well as viewing transaction history for selected dates, enabling efficient inventory tracking and management.

2.2 Design Requirements

This section has the following three subsections:

2.2.1 Functions

- Add Items: Allow users to add new items to the inventory with details.
- View Items: Display a list of all items in the inventory, allowing users to see item details.
- Edit Items: Enable users to update existing item details.
- Delete Items: Provide functionality to remove items from the inventory.
- Filter Items: Allow users to filter the item list based on criteria.
- Sell Items: Enable users to mark items as sold, updating the inventory accordingly.

2.2.2 Objectives

- User-Friendly: The system should be easy to use with a clean and intuitive user interface.
- Scalable: The system should be able to handle an increasing number of items and users without performance degradation.
- Reliable: The system should be highly reliable, with minimal downtime and robust error handling.
- Maintainable: The codebase should be maintainable, with clear documentation and modular design to facilitate future updates and maintenance.
- Efficient: The system should efficiently handle operations, minimizing load times and resource usage.

2.2.3 Constraints

1. **Economic factors**
    - Cost Efficiency: Utilizing pen-source technologies like Angular, Spring Boot, MySQL, and Docker can significantly reduce costs compare to using software.
    - By choosing modern frameworks and tools, we can reduce development time and maintenance costs due to the availability of documentation and community support.

2. **Reliability**

- Test cases ensure that the application functions as expected, catching bugs and issues early in the development process. This leads to a more reliable and stable application.

3. Sustainability and Environmental factors

- Efficient Resource Usage: Docker can help optimize resource usage by running multiple service on the same hardware, reducing the overall environmental footprint.

4. Ethics

- Design the system to be accessible to all users.
- Be transparent about how inventory data is managed, who has access, and for what purposes it is used.[2][3][4]

3 Solution

3.1 Solution 1

Inventory Management with Angular and Spring Boot

- **Description:** This basic solution would leverage Angular for a user-friendly interface and Spring Boot for robust backend functionality. Users could add items with basic details (ID, name, cost, price, quantity) using Angular forms. Spring Boot would handle data persistence through Spring Data JPA and expose REST APIs for CRUD operations.
- **Limitations:** This solution lacks features like:
    - Transaction tracking (selling, inserting)
    - Item history view
    - Filtering by ID or name
    - While functional, it wouldn't provide a comprehensive inventory management experience.

3.2 Solution 2

Inventory Management with Basic Transactions and Angular Filtering

- **Description:** Building on Solution 1, this approach would introduce functionalities for basic transactions and leverage Angular for item filtering:
    - Selling items: Users could sell items at set prices or custom prices through the Angular interface. Spring Boot would handle the sale logic and update inventory levels.
    - Inserting items: This would allow recording returned items or adding previously missing stock, again using Angular forms and Spring Boot APIs.
    - Filtering by ID or name: We would implement search functionality using keywords to filter the displayed item list. This could utilize Spring Boot's APIs for efficient data retrieval based on filter criteria.
- **Limitations:** While improved, this solution still lacks:
    - Item history view for detailed transaction tracking
    - This might not be sufficient for businesses needing advanced features like purchase orders or low-stock alerts.[6][7]

3.3 Final Solution

The final solution is better than other solutions due to its comprehensive feature set, enhanced user experience, and robust backend. Below is a detailed comparison and explanation of why this solution was selected.

- **Table for Comparison**

Table 1 Table for Comparison

Feature	Solution 1	Solution 2	Final Solution
Item Management	Basic CRUD	Basic CRUD with selling/inserting	Comprehensive CRUD, additional details, supplier, category
Transactions	None	Selling, Inserting	Purchase orders, returns, transfers, history
Item History	None	None	Detailed history with transaction types, dates, quantities, prices
Filtering	None	By ID or name	Multiple criteria, advanced search
User Interface	Angular	Angular	Enhanced UI with data visualization
Backend	Spring Boot	Spring Boot	Spring Boot with advanced features, potential message queue
Additional Features	None	None	Inventory storage, picking, packing, data visualization
Deployment	None	None	Docker, Docker Compose
Development Approach	Basic development	Incremental improvements	Iterative development, comprehensive testing

- **Reasons for Selecting the Final Solution**

1. **Comprehensive Functionalities**:
    - **Item Management**: The final solution expands on basic CRUD operations by including additional item details such as description, category, and supplier information.
    - **Transactions**: It handles various transaction types (purchase orders, returns, transfers) and provides detailed transaction history, which is crucial for businesses needing advanced inventory tracking.
    - **Item History**: Unlike the previous solutions, this solution implements a detailed item history view, enabling better tracking of inventory changes over time.
    - **Filtering**: Enhanced filtering options allow users to search and filter items based on multiple criteria, improving usability and efficiency.
2. **User Experience**:

- **User Interface**: The final solution maintains a user-friendly Angular interface but adds data visualization capabilities, providing users with better insights and analysis tools.
- **Enhanced UI**: The inclusion of Angular Material improves the overall look and feel, making the interface more intuitive and easier to navigate.

3. **Robust Backend**:
    - **Spring Boot**: Utilizes a robust backend with Spring Boot, ensuring reliable performance and scalability.
    - **Data Persistence**: Through Spring Data JPA, the solution ensures efficient and reliable data storage and retrieval.
    - **Potential Integration with Message Queue**: The final solution can integrate with a message queue, enhancing the system's ability to handle asynchronous tasks and improve performance.
4. **Development and Deployment**:
    - **Iterative Development**: The solution follows an iterative development approach, allowing for incremental feature additions and continuous improvement.
    - **Testing**: Comprehensive testing (unit, integration, and end-to-end tests) ensures high-quality and reliable software.
    - **Deployment**: Utilizes Docker and Docker Compose for consistent and efficient deployment, simplifying the setup and scaling of the application.

- **Detailed Description of the Final Solution**
- **Core Functionalities**

1. **Item Management**:
    - **CRUD Operations**: Create, read, update, and delete operations for items, including detailed information like ID, name, description, cost, price, quantity, category, and supplier.
2. **Transactions**:
    - **Purchase Orders**: Manage purchase orders including details like supplier, item, quantity, and price.
    - **Returns**: Record and manage returned items.
    - **Transfers**: Handle transfers of items between different locations.
    - **History**: Maintain a detailed history of all transactions with type, date, quantity, and price.

3. **Item History**:
    - **Transaction History**: View detailed transaction history for each item.
4. **Filtering**:
    - **Advanced Filtering**: Filter items based on multiple criteria such as category, price range, and quantity.
5. **User Interface**:
    - **Data Visualization**: Implement data visualization features to provide insights into inventory status and trends.
    - **Enhanced UI**: Use Angular Material for a modern, responsive, and intuitive user interface.

- **Backend**

1. **Spring Boot**:
    - **REST APIs**: Expose robust REST APIs for all functionalities.
    - **Data Persistence**: Use Spring Data JPA for efficient data management.
    - **Potential Message Queue**: Integrate with a message queue for handling asynchronous tasks.

- **Additional Features**

1. **Inventory Storage, Picking, and Packing**:
    - **Storage Management**: Manage inventory storage locations.
    - **Picking and Packing**: Streamline the picking and packing processes for orders.

- **Technology Stack**
- **Frontend**: Angular, Angular Material, TypeScript
- **Backend**: Spring Boot, Spring Data JPA
- **Database**: H2
- **Deployment**: Docker, Docker Compose
- **Development Approach**
- **Iterative Development**: Prioritize core functionalities and add features incrementally.
- **Testing**: Implement unit, integration, and end-to-end tests to ensure quality.
- **Environmental, Societal, Safety, and Economic Considerations**

1. **Environmental Considerations**:

    - **Reduced Paper Consumption**: Digitizing inventory records reduces paper usage.
    - **Optimized Inventory Levels**: Prevents overstocking, reducing production and transportation emissions.

2. **Societal Considerations**:
    - **Improved Supply Chain Efficiency**: Optimizes supply chain operations, improving customer satisfaction.
    - **Job Creation**: May create jobs for data entry, system administration, and analysis.

3. **Economic Considerations**:
    - **Return on Investment (ROI)**: Improves inventory turnover, reduces stockouts, and optimizes purchasing decisions.
    - **Economic Impact**: Contributes to economic growth by enabling efficient and competitive business operations.

- **Limitations**
- **Complexity Management**: As the system grows, managing and maintaining feature complexity can be challenging.
- **Data Accuracy**: Reliance on human input can lead to inaccuracies affecting inventory levels.
- **Technology Dependency**: System functionality heavily relies on technology, and disruptions can impact operations.
- **Ongoing Costs**: Implementation and maintenance involve ongoing costs for hardware, software, and personnel.
- **User Training**: Effective training and support are crucial for user acceptance and system utilization.
- **Test Cases**

Detailed test cases for ItemActionServiceImpl.java, ItemSummaryServiceImpl.java, and ItemServiceImpl.java cover various testing techniques like boundary value testing, equivalence class testing, decision table testing, state transition testing, and use case testing. These ensure comprehensive validation of the solution's functionality and reliability.

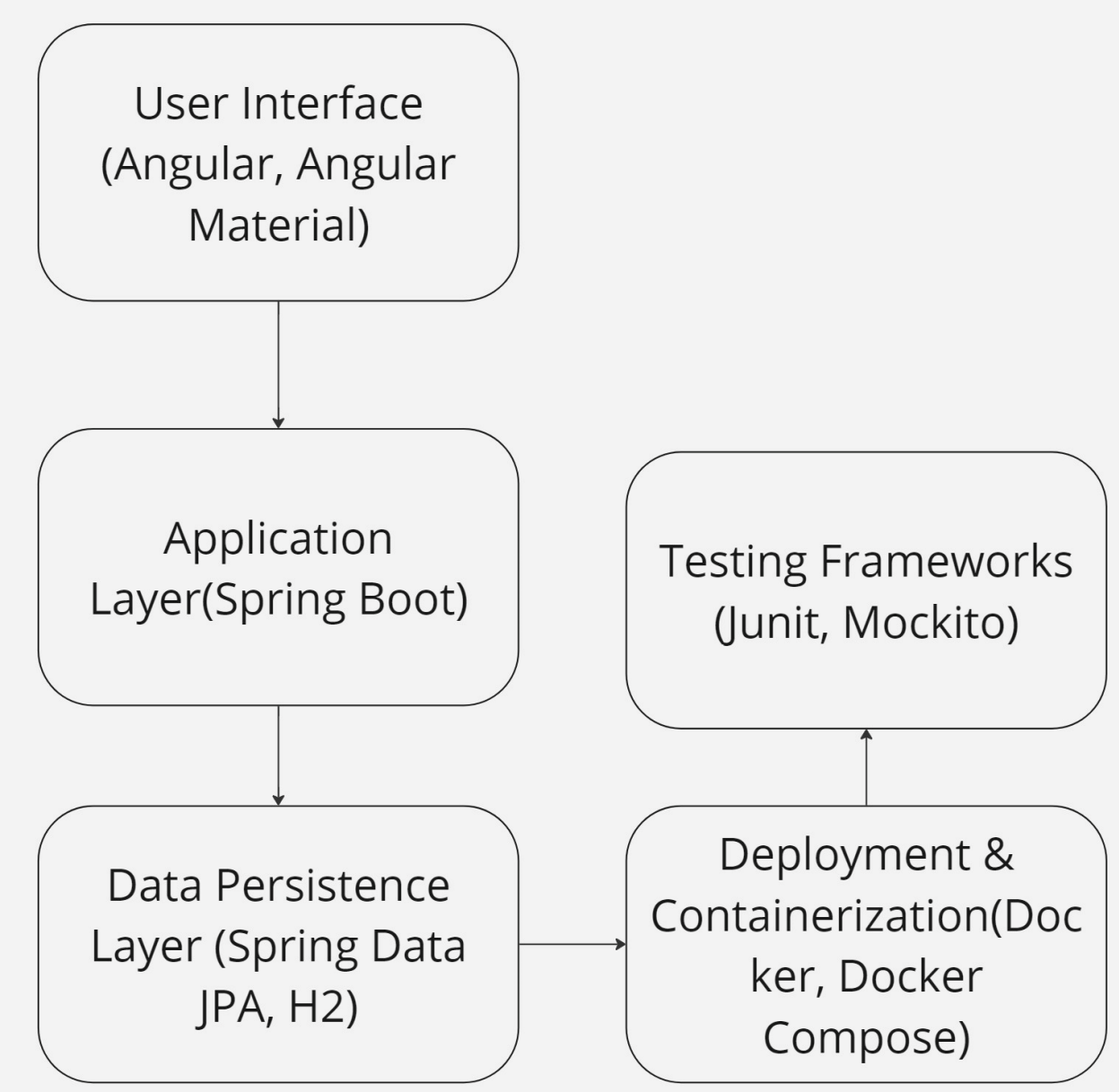

Figure 1 Block Diagram

3.3.1 Components

The final solution consists of several components, each serving a specific purpose to ensure the system's functionality, reliability, and user-friendliness. Below are the main components used in the solution:

1. **Frontend: Angular**
    - **Main Purpose**: Provides a user-friendly interface for interacting with the inventory system. Angular Material enhances the UI/UX with responsive and modern design elements.

2. **Backend: Spring Boot**
    - **Main Purpose**: Handles the business logic, data processing, and serves as the server-side application framework. It exposes REST APIs for CRUD operations and integrates with the data persistence layer.
3. **Data Persistence: Spring Data JPA**
    - **Main Purpose**: Manages the database operations and ensures efficient data storage and retrieval using ORM (Object-Relational Mapping).
4. **Database: H2**
    - **Main Purpose**: Acts as the in-memory database for development and testing purposes, providing a lightweight and fast database solution.
5. **Deployment: Docker & Docker Compose**
    - **Main Purpose**: Facilitates consistent and efficient deployment of the application across different environments, ensuring scalability and ease of setup.
6. **Data Visualization: Angular Material**
    - **Main Purpose**: Adds data visualization capabilities to the frontend, allowing users to analyze inventory data through charts and graphs.
7. **Message Queue (Potential Integration)**
    - **Main Purpose**: Handles asynchronous tasks and improves the performance and scalability of the system (optional component).
8. **Testing Frameworks: JUnit, Mockito**
    - **Main Purpose**: Ensures comprehensive testing of the application, including unit, integration, and end-to-end tests

3.3.2 Features

Table 2 Features

Feature	Description
Item Management	Comprehensive CRUD operations, including additional item details like description, category, supplier.
Transactions	Handles various transaction types such as sales, purchases, returns, and transfers between locations.
Item History	Provides a detailed record of item transactions, including date, quantity, price, and transaction type.

Feature	Description
Filtering	Advanced filtering options, allowing search by multiple criteria (e.g., category, price range, quantity).
User Interface	Enhanced user-friendly interface with Angular Material for data visualization and improved UX.
Backend Services	Robust backend services using Spring Boot, including REST APIs and data persistence with Spring Data JPA.
Data Visualization	Visualization of inventory data through charts and graphs for better analysis and decision-making.
Deployment	Consistent and efficient deployment using Docker and Docker Compose.

3.3.3 Environmental, Societal, Safety, and Economic Considerations

Environmental Considerations

- Reduced Paper Consumption: By digitizing inventory records, the solution minimizes the need for paper, leading to conservation of resources and reduction in waste.
- Optimized Inventory Levels: Effective inventory management prevents overstocking, reducing unnecessary production and transportation, which in turn lowers greenhouse gas emissions.

Societal Considerations

- Improved Supply Chain Efficiency: Accurate and timely inventory data optimize supply chain operations, reducing product shortages and enhancing customer satisfaction.
- Job Creation: Implementation of the system may create jobs in areas like data entry, system administration, and data analysis.

Economic Considerations

- Return on Investment (ROI): The system is designed to improve inventory turnover, reduce stockouts, and optimize purchasing decisions, thereby offering a clear ROI.

- Economic Impact: By enabling businesses to operate more efficiently and competitively, the system contributes to broader economic growth.

Safety Considerations

- Data Accuracy: Ensuring data accuracy through validation and error-handling mechanisms enhances the reliability of the system.

- Technology Reliability: The use of robust frameworks (Spring Boot, Angular) and comprehensive testing ensures the system's reliability and safety in operations.

3.3.4 Test Cases and results

Test Suites Designed

- **Boundary Value Testing**: Tested the limits of input values to ensure the system handles edge cases correctly.
- **Equivalence Class Testing**: Grouped inputs into equivalence classes to ensure that each class is tested at least once.
- **Decision Table Testing**: Used decision tables to test combinations of inputs and their corresponding outputs.
- **State Transition Testing**: Tested the system's behavior under different states and transitions between these states.
- **Use Case Testing**: Verified the system's functionality through real-world scenarios that represent typical user interactions.

Execution of Test Cases

- **Unit Testing**: Used JUnit and Mockito frameworks to execute unit tests, ensuring individual components function as expected.
- **Integration Testing**: Verified that different components of the system work together correctly.
- **End-to-End Testing**: Conducted end-to-end tests to ensure the system meets all functional requirements from the user's perspective.

3.3.5 Limitations

Complexity Management: As the system grows, the complexity of managing and maintaining various features and integrations can increase, potentially impacting performance and scalability.

Data Accuracy: The system relies on human input for data entry, which can lead to inaccuracies affecting inventory levels.

Technology Dependency: The system's functionality heavily relies on technology, and any disruptions or failures can impact operations.

Ongoing Costs: Implementing and maintaining the system involves ongoing costs for hardware, software, and personnel.

User Training: Effective training and support are crucial for user acceptance and optimal system utilization. Without proper training, users may not fully leverage the system's capabilities.[9][10]

4 Team Work

4.1 Meeting 1

Time: 15 th May 2024, Year, hour: 5:30 PM to hour: 8:30 PM
Agenda: Distribution of Project Tasks

Table 3 Meeting 1

Team Member	Previous Task	Completion State	Next Task
Drashti	Problem Definition and Research	100%	Identify and document business requirements.
Vishva	Design Constraints and Functionalities	80%	Define system requirements and specifications.
Premal	Objectives	90%	Conduct stakeholder interviews.

4.2 Meeting 2

Time: 20th May 2024, Year, hour: 10:00 AM to hour: 12:00 PM
Agenda: Review of Individual Progress

Table 4 Meeting 2

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	80%	Review existing inventory management systems.
Vishva	Research more on Functionalities	100%	Gather user feedback and requirements.

Premal	Basic details using Angular forms	90%	Design the system architecture.

4.3 Meeting 3

Time: 23rd May 2024, Year, hour: 2:00 PM to hour: 4:00 PM
Agenda: Review of Individual Progress

Table 5 Meeting 3

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	98%	Create user interface mockups.
Vishva	Research more on Functionalities	80%	Define the database schema.
Premal	Basic details using Angular forms	90%	Develop use case diagrams.

4.4 Meeting 4

Time :28th May 2024, Year, hour: 9:00 AM to hour: 11:00 AM
Agenda: Review of Individual Progress

Table 6 Meeting 4

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	100%	Design API endpoints.
Vishva	Research more on Functionalities	100%	Set up Angular project.
Premal	Basic details using Angular forms	100%	Implement login and authentication.

4.5 Meeting 5

Time: 31st May 2024, Year, hour: 3:00 PM to hour: 5:00 PM
Agenda: Review of Individual Progress

Table 7 Meeting 5

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	98%	Develop the item listing page.

Vishva	Research more on Functionalities	90%	Create forms for adding/editing items.
Premal	Basic details using Angular forms	80%	Implement item filtering functionality.

4.6 Meeting 6

Time: 4th June 2024, Year, hour: 1:00 PM to hour: 3:00 PM
Agenda: Review of Individual Progress

Table 8 Meeting 6

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	80%	Develop transaction history page.
Vishva	Research more on Functionalities	100%	Add data visualization features.
Premal	Basic details using Angular forms	90%	Set up Spring Boot project.

4.7 Meeting 7

Time: 6th June 2024, Year, hour: 4:00 PM to hour: 6:00 PM
Agenda: Review of Individual Progress

Table 9 Meeting 7

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	98%	Implement CRUD operations for items
Vishva	Research more on Functionalities	80%	Implement data persistence with Spring Data JPA
Premal	Basic details using Angular forms	97%	Set up Docker for backend deployment.

4.8 Meeting 8

Time: 10th June 2024, Year, hour: 11:00 AM to hour: 1:00 PM
Agenda: Review of Individual Progress

Table 10 Meeting 8

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	100%	Update project documentation.
Vishva	Research more on Functionalities	100%	Perform end-to-end testing.
Premal	Basic details using Angular forms	90%	Fix identified bugs and issues.

4.9 Meeting 9

Time: 12th June 2024, Year, hour: 5:00 PM to hour: 7:00 PM
Agenda: Review of Individual Progress

Table 11 Meeting 9

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	80%	Deploy the application to a staging environment
Vishva	Research more on Functionalities	100%	Perform user acceptance testing.
Premal	Basic details using Angular forms	100%	Deploy the application to production.

4.10 Meeting 10

Time: 14th June 2024, Year, hour: 8:00 AM to hour: 10:00 AM
Agenda: Review of Individual Progress

Table 12 Meeting 10

Team Member	Previous Task	Completion State	Next Task
Drashti	Research on building project on Angular and Spring Boot	85%	Monitor post-deployment performance.
Vishva	Research more on Functionalities	80%	Gather feedback from users.
Premal	Basic details using Angular forms	92%	Identify potential improvements.

4.11 Meeting 11

Time: 16th June 2024, Year, hour: 2:30 PM to hour: 4:30 PM Agenda: Review of Individual Progress

Table 13 Meeting 11

Team Member	Previous Task	Completion State	Next Task
Drashti	Spring Data JPA and expose REST APIs for CRUD operations	90%	Prepare final project report and presentation.
Vishva	Main CRUD operations	80%	Document lessons learned
Premal	Angular forms and Spring Boot APIs	100%	Plan for future updates.

5 Project Management

Table 14 Gantt Chart

Task/Phase	Start Date	End Date	Duration
Solution 1			
- Requirement Gathering	05/15/2024	05/19/2024	5 days
- Design	05/20/2024	05/24/2024	5 days
- Development	05/25/2024	06/08/2024	15 days
- Testing	06/09/2024	06/13/2024	5 days
- Review & Refine	06/14/2024	06/18/2024	5 days
Solution 2			
- Requirement Gathering	06/19/2024	06/23/2024	5 days
- Design	06/24/2024	06/28/2024	5 days
- Development	06/29/2024	07/12/2024	14 days
- Testing	07/13/2024	07/17/2024	5 days
- Review & Refine	07/18/2024	07/20/2024	3 days
Final Solution			
- Requirement Gathering	05/15/2024	05/19/2024	5 days
- Design	05/20/2024	05/24/2024	5 days
- Development	05/25/2024	06/22/2024	29 days
- Testing	06/23/2024	06/29/2024	7 days
- Review & Refine	06/30/2024	07/20/2024	21 days

6 Conclusion and Future Work

The development of this inventory management system represents a significant step forward in enhancing operational efficiency and decision-making capabilities for businesses. By integrating Angular for the frontend and Spring Boot for the backend, the system offers a seamless user experience with robust data handling capabilities. The implementation of advanced features such as transaction tracking, item history viewing, and efficient filtering mechanisms ensures that the system meets the complex demands of modern inventory management.

The iterative development approach, coupled with comprehensive testing, has resulted in a reliable and maintainable codebase. The use of Docker for deployment ensures scalability and ease of use across different environments. The system's design considerations, including environmental, societal, safety, and economic factors, reflect a commitment to sustainable and ethical business practices.

In conclusion, this inventory management system not only addresses the immediate needs of businesses but also positions them for future growth and adaptability. The project team's collaborative efforts and meticulous planning have culminated in a solution that is poised to make a tangible impact on the operations of businesses that adopt it. As technology and business environments continue to evolve, this system stands as a testament to the potential of innovative software solutions to drive efficiency and effectiveness in inventory management.[11][12][13][14][15]

7 References

[1] "Inventory Management System Project," Introduction and Conclusion, Unpublished manuscript.

[2] Angular, "Angular Documentation," [Online]. Available: https://angular.io/docs.

[3] Spring Boot, "Spring Boot Documentation," [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/.

[4] MySQL, "MySQL Documentation," [Online]. Available: https://dev.mysql.com/doc/.

[5] Docker, "Docker Documentation," [Online]. Available: https://docs.docker.com/.

[6] JUnit, "JUnit Documentation," [Online]. Available: https://junit.org/junit5/docs/current/user-guide/.

[7] Mockito, "Mockito Documentation," [Online]. Available: https://site.mockito.org/.

[8] H2 Database, "H2 Database Documentation," [Online]. Available: http://www.h2database.com/html/main.html.

[9] Angular Material, "Angular Material Documentation," [Online]. Available: https://material.angular.io/.

[10] Spring Data JPA, "Spring Data JPA Documentation," [Online]. Available: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/.

[11] "Inventory Management: A Comprehensive Guide," Business News Daily, 2020. [Online]. Available: https://www.businessnewsdaily.com/10715-inventory-management.html.

[12] "The Importance of Efficient Inventory Management," Forbes, 2018. [Online]. Available: https://www.forbes.com/sites/forbesfinancecouncil/2018/05/10/the-importance-of-efficient-inventory-management/.

[13] "How Technology is Changing Inventory Management," Harvard Business Review, 2019. [Online]. Available: https://hbr.org/2019/01/how-technology-is-changing-inventory-management.

[14] "The Role of Inventory Management in Supply Chain Efficiency," Supply Chain Management Review, 2017. [Online]. Available: https://www.scmr.com/article/the_role_of_inventory_management_in_supply_chain_efficiency.

[15] "Best Practices for Inventory Management," Entrepreneur, 2020. [Online]. Available: https://www.entrepreneur.com/article/353472.