PSTU Inventory Management System

Submitted by:

Rakibul Islam Rehan

Registration No: 08767

Student ID: 1902061

Supervised By:

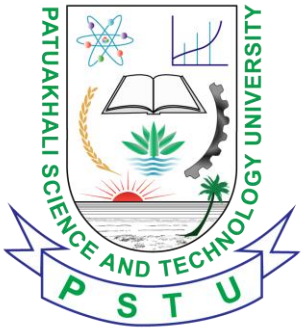
Dr. Md. Samsuzzaman

Professor

Department of Computer and Communication Engineering

Faculty of Computer Science and Engineering

A PROJECT IN THE FACULTY OF COMPUTER SCIENCE AND ENGINEERING PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Faculty of Computer Science and Engineering

Patuakhali Science and Technology University

Patuakhali, Bangladesh

July 2025

DECLARATION

This is to certify that the work presented in this project, titled “PSTU Inventory Management System”, is the outcome of the investigation and research carried out by us under the supervision of Prof. Dr. Md. Samsuzzaman.

It is also declared that neither this project nor any part thereof has been submitted anywhere else for the award of any degree, diploma, or other qualifications.

……
Rakibul Islam Rehan
1902061

CERTIFICATION

This project, titled “PSTU Inventory Management System”, was submitted by Rakibul Islam Rehan (Id: 1902061) has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in July 2025.

Approval of the Supervisor

Dr. Md. Samsuzzaman

Professor

Department of Computer and Communication Engineering

Faculty of Computer Science and Engineering

Board of Examiners

Signature : _____

Name of the Supervisor : Dr. Md. Samsuzzaman

Professor, Department of Computer and Communication Engineering

Signature : _____

Chairman of the CCE & Internal Member: Golam Md. Muradul Bashir

Professor, Department of Computer and Communication Engineering

Signature : _____

Name of Internal Member : Sarna Majumder

Associate Professor, Department of Computer and Communication Engineering

Signature : _____

Name of Internal Member : Arpita Howlader

Assistant Professor, Department of Computer and Communication Engineering

Signature : _____

Name of External Member : Dr. Md. Abdul Masud

Professor, Department of Computer Science and Information Technology

ACKNOWLEDGEMENTS

First and foremost, I express my heartfelt gratitude to the Almighty for granting me the strength, patience, and determination to complete this project successfully.

I would like to extend my deepest appreciation to my supervisor, Prof. Dr. Md. Samsuzzaman, for his invaluable guidance, continuous support, and insightful feedback throughout the development of this project. His encouragement and expertise have been instrumental in shaping the direction and quality of this work.

I am also thankful to the faculty members of the Faculty of Computer Science and Engineering, Patuakhali Science and Technology University, for equipping me with the foundational knowledge and technical skills necessary to undertake this project.

Finally, I am deeply grateful to my family and friends for their constant encouragement, emotional support, and understanding throughout the entire journey.

With Best Regards,
Rakibul Islam Rehan

Abstract

The “PSTU Inventory Management System” is designed to provide a secure, scalable, and efficient platform for managing inventory at Patuakhali Science and Technology University (PSTU). The system aims to streamline inventory management by allowing administrators and department staff to track and manage inventory items, suppliers, stock in and out processes, and reports efficiently.The platform enhances inventory tracking by offering real-time updates on stock levels, easy item categorization, and historical records of stock movements. The system integrates role-based access control to ensure that only authorized users can access or modify specific data, thus enhancing security and accountability.Key features of the system include user authentication, item management, department-specific stock allocation, stock issuance, and report generation. The system ensures seamless operation by providing an intuitive interface for inventory management, making it easy for users to interact with the system without technical expertise.This report presents the architecture, implementation, and functionality of the PSTU Inventory Management System, showcasing its potential as a comprehensive tool for managing and tracking the university’s inventory in a secure and efficient manner.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Background

At Patuakhali Science and Technology University (PSTU), managing the university's inventory has traditionally been a manual and time-consuming process. Faculty, staff, and administrative personnel rely on pen-and-paper methods to track items, which not only increases the risk of human error but also makes it challenging to maintain an accurate, up-to-date record of the university's resources. From office supplies and laboratory equipment to essential materials for academic departments, managing inventory without a digital solution has led to inefficiencies, delays in procurement, and a lack of real-time visibility into the availability of critical items.
In this manual system, each department must maintain its records, leading to inconsistencies and redundancies across various sections. For instance, when faculty members request supplies, these requests are often lost in paperwork, or their approval is delayed, resulting in stockouts or overstocking. Without a centralized system, there's little accountability or easy tracking of where items are located, which can lead to misplaced or wasted resources.
The existing method of maintaining inventory on paper, while functional in the past, is no longer adequate for the needs of an expanding institution like PSTU. The need for an efficient, transparent, and automated solution has become critical to ensure better resource management, accountability, and improved decision-making.

1.2 Problem Statement

The offline, pen-and-paper inventory management system at PSTU faces several significant challenges that hinder its effectiveness:

Manual Record-Keeping: The reliance on manual tracking creates opportunities for human error, leading to discrepancies between physical stock and recorded data. Misplaced records and inconsistencies in data are common, making it difficult to trust inventory information.

Inefficiencies in Stock Management: Without a centralized system, faculty and staff must manually track the availability of resources, leading to unnecessary delays in procurement and a lack of real-time awareness regarding stock levels.

Lack of Accessibility and Visibility: As the inventory data is not digitized, accessing and sharing inventory information is cumbersome. Stakeholders must physically visit departments to request stock updates, leading to inefficiencies in resource allocation.
Difficulties in Reporting: Generating reports for resource allocation, budgeting, or audits is a slow and error-prone process. Manual reports are not only time-consuming but also prone to inaccuracies, making it challenging for administrators to track and plan effectively.

In light of these challenges, there is an urgent need for a centralized, automated, and secure inventory management system to streamline the process, increase transparency, and reduce the burden on university staff.

1.3 Objectives

The main objective of this project is to design and develop a digital, secure, and scalable inventory management system for PSTU, replacing the outdated pen-and-paper process. By transitioning to a more modern and efficient solution, the system aims to:

Automate Inventory Tracking: Eliminate the need for manual record-keeping by automating the tracking of stock items, ensuring that all data is updated in real time and accurately reflects the actual stock levels.

Centralized Data Access: Provide a single platform for all inventory data, accessible to authorized users across departments. This will enable real-time updates and access to accurate information from anywhere within the university.

Enable Role-Based Access Control: Implement a secure system that grants varying levels of access to different users based on their role (e.g., Admin, Department Head, Staff). This will ensure that only authorized personnel can update or manage inventory data.
Generate Real-Time Reports: Allow users to generate automated reports based on inventory data, including stock levels, department-wise usage, and procurement status. These reports can be used for decision-making, budgeting, and resource allocation.

1.4 Motivation

The motivation for developing the PSTU Inventory Management System stems from my personal experience and the challenges I witnessed during my time at PSTU. The existing pen-and-paper system was not only outdated but also increasingly unmanageable as the university's resources grew. Faculty and administrative staff faced constant struggles with tracking and managing inventory efficiently, leading to frustration and delays in procurement and inventory updates.During my involvement with the university's administrative processes, I realized that the university needed a solution that could automate stock tracking, ensure real-time updates, and provide users with easy access to accurate data.

1.5 Summary

This chapter provided an overview of the background, problem statement, and objectives behind the development of the PSTU Inventory Management System. It highlighted the inefficiencies of the current pen-and-paper system and outlined the challenges faced by the university in managing its resources. The chapter also introduced the system's objectives,

focusing on automation, security, transparency, and real-time reporting. Lastly, the motivation behind the project was discussed, driven by the need to replace the outdated manual system with a more efficient and scalable digital solution.In the subsequent chapters, the system design, architecture, and technical implementation will be discussed in detail, providing insights into how the PSTU Inventory Management System will solve the current challenges and improve the management of university resources.

Chapter 2: Literature Review

2.1 Introduction

This chapter reviews existing inventory management systems and their features related to stock tracking, procurement management, and report generation. It examines the strengths and limitations of current systems used in educational institutions and businesses globally, highlighting the challenges faced by administrators in efficiently managing university resources. Furthermore, this chapter explores how many existing systems struggle with issues such as manual tracking, lack of real-time updates, and insufficient reporting capabilities. By comparing these existing solutions with the design goals of the PSTU Inventory Management System, this chapter illustrates how the new system aims to address these gaps by providing a secure, efficient, and user-friendly platform tailored to the specific needs of PSTU.

2.2 Existing Systems

Over the years, various inventory management systems have been developed to help organizations keep track of their resources. These systems vary widely in terms of features, scalability, and integration capabilities. Below is an overview of some of the existing systems that are relevant to this project:

1. Spreadsheets and Manual Records: Basic but Error-Prone
   The most common system used in educational institutions, including PSTU, is the manual entry of inventory data into spreadsheets. While this method allows basic record-keeping, it is highly prone to errors, especially with large inventories. It lacks real-time tracking, automatic updates, and generates reports manually, which can be time-consuming and error-prone.

2. Legacy Inventory Management Systems: Traditional but Inflexible
   Older inventory systems, often used by large institutions, involve complex desktop applications with limited integration options. These systems typically require manual updates for stock entries, leading to potential delays and inaccuracies. Moreover, they are often hard to scale and not user-friendly for modern users, especially when departments want to make updates independently.

3. Cloud-Based Inventory Management Systems: Accessible but Limited for Institutions
   Several cloud-based systems (e.g., Zoho Inventory, TradeGecko) provide an accessible solution to inventory management. These systems support real-time tracking and reporting, and can be accessed from multiple devices. However, most of these platforms are designed for businesses and not tailored to the unique needs of educational institutions like PSTU. Many systems lack detailed customization, access control, or integration with other campus systems.

4. Open-Source Inventory Management Systems: Customizable but Complex
   Open-source solutions like Odoo and Snipe-IT offer highly customizable features, such as stock tracking, asset management, and reporting. While they provide flexibility and

cost savings, they often require substantial technical knowledge to deploy and maintain. The complexity of these systems can make them impractical for educational staff without specialized IT support.

5. Enterprise Resource Planning (ERP) Systems: Comprehensive but Overkill
   Larger institutions may opt for full-scale ERP systems like SAP or Oracle ERP, which include inventory management as part of a broader suite of functionalities. While these systems offer powerful tracking and reporting tools, they are often costly and difficult to implement for smaller organizations like PSTU. Furthermore, they come with many features that may not be necessary for university-level inventory management, making them overly complex.

2.3 Comparison with PSTU Inventory Management System

The PSTU Inventory Management System was developed to directly address the key limitations found in existing inventory management solutions, especially in the context of resource tracking, departmental management, and data reporting. Below is a comparison of key features:

1. Real-Time Stock Tracking vs. Manual Updates
   Existing systems, particularly spreadsheets, require manual updates to reflect current stock levels, which can lead to discrepancies and delays. The PSTU Inventory Management System automates stock updates in real-time, ensuring that inventory data is always accurate and up-to-date.

2. Centralized System vs. Decentralized Record-Keeping
   While many systems require departments to manage their own inventory separately, leading to inconsistent data, the PSTU system centralizes all inventory data in one place. This ensures a unified view of resources across the entire university, allowing administrators to manage stock levels, requisitions, and supplies from a central dashboard.

3. Role-Based Access Control vs. Lack of Security
   Unlike many traditional systems where access control is limited, the PSTU Inventory Management System introduces role-based access, ensuring that only authorized personnel can modify inventory data. This helps prevent unauthorized changes and enhances the security of sensitive information.

4. Customizable Reports vs. Generic Reporting
   Many existing systems, such as legacy software and cloud-based platforms, generate generic reports that are not always suited to the specific needs of educational institutions. The PSTU system enables the generation of customizable reports, which

can be filtered by department, office, or date range, making it easier for university administrators to analyze inventory usage and trends.

2.4 Summary

This chapter reviewed existing inventory management systems, including manual record-keeping, legacy systems, cloud-based solutions, open-source platforms, and ERP systems, highlighting their strengths and limitations. While many of these systems offer useful features, they often fall short in providing real-time tracking, secure access control, intuitive interfaces, and the ability to generate customized reports.By addressing these gaps, the PSTU Inventory Management System provides a secure, scalable, and user-friendly solution for managing the university's resources. With features such as real-time stock updates, role-based access control, and customizable reporting, the system is tailored to the unique needs of PSTU, offering an efficient way to track and manage inventory across departments.The following chapters will detail the design, implementation, and testing phases of the PSTU Inventory Management System, demonstrating how it provides a modern and efficient approach to inventory management at PSTU.

Chapter 3: Methodology

3.1 Introduction

The methodology for developing the PSTU Inventory Management System is grounded in established software development practices and tailored to meet the specific needs of Patuakhali Science and Technology University (PSTU). This section outlines the various techniques and processes employed throughout the project to ensure a systematic and efficient approach to development. The chosen methodology emphasizes thorough requirements gathering, structured design, implementation, and testing to create a user-friendly and robust inventory management system. By adhering to industry best practices, the project aims to deliver a solution that not only meets current inventory management demands but also remains flexible enough to accommodate future enhancements and integrations. The methodology is organized into distinct sections, covering fact-finding techniques, the Software Development Life Cycle (SDLC), database design, and system integration, each playing a critical role in the successful development of the system.

3.2 Fact-Finding Techniques

To develop an effective inventory management system for PSTU, various fact-finding techniques were employed during the requirements gathering phase. These techniques facilitated a comprehensive understanding of the users' needs, preferences, and pain points.
In-depth discussions were conducted with administrative staff, department heads, and IT cell members at PSTU to gather qualitative insights. These conversations allowed for a deeper exploration of the existing manual inventory system, its inefficiencies, and specific needs related to resource management. Focus group discussions helped in understanding the challenges faced by different departments when managing stock and procurement.
Additionally, observational studies were conducted in various departments and offices to understand workflow patterns and challenges faced by staff when managing physical inventories. This technique helped uncover specific issues that may not have been articulated in surveys or interviews.

3.3 Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a structured approach essential for the systematic development of the PSTU Inventory Management System. The following phases outline the key steps involved in the SDLC:

1. Planning: In this initial phase, project goals, scope, and timelines were defined. Stakeholder meetings were conducted to outline the project's vision and ensure alignment with the needs of PSTU's IT Cell.

2. Requirements Gathering: This phase utilized the fact-finding techniques previously discussed to compile a comprehensive list of user needs. Interviews and focus groups

were instrumental in capturing both quantitative and qualitative data on user expectations and departmental requirements.

3. Design: The design phase focused on creating the system architecture, user interfaces, and database schemas based on the gathered requirements. Emphasis was placed on ensuring a user-friendly experience for non-technical staff while maintaining robustness and security.

4. Implementation: During the implementation phase, the development of the backend and frontend was carried out. Modern technologies were employed to create a responsive, efficient, and secure platform. A RESTful API was developed for user management, stock management, and real-time updates.

5. Testing: The testing phase was critical for verifying that the system functioned as intended. It included:
   Unit Testing: Testing individual components for functionality.
   Integration Testing: Ensuring that different modules work together seamlessly.
   User Acceptance Testing (UAT): Validating the system with actual users to confirm it meets their needs and expectations.

6. Deployment: In this phase, the system was deployed on the university's local servers, ensuring that all data related to inventory tracking remained secure and accessible only by authorized users.

7. Maintenance and Evaluation: After deployment, ongoing evaluation and maintenance processes were established to address user feedback, implement updates, and ensure that the system continued to meet the evolving needs of PSTU.

3.3.1 Software Process Model

The software development process for the PSTU Inventory Management System follows a structured approach, starting from requirements analysis and design, leading to development, testing, deployment, and maintenance. The system utilizes React.js and TailwindCSS for the frontend, and Node.js, Express.js, MongoDB, and Redis for the backend. This combination ensures a scalable, responsive, and secure inventory management solution that aligns with the needs of PSTU.

3.3.2 Feasibility Study

The feasibility study assesses the viability of the PSTU Inventory Management System by examining technical, operational, financial, and scheduling aspects:

1. Technical Feasibility:

The project utilizes a reliable technology stack, including React.js, Node.js, Express.js, MongoDB with Mongoose, and Redis. The use of a RESTful API ensures scalability and modularity, making the project technically feasible. The stack's stability and strong community support ensure long-term sustainability and ease of future enhancements.

2. Operational Feasibility:

   The system addresses the operational needs of PSTU by providing a web-based platform for inventory management. Features such as real-time stock updates, role-based access control, and automated report generation streamline the lab management and procurement processes for both instructors and administrators. The user-friendly interface ensures the system is easy to use, even for non-technical staff.

3. Financial Feasibility:

   Development costs are minimized by utilizing open-source tools and technologies. For hosting and maintenance, PSTU's existing server infrastructure will be used, which is offered free for university projects. This ensures that the project is financially viable for the university, with minimal ongoing operational costs.
4. Schedule Feasibility:

   The project follows an iterative development model with well-defined milestones. The timeline ensures that resources are allocated efficiently, and risk management strategies are in place to meet project deadlines. The structured plan allows for timely completion, ensuring the system will be ready for deployment within the planned timeframe.

3.4 Database Design

The database design is fundamental to ensuring data integrity, performance, and scalability for the PSTU Inventory Management System. The system utilizes MongoDB as its NoSQL database, which is structured into collections tailored to meet the university's inventory and user management requirements. Below is a detailed explanation of the key collections and their schemas.

3.5 Key Collections and Their Schemas

3.5.1 Department Schema

The Department schema stores information about each department within the university. The fields include:
- name: The department's name.
- code: Unique identifier or code for the department.
- description: A short description of the department.
- faculty: The faculty to which the department belongs.
Timestamps: Automatically stores the created and updated timestamps.

3.5.2 Office Schema

The Office schema represents the various administrative offices within the university. The fields include:
- name: Name of the office.
- description: A brief description of the office.
- section: The specific section or unit within the office.
Timestamps: Automatically stores the created and updated timestamps.

3.5.3 Supplier Schema

The Supplier schema stores information about the suppliers who provide the inventory items. The fields include:
- name: Name of the supplier.
- contactPerson: The primary contact person for the supplier.
- phone: Supplier’s phone number.
- email: Supplier’s email address.
- address: The physical address of the supplier.
Timestamps: Automatically stores the created and updated timestamps.

3.5.4 Item Schema

The Item schema stores information about each inventory item. The fields include:
- name: Name of the inventory item.
- description: A description of the item.
- category_id: Reference to the Category collection that categorizes the item (e.g., stationery, electronics).
- unit: The unit of measurement for the item.
Timestamps: Automatically stores the created and updated timestamps.

3.5.5 StockIn Schema

The StockIn schema stores information about items added to the inventory. The fields include:
- user_id: The user who added the stock .
- department_id: Reference to the department receiving the stock.
- office_id: Reference to the office receiving the stock.
- item_id: Reference to the item being stocked.
- supplier_id: Reference to the supplier providing the stock.
- quantity: Quantity of the item being added.
- unit_price: Price per unit of the item.
- total_price: Total price for the stocked quantity.
- purchase_date: Date when the stock was received.
- invoice_no: The invoice number associated with the purchase.
- remarks: Additional notes or comments.
Timestamps: Automatically stores the created and updated timestamps.

3.5.6 StockOut Schema

The StockOut schema stores information about items issued or removed from inventory. The fields include:
- user_id: The user who issued the stock.
- department_id: Reference to the department receiving the item.
- office_id: Reference to the office receiving the item.
- item_id: Reference to the item being issued.
- issue_type: Type of issue.
- issue_by: The person who authorized the stock out.
- issue_date: Date of the stock issue.
- quantity: Quantity of the item issued.
- remarks: Additional comments or reasons for stock out.
Timestamps: Automatically stores the created and updated timestamps.

3.5.7 DeadStock Schema

The DeadStock schema stores information about damaged, expired, or unusable inventory items. The fields include:
- user_id: The user reporting the dead stock.
- item_id: Reference to the item being reported as dead stock.
- stock_out_id: Reference to the related StockOut entry.
- quantity: Quantity of the dead stock item.
- reason: The reason for the item being reported as dead stock.
- reported_at: Date the dead stock was reported.
Timestamps: Automatically stores the created and updated timestamps.

3.5.8 User Schema

The User schema stores information about system users, including faculty, staff, and administrators. The fields include:
- name: User's full name.
- email: User's email address.
- password: Hashed password for secure authentication.
- role: The user's role in the system.
- phone_number: User's contact number.
- department_id: Reference to the Department the user is associated with.
- office_id: Reference to the Office the user is part of.
Timestamps: Automatically stores the created and updated timestamps.

3.5.9 Category Schema

The Category schema is used to classify inventory items into various categories. The fields include:
- name: Name of the category.
- description: A brief description of the category.
- created_at: Timestamp for when the category was created.
- updated_at: Timestamp for when the category was last updated.
Timestamps: Automatically stores the created and updated timestamps.

3.5.10 StockHistory Schema

The StockHistory schema stores logs of actions performed on inventory items . The fields include:
- item_id: Reference to the Item being acted upon.
- action: The action performed on the item .
- reference_id: Reference to the relevant StockIn, StockOut, or DeadStock record.
- quantity: Quantity of the item involved in the action.
- performed_by: User who performed the action .
- date: Date the action was performed.
Timestamps: Automatically stores the created and updated timestamps.

3.5.11 Report Schema

The Report schema stores details of generated reports.
- report_type: Type of report .
- generated_by: Reference to the User who generated the report

3.6 Database ER Diagram

To visualize the Entity-Relationship (ER) Diagram for your PSTU Inventory Management System, I will design an ER diagram based on the collections and relationships within your database, focusing on the connections between departments, items, stock management, users, and other entities.

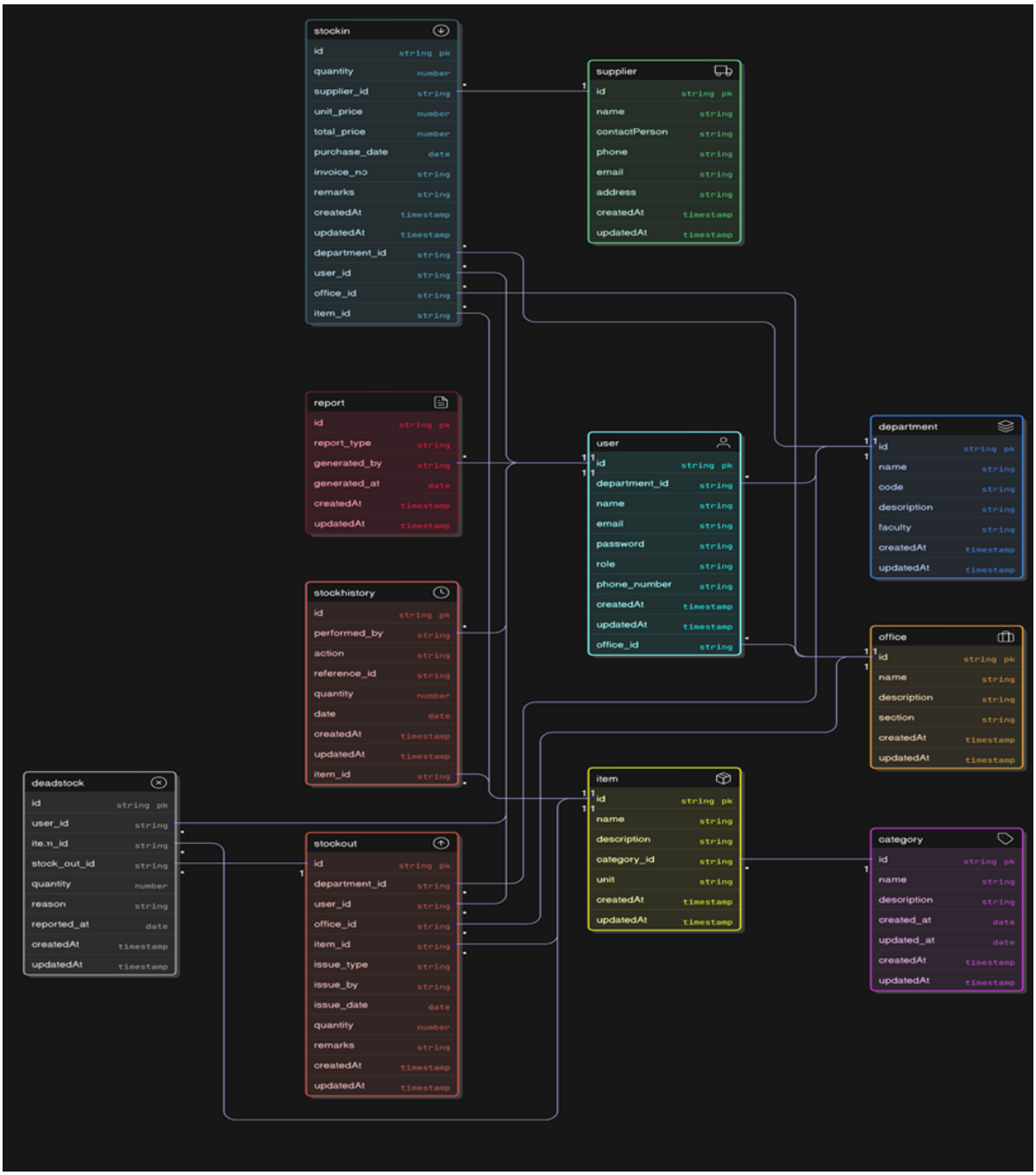


Fig 3.1. Database ER Diagram

3.6.1 Key Entities:

1. Supplier: This entity stores information about suppliers who provide items. It includes attributes like:
    - supplier_id, name, contact_person, contact_email, etc.
    - Relationships: It is connected to the item entity, showing that a supplier can supply many items.
2. Item: Represents the products or items in the inventory. Key attributes include:
    - item_id, description, quantity, unit_price, etc.
    - Relationships: It is connected to stockout (for managing item issues), supplier (from which the item is sourced), category (for item classification), and office (for location or assignment).
3. Report: Used for generating inventory reports. Attributes like:
    - report_id, report_type, generated_date, etc.
    - Relationships: It seems to be connected to office, indicating that reports are associated with specific offices within the system.
4. Department: Represents various departments in the university. It holds attributes like:
    - department_id, name, faculty, etc.
    - Relationships: Departments can request items, manage stock, and receive stock from suppliers.
5. StockHistory: Tracks the history of stock movements (e.g., addition, reduction). Important attributes:
    - history_id, item_id, quantity_changed, date, etc.
    - Relationships: Connected to item to show which items have had stock movements.
6. Office: Represents different locations or sections within the university, like administrative offices or faculties. It holds:

- office_id, name, location, etc.
- Relationships: Items are assigned to specific offices for usage.

7. Category: This categorizes items into different groups like furniture, electronics, etc. It contains:
   - category_id, category_name.
   - Relationships: Items are associated with a specific category.
8. User: Represents users who can interact with the system (e.g., admins, staff, etc.). Key attributes:
   - user_id, role, email, etc.
   - Relationships: Users are associated with various roles and actions within the system, such as creating requests, managing inventory, and more.
9. StockOut: Tracks the issue of items to different departments or offices. Key fields:
   - stockout_id, item_id, quantity, issued_to, etc.
   - Relationships: It connects with the item and department entities, showing which items are issued to which departments.
10. Location: Represents the locations (like office, faculty areas, etc.) within the university where the inventory items are assigned. It may link to:
    - The office or department entities.

3.7 Summary

This chapter detailed the methodology applied throughout the PSTU Inventory Management System project lifecycle, covering development models, system architecture, database design,

feasibility studies, and testing strategies. A structured approach, based on industry best practices, was followed to ensure the development of a secure, efficient, and user-friendly platform for inventory management at Patuakhali Science and Technology University (PSTU).

The database design was critical to ensure scalability, data integrity, and performance, with MongoDB as the chosen NoSQL database, structured into collections tailored for user management, inventory tracking, stock management, and reporting. A feasibility study was conducted to assess the project's technical, operational, financial, and schedule feasibility, ensuring that the system is viable and sustainable.

The next chapter will focus on the system design specifics, including the architectural diagrams, detailed interface descriptions, and how the system components interact to provide a seamless inventory management solution.

Chapter 4: PSTU Inventory Management System Design

4.1 Introduction

This chapter presents a comprehensive system design for the PSTU Inventory Management System, a sophisticated web-based platform developed to streamline and automate inventory management processes at Patuakhali Science and Technology University (PSTU). The primary objective of this system is to deliver an interactive, scalable, and secure environment for managing inventory, tracking stock movements, and generating reports across various departments, offices, and administrative units at the university.The design emphasizes modularity, scalability, and responsiveness, incorporating modern software engineering paradigms such as microservices architecture, containerization, and real-time data processing. This chapter elaborates on the system architecture, core components, backend and frontend implementations, infrastructure services, and detailed implementation strategies, providing a thorough blueprint for the platform's development.

4.2 Overall System Design

The PSTU Inventory Management System is architecturally crafted to ensure scalability, modularity, and a seamless user experience while prioritizing robust security and data integrity. The system integrates multiple subsystems that operate cohesively to deliver high performance, reliability, and maintainability. It adheres to contemporary software development standards and is built to accommodate diverse user needs, from inventory tracking and procurement requests to administrative oversight. The system is designed to replace the university's outdated pen-and-paper inventory management system by automating inventory operations, improving accuracy, and providing real-time updates to stakeholders.

4.2.1 System Architecture

The platform adopts a modular client-server architecture, organized into distinct layers to ensure separation of concerns, scalability, and ease of maintenance. Below is an outline of the various layers:

4.2.1.1 Frontend Layer:

The frontend layer provides a dynamic and responsive user interface that ensures ease of use for faculty, staff, and administrators. The core components of the frontend include:

Interactive Dashboard: Provides an overview of inventory data, stock levels, procurement requests, and inventory reports.

Inventory Management Interface: Allows users to track, add, update, and remove items from inventory. Users can request items, monitor stock in/out, and report dead stock.

User Management Panel: Admins can create, modify, and assign user roles, ensuring role-based access to sensitive inventory data.

Real-time Reporting: Allows administrators to generate real-time reports on inventory status, stock movements, department-wise usage, and more.

Responsive Design: Built with React.js and Tailwind CSS, ensuring the platform is accessible on desktops and mobile devices.

4.2.1.2 Backend Layer:

The backend exposes a RESTful API that serves as the intermediary between the frontend and the infrastructure services. The backend manages the following:

User Authentication & Authorization: Ensures secure user login, role-based access control using JWT tokens.

Database Operations: Handles CRUD operations for inventory data, user management, and stock records via MongoDB.

Stock Management: Manages stock in, stock out, and stock tracking, ensuring that inventory data is updated in real-time.

Report Generation: Processes and generates reports based on inventory data, including stock levels, procurement requests, and user activities.

4.2.1.3 Infrastructure Services:

Infrastructure services form the backbone of the system, providing scalability, security, and data integrity.

Queue Management System: Utilizes Redis for real-time processing of stock requests, updates, and report generation. It ensures asynchronous processing and efficient task execution.

Containerization: Implements containerized environments using Docker to isolate sensitive inventory data processes, ensuring that each department or office's data is handled independently.

Database: A MongoDB NoSQL database stores inventory data, including items, users, departments, and stock transactions. The database is scalable and provides real-time updates for inventory management.

Email Notifications: Integrated email services notify users when stock requests are processed or when important updates are made to inventory data.

4.2.1.4 Communication Protocols:

HTTP/HTTPS: Used for secure communication between the frontend and backend layers.

WebSocket: Used for real-time updates, ensuring that changes in inventory data are immediately reflected in the user interface.

4.2.1.5 Security:

The system implements best practices for data security, including:

JWT Authentication: Ensures secure user login and session management.

Data Encryption: Protects sensitive data (e.g., passwords, transaction data) using encryption.

Role-Based Access Control: Restricts access to specific data based on user roles, ensuring that only authorized personnel can access or modify critical inventory information.

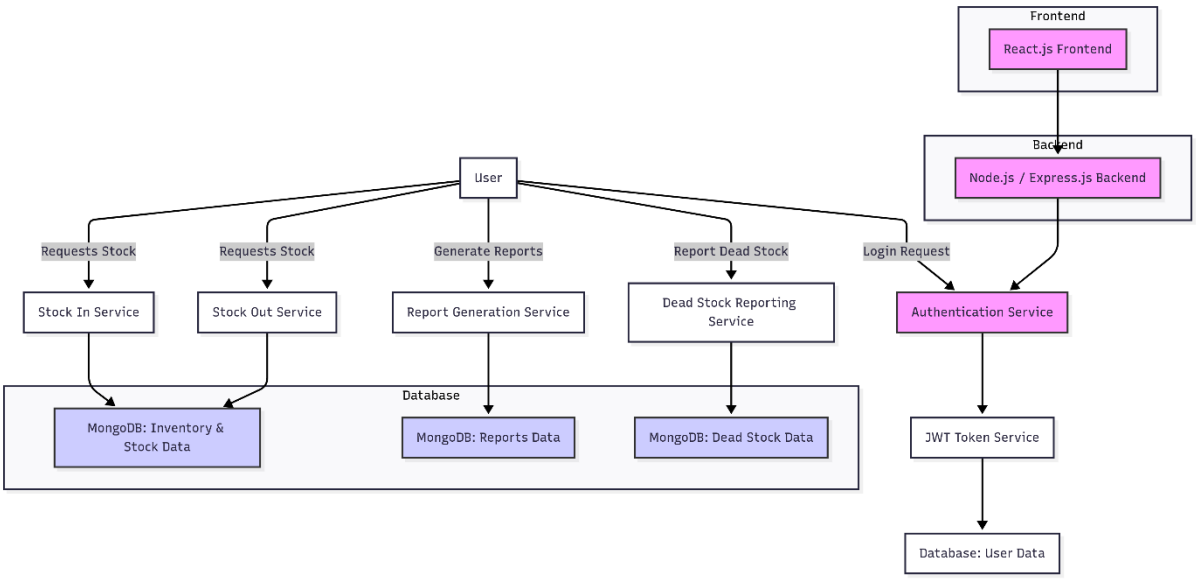


Fig 4.1. System Architecture of PSTU Inventory Management System

4.2.2 Infrastructure Services

The PSTU Inventory Management System includes several key infrastructure services that handle critical functionalities in a distributed manner, ensuring scalability, reliability, and efficient resource allocation. Two primary services are central to the system: the Email Service and the Stock Request Processing Service. These services are designed to operate efficiently and asynchronously to support the seamless functioning of the system.

The Stock Request Processing Service is designed to manage and execute stock-related tasks, such as stock requisition, stock-out requests, and report generation. The service ensures that inventory requests and actions are processed efficiently, without overloading the backend system.

Redis Submission Queue**:** The Stock Request Processing Service uses Redis to queue stock request tasks, including adding new stock items, processing requests, and generating inventory reports. Queuing these tasks allows the system to handle multiple requests in parallel without overburdening the backend, ensuring smooth operation even during periods of high request volume.

Execution Engine: The execution engine is the core component that processes the stock request tasks. It retrieves tasks from the Redis queue, performs the necessary actions (e.g., updating inventory data or generating reports), and ensures that each action is completed in the appropriate order.

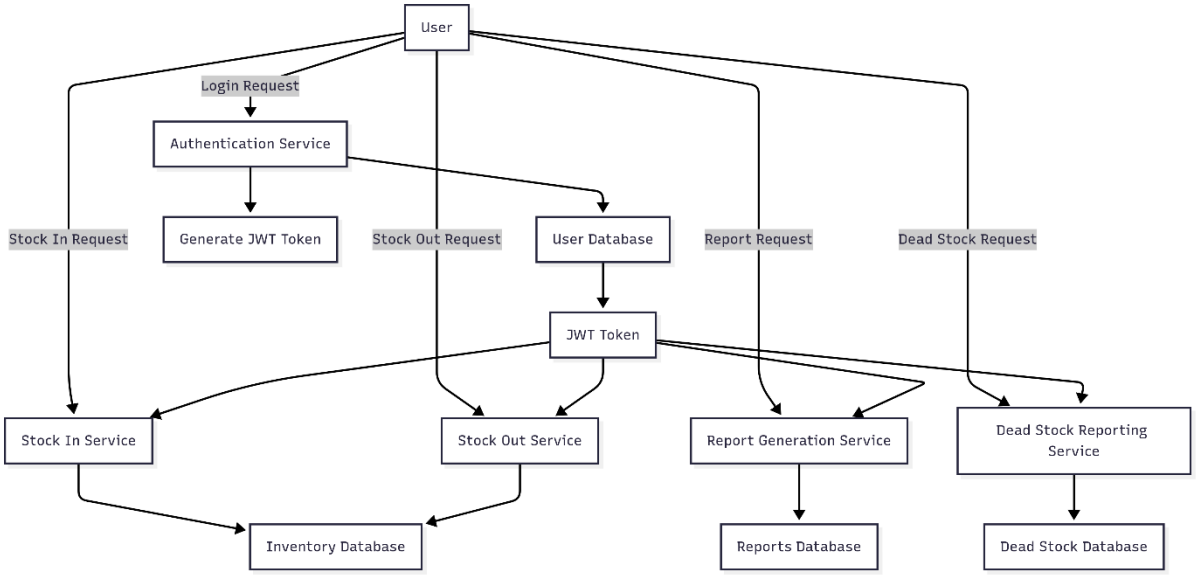


Fig 4.2. Code Execution Workflow

4.2.3 User Activity Diagram

The User Activity Diagram for the PSTU Inventory Management System provides an overview of the main interactions that users undertake within the system, illustrating the sequence of actions from logging in to managing inventory tasks and receiving notifications.

Login Process: Users begin by securely logging into the system. Authentication is processed through the backend via API calls, ensuring that only authorized users can access the system. Once logged in, users are directed to their personalized dashboard based on their role (e.g., admin, department head, staff).

Inventory Management Tasks:

Adding Items: Users can add new inventory items by filling out a form with details like item name, description, category, and unit. This data is submitted to the backend, where it is stored in the MongoDB database.

Updating Stock: Users can modify stock information, such as updating stock quantities after new supplies are added (via Stock In) or when items are issued to departments (via Stock Out).

Stock Requests:

Requesting Items: Users, such as department heads or staff, can submit stock requests, which are processed through the Stock Request Management system. These requests are placed in a queue using Redis, ensuring smooth task processing.

Admin Review: The admin reviews, approves, or rejects requests. If approved, items are issued to the requesting department.

Execution of Stock Movements:

Stock In and Stock Out: Once requests are approved, the stock movements (in and out) are executed. This information is updated in real-time, and the system ensures that inventory data remains accurate and up-to-date.

Real-Time Feedback and Reporting:

Real-Time Updates: The frontend receives real-time feedback from the backend whenever stock data changes. For example, when a stock request is approved or an item is issued, the dashboard is updated instantly to reflect the new data.

Past Submissions and Reports: Users can view the history of stock transactions (e.g., items added, issues made) through a simple interface. These records are fetched from the database, providing a comprehensive history of stock movements.

Notifications:

Email Notifications: For important events like successful registration, stock request approvals, or password recovery, the system triggers email notifications to keep users informed. These notifications are managed through the Email Service, ensuring that all relevant users receive timely updates.

This activity flow promotes a smooth, intuitive user experience, focusing on secure data handling, real-time updates, and easy access to past records. The system's design ensures that each user interacts with the platform seamlessly, whether they are requesting inventory, updating stock levels, or reviewing historical data.

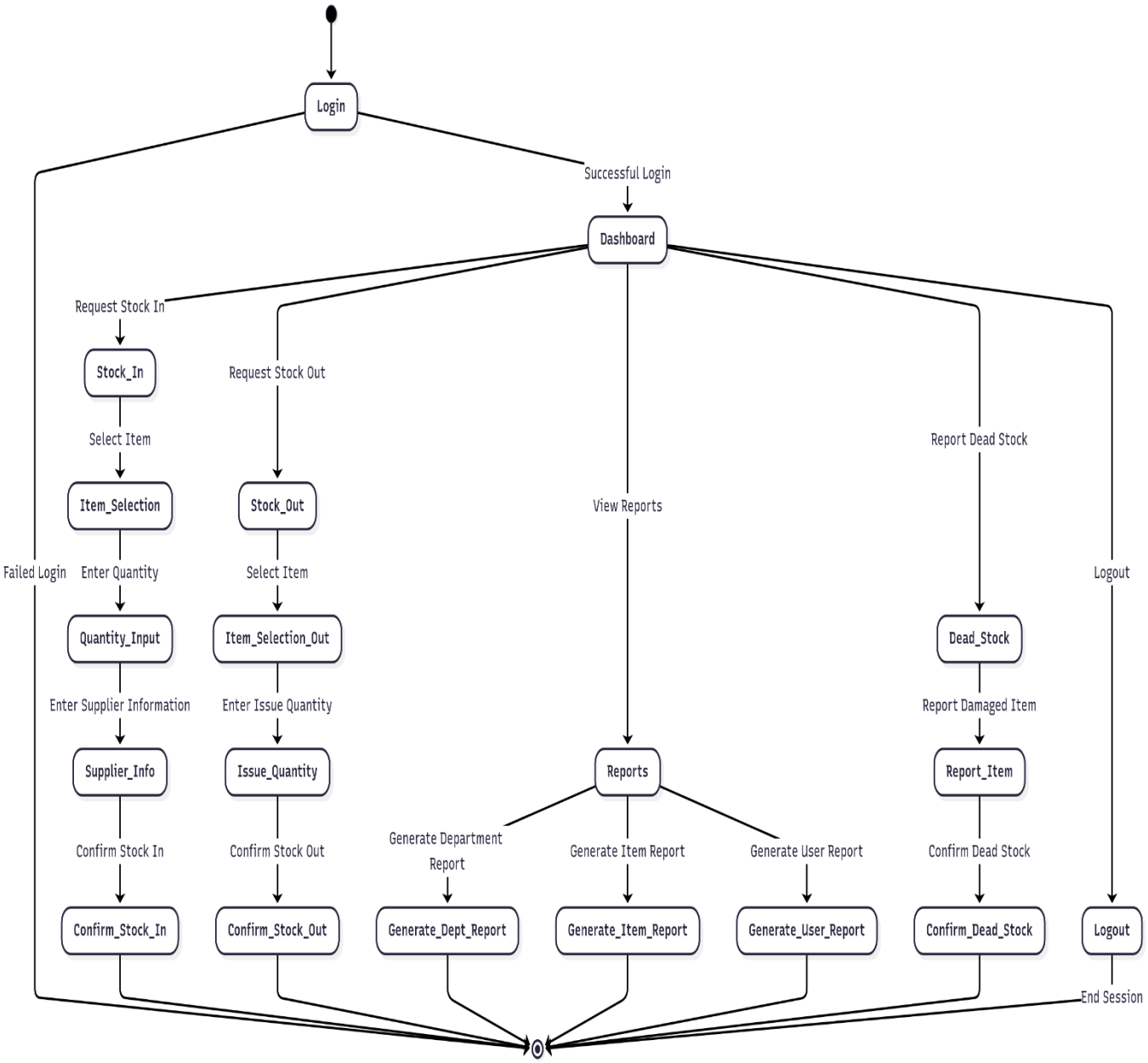


Fig 4.3. User Activity Diagram

4.2.4 Data Flow Diagram

The Data Flow Diagram (DFD) for the PSTU Inventory Management System visually represents how data flows through the system, showcasing the movement and transformation of information from one component to another. It provides an overview of the system's processes and how they interact with various data sources, inputs, and outputs.The DFD helps in understanding how different components (such as inventory management, stock tracking, user management, and reporting) communicate and exchange information. This diagram is crucial for identifying system processes, understanding data dependencies, and ensuring that all components work together efficiently.

4.2.4.1 Level 0 - Context Diagram

At the highest level, the Context Diagram provides a broad overview of the PSTU Inventory Management System. It shows the main external entities that interact with the system, along with a single process representing the entire system.

External Entities:

Admin: Admin manages users, inventory, and processes stock requests.

Department/Faculty Staff: Users who request, issue, and manage inventory within their departments.

Suppliers: External vendors who supply items for the inventory.

Email Service: Responsible for sending notifications about inventory requests, stock approvals, and updates.

Reports: Generated from the inventory data, providing insights into stock levels, transactions, and departmental needs.

System:

PSTU Inventory Management System: The central system that handles user requests, manages inventory, processes stock movements, and generates reports.

The context diagram typically includes the following processes:

User Authentication: Verifies login credentials and grants access based on user roles.

Inventory Management: Handles adding, updating, removing, and tracking items.

Stock Movement Management: Manages stock requests (in and out) and updates inventory accordingly.

Reporting: Generates reports on inventory levels, stock transactions, and departmental requests.

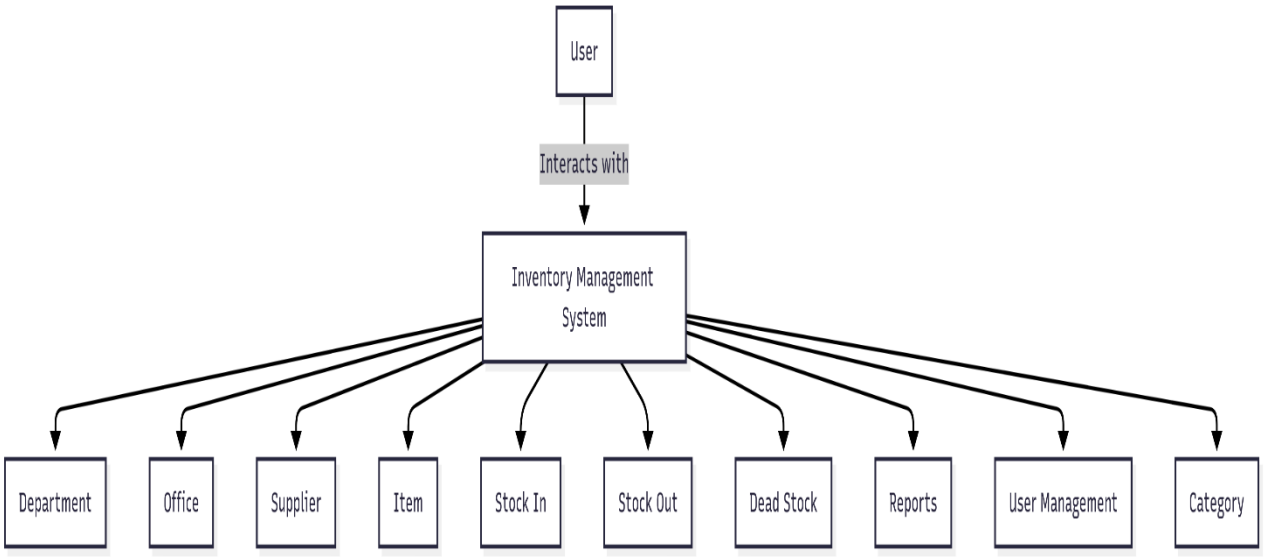


Fig 4.4. Level-0 Data Flow Diagram

4.2.4.2 Level 1 - Decomposition of Major Processes

In Level 1, the diagram breaks down the PSTU Inventory Management System into its major processes. These processes describe the main functional areas of the system and how they interact with data.

1. Process 1: User Authentication
    - Input: User credentials (username, password)
    - Output: Access token (JWT), user profile, role-based access permissions
    - Data Store: User Database (MongoDB)
2. Process 2: Inventory Management
    - Input: Item details (name, description, category, unit), stock data (quantity, supplier info)
    - Output: Updated inventory data, stock records, item details
    - Data Store: Item Database (MongoDB), StockIn Database, StockOut Database
3. Process 3: Stock Movement Management
    - Input: Stock requests (quantity, item type), stock data (current inventory levels)
    - Output: Updated stock data, issued items, stock-out records
    - Data Store: StockIn Database, StockOut Database
4. Process 4: Reporting
    - Input: Inventory and stock data
    - Output: Reports (department-wise, item-wise, office-wise)
    - Data Store: MongoDB (stock, transactions)
5. Process 5: Notifications (Email Service)
    - Input: Notification triggers (approval of stock request, stock updates)
    - Output: Email notifications (status of requests, confirmations)
    - External System: Email API (e.g., Gmail API)

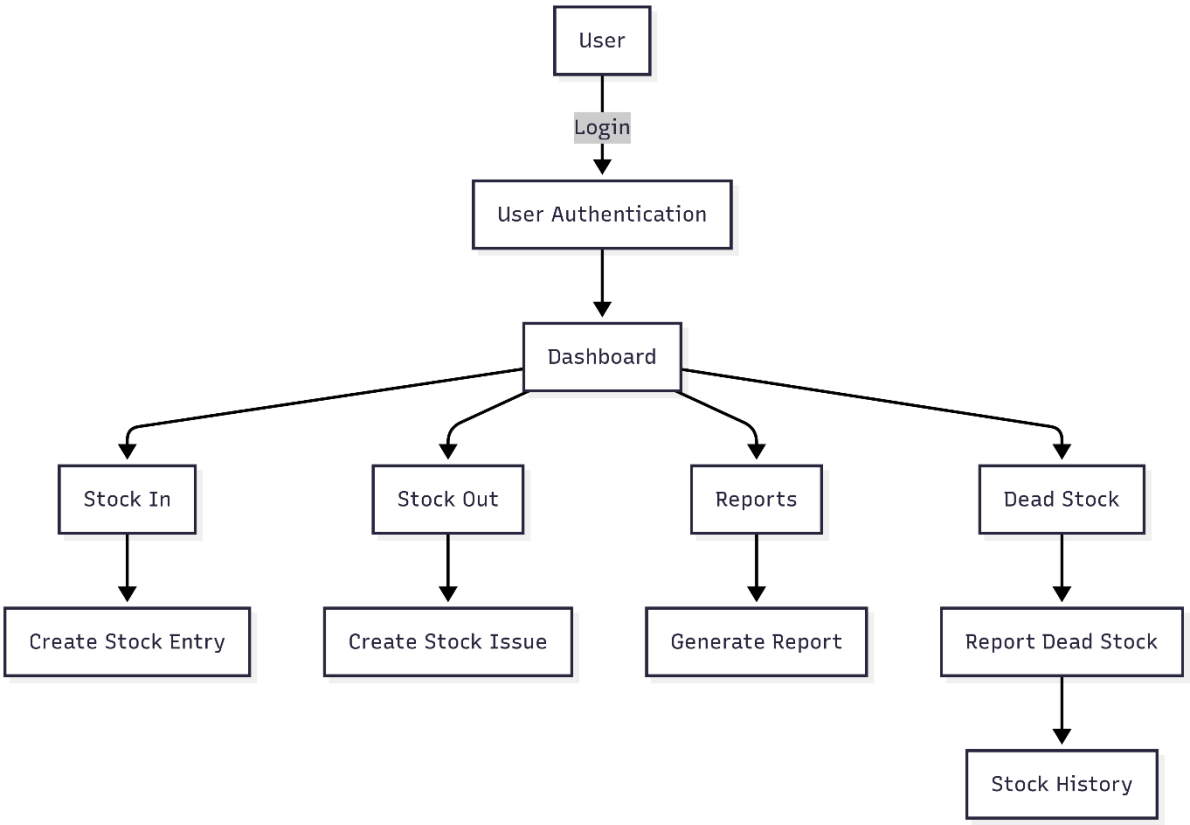


Fig 4.5. Level-1 Data Flow Diagram

4.2.4.3 Level 2 - Detailed Process Decomposition

At Level 2, we can further decompose specific processes into their detailed components and data flows. For example, Inventory Management can be broken down into the following:

Inventory Data Entry

Input: Item name, description, category, unit

Output: New item record

Data Store: Item Database

Stock Tracking and Updates

Input: Stock levels, stock movements (in and out)

Output: Updated stock data, stock history

Data Store: StockIn Database, StockOut Database

Data Flow for Key Operations:

Stock In Operation:

Input: Stock data (item name, quantity, supplier details)

Process: The StockIn Process receives the data, updates inventory levels, and records stock movement in the StockIn Database.

Output: Updated stock levels, transaction records.

Dead Stock Operation:

Input: Stock request (item name, quantity, department)

Process: The StockOut Process verifies available stock, processes the request, and updates inventory.

Output: Updated stock levels, transaction logs, stock-out records.

Reporting Process:

Input: Filtered stock data (department-wise, item-wise)

Process: The Reporting Process retrieves the relevant data, formats it, and generates reports for users.

Output: Printable reports, visual analytics (e.g., charts).

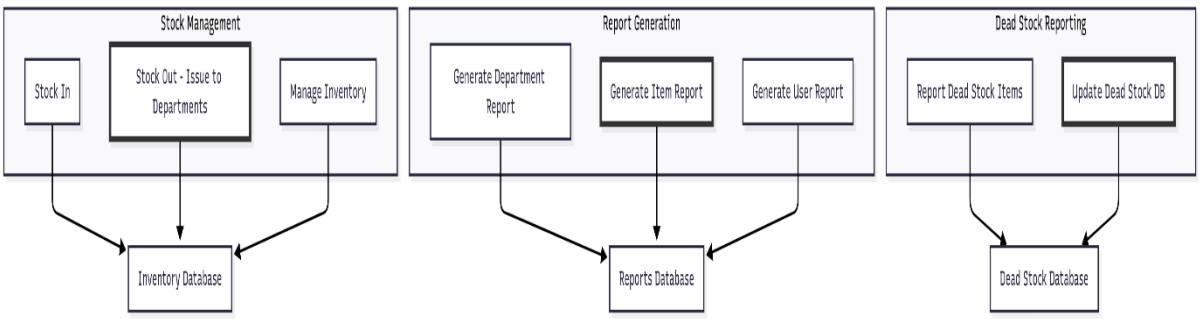


Fig 4.6. Level-2 Data Flow Diagram

4.3 Application Features

The PSTU Inventory Management System is designed to offer a range of features that facilitate efficient inventory management and resource tracking across departments, offices, and faculties at Patuakhali Science and Technology University (PSTU). These features aim to provide a seamless experience for administrators, faculty members, and staff, making inventory management, stock tracking, and reporting straightforward and efficient. Below are the key features of the application:

4.3.1 Inventory Management

The core functionality of the system revolves around inventory management, ensuring that all resources are tracked and updated efficiently:

Real-Time Stock Updates: The system tracks stock levels in real time, providing users with up-to-date information on available resources across departments and offices.

Item Management: Users can add, edit, or remove inventory items with ease. Each item is categorized, and detailed information (e.g., description, unit, supplier) is maintained.

Stock In/Out Management: Authorized users can record stock entries (when items are received) and stock withdrawals (when items are issued), with all actions logged for traceability.

Role-Based Access Control: Only authorized users, such as department heads or administrators, can perform sensitive actions like updating stock levels, issuing items, or generating reports.

4.3.2 Stock Request and Approval

One of the key aspects of the system is its ability to handle stock requests from departments, ensuring that items are issued or requisitioned when needed:

Request Creation: Department heads or staff can create stock requests for items needed, including quantity and department specifications.

Approval Workflow: The requests go through an approval process, allowing administrators or managers to approve or reject stock requests based on inventory availability and departmental needs.

Real-Time Notifications: Once a stock request is approved or rejected, the requester is notified via email, ensuring seamless communication and transparency.

4.3.3 Reporting and Analytics

The reporting module enables users to generate insightful reports based on inventory data. This is essential for decision-making, auditing, and ensuring efficient resource allocation:

Customizable Reports: Users can generate reports based on various filters such as department-wise usage, item-wise stock levels, date-wise transaction history, and more.

Printable Reports: Reports are printable with the PSTU logo and official format, making it easy to share hard copies for administrative or auditing purposes.

Visual Analytics: The dashboard provides graphical representations of inventory usage, including bar charts and pie charts for easy visualization of stock levels, trends, and departmental needs.

4.3.4 User Dashboard

The User Dashboard provides a centralized view of a user's activities and the overall system status, enabling efficient management of inventory tasks:

Recent Transactions: Users can view a list of recent stock movements (both stock in and stock out), including the date, quantity, item, and status (e.g., "approved," "pending").

Stock Overview: The dashboard includes a quick summary of the current stock levels across all departments and offices, allowing users to quickly identify items that are running low or require replenishment.

Notifications: Users are notified about key activities like stock updates, pending requests, or system announcements.

4.3.5 Item Categorization and Search

The system provides categorization and search functionality to make managing a large inventory more efficient:

Category Management: Items are categorized based on type (e.g., office supplies, laboratory equipment, furniture), making it easier for users to find the resources they need.

Search Functionality: Users can search for specific items, view detailed item information, and manage stock levels based on the item's category, name, or supplier.

4.3.6 Supplier and Procurement Management

The supplier management feature allows departments to keep track of suppliers and streamline procurement processes:

Supplier Information: Each supplier's contact details, address, and inventory items they supply are stored for easy reference.

Procurement Tracking: Departments can track procurement orders from suppliers, including order status, expected delivery dates, and received items.

Stock Purchase History: The system maintains a history of all stock purchases, linking them to suppliers for easy tracking and future procurement planning.

4.3.7 Security and Data Integrity

Security is a primary consideration in the design of the system, ensuring that inventory data is kept secure and accessible only to authorized personnel:

Role-Based Access Control (RBAC): Users are assigned roles (e.g., Admin, Staff, Department Head) with access rights tailored to their responsibilities within the system.

Data Encryption: Sensitive data (e.g., passwords, user information) is encrypted for added security.

Audit Logs: All critical actions, such as stock updates and requests, are logged with timestamps, providing an audit trail for transparency and accountability.

4.4 User Interface (UI) Design

The user interface (UI) of the PSTU Inventory Management System is developed using React.js and styled with Tailwind CSS, prioritizing simplicity, responsiveness, and ease of use. The UI design aims to provide an intuitive and seamless experience for all users, including students, faculty, and administrative staff. Whether interacting with the system for stock management, report generation, or inventory tracking, users can easily navigate the platform with minimal friction.

4.4.1 Home Page

The home page of the PSTU Inventory Management System serves as the primary entry point for users, offering a simple and intuitive interface. Upon accessing the platform, users are presented with clear options to log in**,** register, or learn more about the system's features. The design is clean and uncluttered, guiding users effortlessly to their next actions. With responsive design built using React.js and Tailwind CSS, the home page adapts seamlessly to various devices, from desktops to tablets and smartphones, ensuring a smooth and consistent experience across all platforms. Whether users are new or returning, they can easily navigate to the registration form or log in with a single click, providing a user-friendly and accessible start to their journey on the platform.

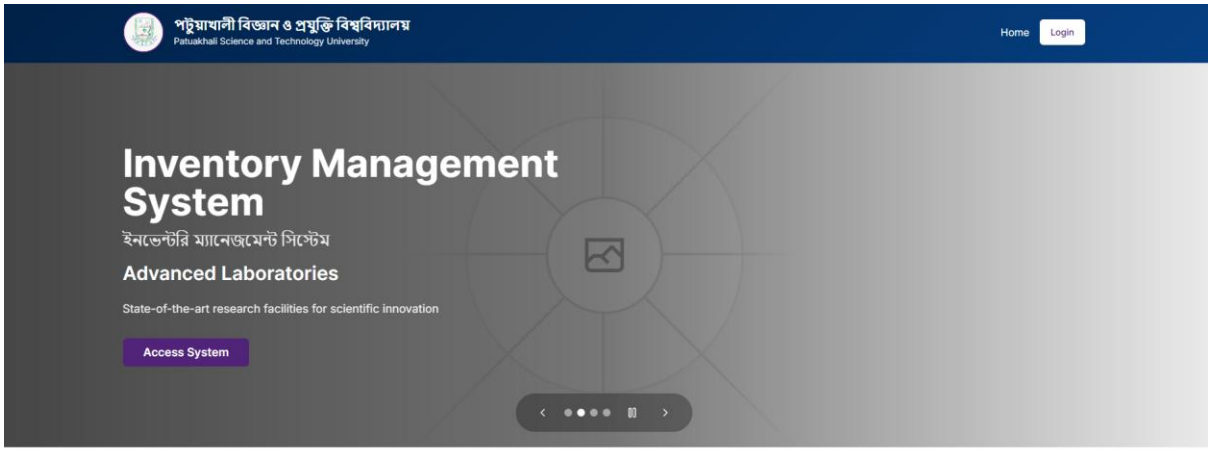


Fig 4.7. Home Page

4.4.2 Login Page

The login page of the PSTU Inventory Management System provides a simple and secure interface for users to access their accounts. It includes input fields for email/username and password, allowing users to easily log in. The page is designed with minimal distractions and includes options for password recovery in case of forgotten credentials. It is fully responsive, ensuring users can log in seamlessly from any device, whether on a desktop, tablet, or smartphone, offering a smooth and consistent experience across all platforms.

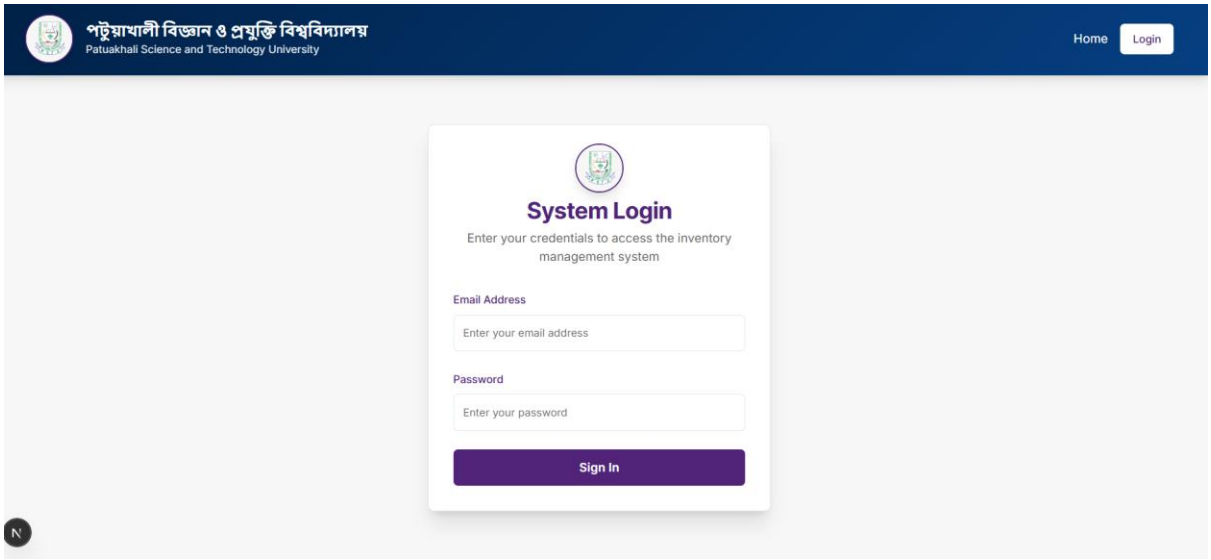


Fig 4.8. Login Page

4.4.3 Admin Dashboard

The admin dashboard of the PSTU Inventory Management System offers a centralized interface for managing the system's various functionalities. It provides administrators with quick access to key features, such as managing users, overseeing stock entries and issues, generating reports, and monitoring system activity. The layout is clean and organized, with intuitive navigation that allows for efficient management tasks. The dashboard is fully responsive, ensuring that administrators can easily manage and oversee operations from any device, whether desktop, tablet, or mobile, providing a seamless and effective user experience across all platforms.

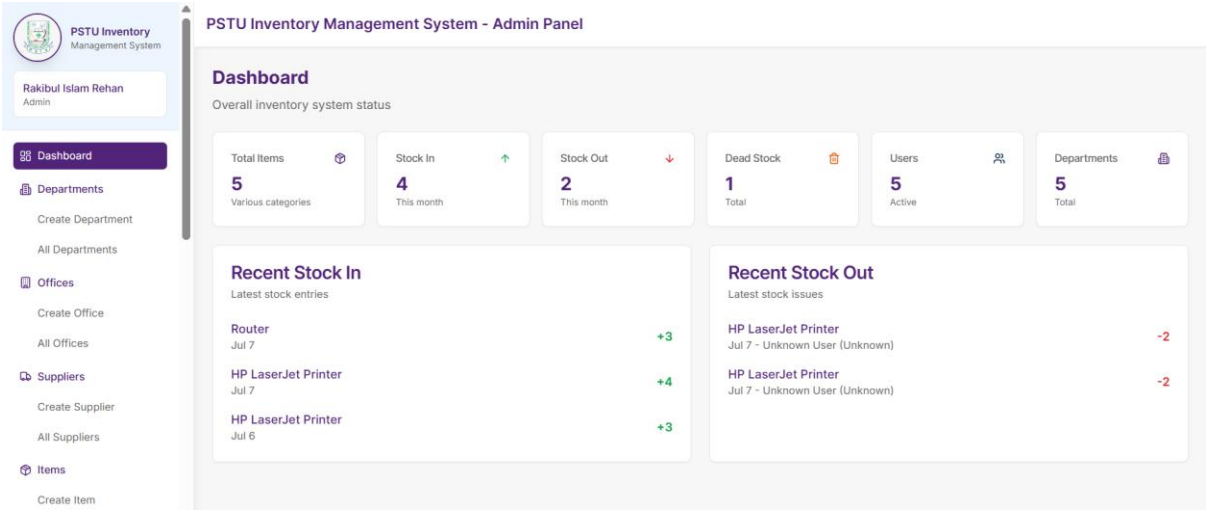


Fig 4.9. Admin Dashboard

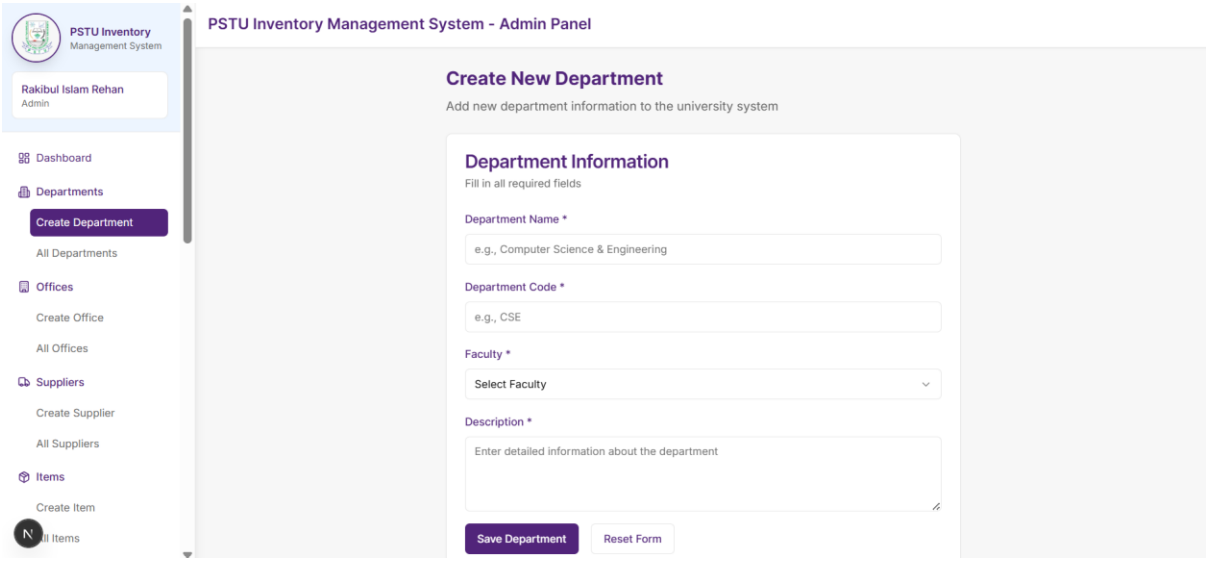


Fig 4.10. Create New Department Page

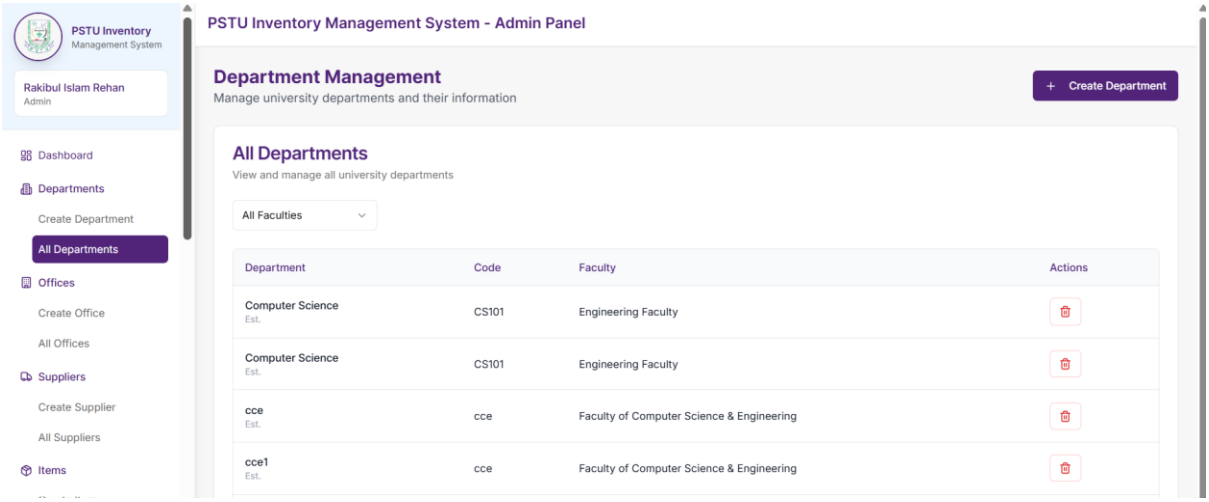


Fig 4.11. All Department Page

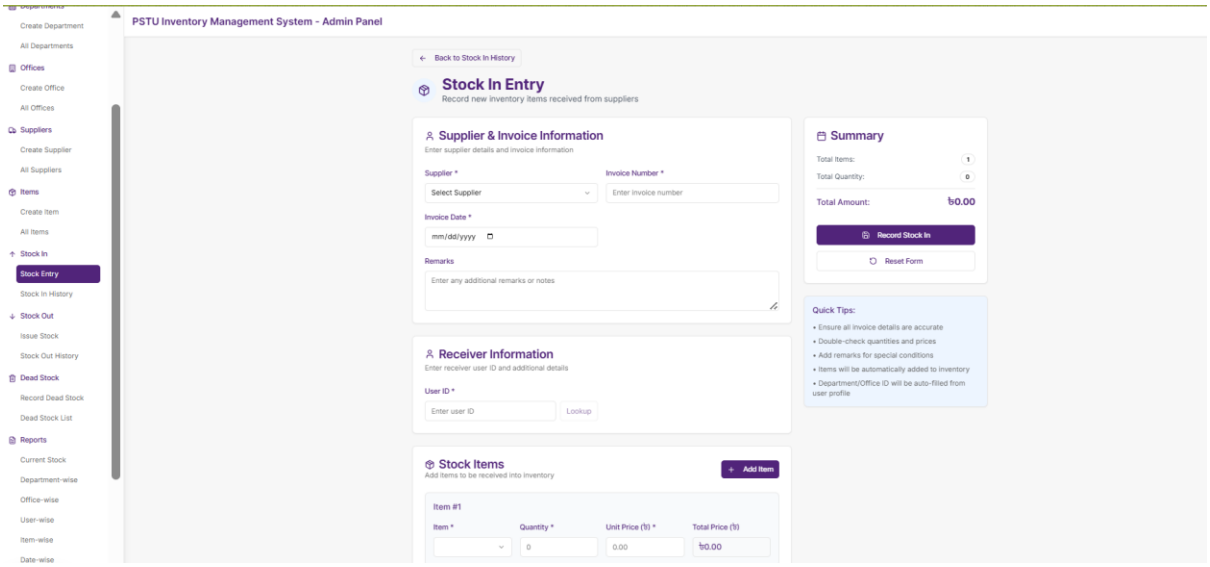


Fig 4.12. Stock Entry Page

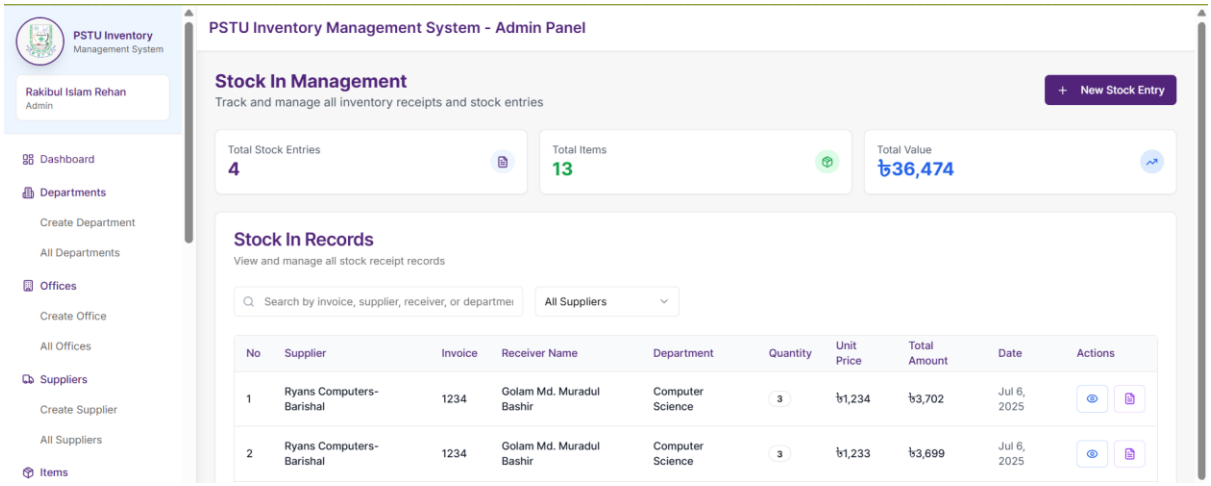


Fig 4.13. Show All Stock In page

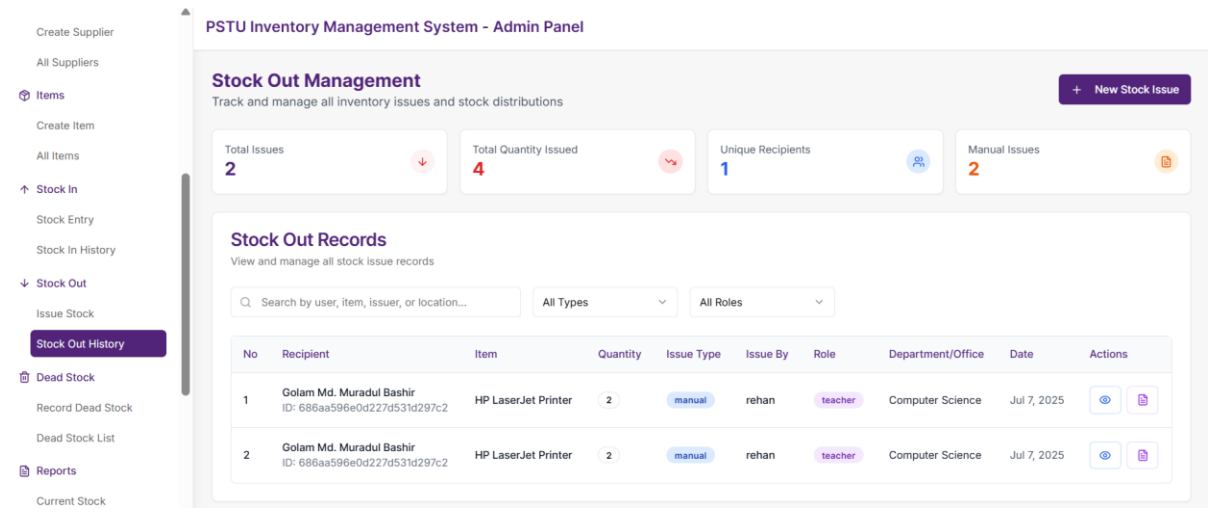


Fig 4.14. Stock Out Page

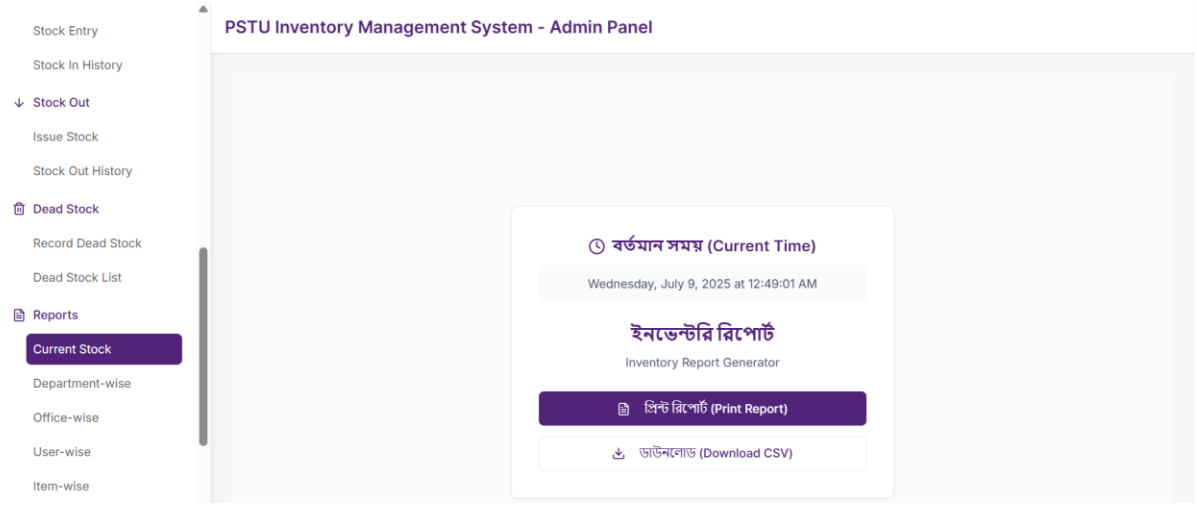


Fig 4.15. Current Stock Report Page

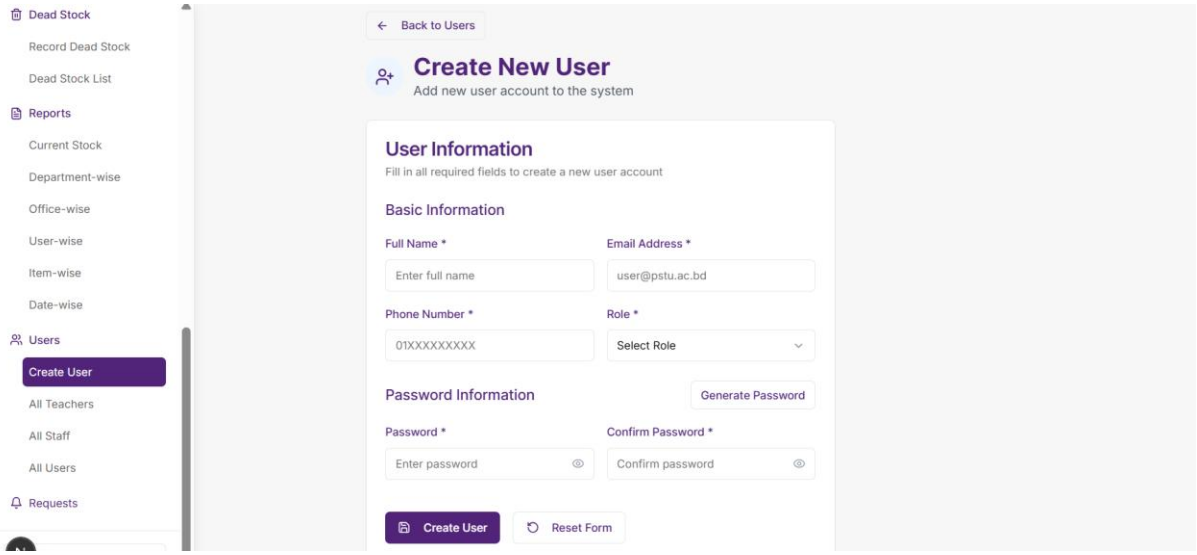


Fig 4.16. New User Create Page

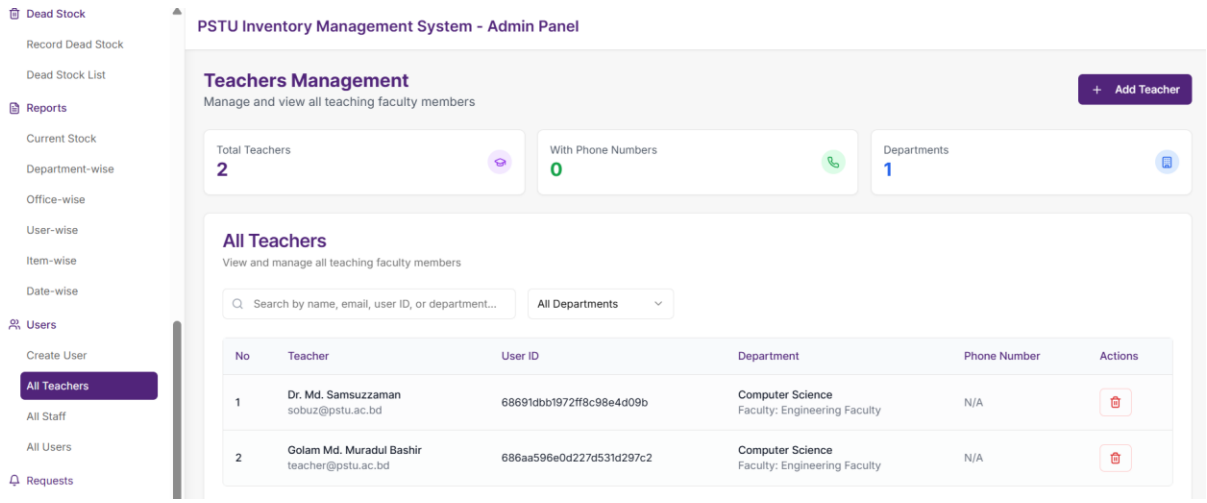


Fig 4.17. All TeacherPage

4.4.4 User Dashboard

The user dashboard of the PSTU Inventory Management System offers a personalized and intuitive interface for users to manage their tasks. It provides quick access to essential features such as viewing stock requests, monitoring stock status, generating reports, and tracking their activity within the system. The layout is simple and well-organized, allowing users to navigate between tasks with ease. The dashboard is fully responsive, ensuring a smooth and consistent experience across all devices, whether accessed on a desktop, tablet, or smartphone, allowing users to efficiently manage their activities anytime, anywhere.

4.5 Implementation

The PSTU Inventory Management System has been developed using a combination of modern technologies and development tools to ensure scalability, security, and maintainability. Below is a detailed explanation of the backend and frontend technologies, along with the implementation strategies used for building the system.

4.5.1 Backend

The backend of the PSTU Inventory Management System is built using a set of robust technologies to ensure scalability, security, and ease of maintenance. The system is structured to handle large amounts of data efficiently and securely, ensuring seamless communication between users and the database. Below is a detailed explanation of the backend technologies and how different modules were developed:

- Node.js:
    - Node.js serves as the runtime environment for the backend, providing an efficient, non-blocking, event-driven architecture. It allows the server to handle multiple requests simultaneously, making it ideal for building scalable and high-performance applications.
    - Version: `16.x`
- Express.js:

    - Express.js is used as the web framework to manage routing, middleware, and request handling in the backend. It simplifies the development of RESTful APIs and helps facilitate communication between the frontend and backend.
    - Version: `5.x`
- MongoDB:
    - MongoDB is a NoSQL database used to store and manage data such as users, items, inventory movements, and transactions. It provides a flexible and scalable solution for handling large volumes of data without predefined schemas, making it perfect for a dynamic system like the PSTU Inventory Management System.
    - MongoDB Atlas is used for cloud-hosted MongoDB services, ensuring easy management and scalability.
    - Version: `5.x`
- Mongoose:
    - Mongoose is used as an Object Data Modeling (ODM) library to interact with MongoDB. It provides a higher-level abstraction for defining and managing data models, validating schema, and performing CRUD operations on the database.
    - Version: `6.x`
- Redis:
    - Redis is used as a message broker and caching mechanism. It helps manage asynchronous queues for processing stock requests, inventory updates, and notifications. Redis improves the system's performance by allowing background tasks like sending emails or processing stock movements without blocking the main application flow.Version: `6.x`

- JWT (JSON Web Tokens):
    - JWT is used for user authentication and session management. It securely manages user sessions by generating access and refresh tokens, ensuring that only authenticated users can access specific resources.
    - Version: `9.x`
- Postman:
    - Postman is used for testing the API endpoints during development. It allows developers to simulate requests, check API responses, and ensure the backend services are working as expected before deployment.
    - Version: `8.x`
- Zod:
    - Zod is used for schema validation in the backend. It ensures that incoming data (such as user input or stock movement data) adheres to the expected format, preventing errors and ensuring data integrity.
    - Version: `3.x`
- Vaul:
    - Vaul is a security-focused library used for managing sensitive data, such as user credentials and tokens. It ensures that the system adheres to best practices in data security.
    - Version: `0.9.6`

Key Backend Functionalities:

User Registration:Users can register by providing necessary details (name, email, password). The system uses bcryptjs to hash passwords before saving them in the database. After registration, an email job is placed in the Redis queue to send a confirmation email to the user.

Stock Request Processing:When a stock request is made, it is queued in Redis. A worker service processes the request, updates the inventory, and records the stock transaction (either stock-in or stock-out).The backend ensures **real-time updates** of stock levels and verifies each stock transaction.

Admin Dashboard:The backend provides APIs to manage user accounts, monitor inventory, and review stock movements. Admin users can access comprehensive reports on stock levels, department-wise usage, and pending requests.

Email Notifications:Notifications related to stock requests, updates, and account changes (like password recovery) are sent through the Email Service. The Redis queue ensures that email sending tasks are processed asynchronously, without interrupting critical operations.

4.5.2 Frontend

The frontend of the PSTU Inventory Management System is developed using React.js to provide a dynamic and user-friendly interface for managing inventory, stock requests, and reports. The frontend is designed to provide a seamless experience for administrators, department heads, and staff, making the system easy to use and highly responsive.

- React.js:
  - React.js is used to build the user interface, enabling a component-based architecture that simplifies maintenance and enhances scalability. React's state management features help in providing real-time updates and dynamic interaction with the system's data.
  - Version: `17.x`
- Next.js:
  - Next.js is utilized for server-side rendering (SSR), making the system more SEO-friendly and improving performance by pre-rendering pages on the server.
  - Version: `10.x`
- Tailwind CSS:
  - Tailwind CSS is used for styling the application, providing a utility-first CSS framework that allows for rapid design and responsive layouts. Tailwind helps ensure that the user interface adapts to different screen sizes and devices.
  - Version: `2.x`
- TypeScript:
  - TypeScript is used to add type safety and reduce runtime errors. The use of TypeScript ensures that the application's components, services, and functions adhere to defined types, enhancing maintainability and readability.
  - Version: `4.x`
- Radix UI:

    - Radix UI provides accessible, low-level UI components like modals, tooltips, and dropdowns, enabling developers to build consistent and accessible user interfaces.
    - Version: `1.x`
- Axios:
    - Axios is used to make asynchronous HTTP requests to the backend API. It helps handle API calls for data fetching, submitting stock requests, and retrieving inventory information.
    - Version: `0.21.x`

Key Frontend Components:

1. User Registration & Login:

The registration form allows users to enter personal details, including their profile image, which are sent to the backend for account creation. Upon successful registration, users receive a confirmation email. The login system uses JWT tokens for secure authentication.

2. Inventory Management Interface:

The inventory management interface enables users to add, update, and view inventory items. It supports functions like stock in/out and manages real-time updates of stock levels.

3. Admin Dashboard:

The admin dashboard displays inventory analytics, recent stock movements, and pending stock requests. It provides admins with a graphical view of stock levels, usage patterns, and system health.

4. Submissions Management:

This feature allows users to track the status of their inventory-related submissions, such as requests for new stock, and view feedback on their actions. It helps keep all users up to date on inventory activities.

The PSTU Inventory Management System uses a combination of robust backend technologies like Node.js, Express.js, and MongoDB, along with a dynamic and user-friendly frontend built with React.js, Tailwind CSS, and Next.js. The backend leverages Redis for queue management and Docker for containerized operations, ensuring scalability and security. By using TypeScript, the system ensures better maintainability and fewer runtime errors. The system's implementation is designed to meet the university's needs for efficient inventory tracking, real-time updates, and secure data management.

Chapter 5: Testing & Security

5.1 Introduction

This chapter focuses on the testing strategies and security measures implemented in the PSTU Inventory Management System. Ensuring a secure and reliable environment is critical for both administrators and users, as the system handles sensitive inventory data, manages user access, and processes requests related to stock movements. This chapter outlines the security strategies, testing methodologies, and best practices followed during the development and deployment of the system to maintain data integrity, protect sensitive information, and provide a secure user experience.

5.2 Security Strategies

To maintain a secure environment, several strategies were employed throughout the design and development of the PSTU Inventory Management System:

5.2.1 Authentication and Authorization

JWT (JSON Web Tokens): JWT is used for user authentication in the system. When users log in, they receive an access token, which is used to verify their identity during subsequent requests. This ensures that only authorized users can access specific features and data.

Role-Based Access Control (RBAC): RBAC is implemented to restrict access to sensitive features based on the user's role. For instance, admin users have full access to the system's features, while department heads and staff have restricted access based on their responsibilities.

5.2.2 Sandboxed Code Execution

The system leverages postman to create a secure environment for handling sensitive data and stock movements. Each operation (like stock in, stock out, or request approval) is executed in an isolated environment, ensuring that any errors or potentially harmful actions do not impact the overall system.This approach helps maintain system integrity while providing a safe environment for sensitive data manipulation and processing.

5.2.3 Rate Limiting and API Throttling

Rate Limiting: To prevent abuse of the system, rate limiting is implemented on API endpoints. This ensures that users do not overload the server by making too many requests in a short period, such as excessive stock requests or login attempts.

API Throttling: API throttling ensures that no user exceeds a set number of requests per minute. This protects the system from denial-of-service (DoS) attacks and ensures that resources are evenly distributed among users.

5.2.4 Data Encryption

Encryption of Sensitive Data: Sensitive information such as passwords, stock transaction details, and user data is encrypted before being stored in the database. This ensures that user and transaction data remains secure, even if unauthorized access is gained.

HTTPS: All communications between the frontend and backend are encrypted using HTTPS, ensuring that data is securely transmitted between users and the server without being intercepted by malicious third parties.

5.3 XP Practices

The development of the PSTU Inventory Management System adhered to Extreme Programming (XP) practices, emphasizing continuous testing, collaboration, and iterative development. XP practices helped ensure that the system was built with quality and adaptability in mind, allowing for easy integration of new features and resolution of issues.

5.3.1 Test-Driven Development (TDD)

TDD was employed to write tests before implementing new features. This approach allowed developers to identify potential issues early in the development process, ensuring that new features were fully tested and worked as expected.

Tests were written for both the backend and frontend components of the system, covering scenarios such as:

- User authentication and role management
- Stock request processing and approval
- Database interactions and CRUD operations
- Data validation for inventory transactions

5.3.2 Continuous Integration (CI)

Continuous Integration (CI) pipelines were integrated to automate the testing process whenever new code was pushed to the repository. This ensured that any new code met the required quality standards before being deployed to the production environment.

Automated tests included:

Unit tests for individual functions (e.g., stock transaction logic, report generation).

Integration tests for API endpoints (e.g., stock request creation and approval).

UI tests for frontend components (e.g., login form validation, inventory display).

5.3.3 XP Testing

XP testing practices were employed to ensure that the PSTU Inventory Management System met high standards of quality and functionality. Key aspects of the testing process included.

5.3.4 Unit Testing

Unit tests were developed for the backend functions using Jest to ensure that each module performed as expected. These tests validated the core features, such as:

User authentication logic

Stock request handling

Database CRUD operations

The frontend components were tested using React Testing Library, ensuring that UI elements such as forms, buttons, and error messages behaved correctly.

5.3.5 Integration Testing

Integration tests were conducted to ensure that different parts of the system worked together seamlessly. For example:Testing the interaction between the frontend and backend, such as user registration, login, and stock request processes.Verifying that the API endpoints correctly processed requests, interacted with the database, and returned the appropriate responses.

5.3.6 End-to-End (E2E) Testing

End-to-End (E2E) testing was conducted using Cypress to simulate real user interactions with the system. E2E tests helped verify the overall workflow of the system, including:

- Stock request creation and approval
- Inventory updates
- User login and dashboard interactions

These tests helped identify issues that could only be detected when testing the entire workflow, ensuring that users had a smooth and error-free experience when interacting with the system.

5.4 Summary

This chapter outlined the security strategies and testing methodologies implemented in the PSTU Inventory Management System. Security strategies such as JWT-based authentication, sandboxed code execution, rate limiting, and data encryption ensure that the system is secure and resilient against common threats. The adoption of XP practices, including Test-Driven Development (TDD) and Continuous Integration (CI), ensured that the system was developed with a focus on quality, reliability, and adaptability. The comprehensive testing approach, which includes unit testing, integration testing, and end-to-end testing, ensures that the system functions as expected and provides a secure, seamless experience for users.

Chapter 6: Conclusion

6.1 Introduction

This chapter provides an overview of the outcomes achieved through the development of the PSTU Inventory Management System, along with insights into potential future enhancements. The project aimed to streamline inventory management processes at Patuakhali Science and Technology University (PSTU) by replacing the outdated pen-and-paper system with a modern, automated platform. This chapter discusses the results of the project, highlights key takeaways, and outlines possible areas for future improvement.

6.2 Project Outcomes

The PSTU Inventory Management System has successfully achieved its primary goals, offering a range of functionalities that enhance the efficiency of inventory tracking and management at the university. Key outcomes of the project include:

Automated Inventory Management: The system automates the process of managing stock levels, significantly reducing manual errors and saving time for administrative staff. Inventory items are now updated in real time, making the process more efficient and accurate.

Real-Time Stock Updates**:** The system allows users to track inventory levels in real time, ensuring that departments are always aware of available resources. This prevents stockouts and overstocking, making resource allocation more effective.

Role-Based Access Control**:** By implementing role-based access control, the system ensures that only authorized users can perform specific actions, such as adding items, updating stock levels, or generating reports. This ensures data security and integrity.

User-Friendly Interface**:** The system is designed with an intuitive and responsive interface that simplifies inventory management for both technical and non-technical users. Staff can easily navigate the system to add items, submit stock requests, and generate reports.

Cost-Effective Solution**:** The PSTU Inventory Management System provides a self-hosted solution, which reduces the ongoing costs associated with third-party software subscriptions. The use of open-source technologies makes the system a cost-effective option for the university.

6.3 Future Works

While the PSTU Inventory Management System has achieved its primary objectives, there are several areas where further enhancements can be made:

Mobile App Development**:** Developing a mobile app version of the system would allow users (such as department heads and staff) to access and manage inventory data on-the-go, providing greater flexibility and convenience.

Integration with Other University Systems: Future updates could include integration with other university systems, such as procurement or financial management systems, for a more comprehensive resource management solution.

Expansion of Reporting Features**:** The system could be enhanced with additional reporting capabilities, including visual analytics (charts, graphs) and exportable reports (e.g., PDF, Excel). This would enable administrators to gain deeper insights into inventory usage patterns and trends.

Improved Search Functionality**:** Enhancing the search functionality to allow users to filter and search for items based on multiple criteria (e.g., category, supplier, department) would improve the overall user experience.

Community Contributions**:** As an open-source project, future development could include contributions from the larger developer community. This could help expand and improve the system's features, ensuring it remains relevant and adaptable to the needs of educational institutions.

6.4 Summary

The PSTU Inventory Management System has successfully created a secure, efficient, and user-friendly platform for managing inventory at Patuakhali Science and Technology University. By automating stock tracking, streamlining procurement requests, and providing real-time updates, the system has addressed key challenges faced by administrative staff and faculty. Although there are opportunities for further development, the current version of the system represents a significant step forward in modernizing resource management at PSTU. The outlined future work provides a roadmap for continued enhancements, ensuring that the system evolves to meet the changing needs of the university.

References

[1] Mongoose, "Mongoose ODM Documentation," 2024. [Online]. Available: https://mongoosejs.com/.

[2] Node.js Foundation, "Node.js Official Documentation," 2024. [Online]. Available: https://nodejs.org/.

[3] Express.js, "Express.js Documentation," 2024. [Online]. Available: https://expressjs.com/.

[4] MongoDB, Inc., "MongoDB Documentation," 2024. [Online]. Available: https://docs.mongodb.com/.

[5] Redis Ltd., "Redis Official Documentation," 2024. [Online]. Available: https://redis.io/.

[6] Docker, Inc., "Docker Official Documentation," 2024. [Online]. Available: https://docs.docker.com/.

[7] React.js, "React Documentation," 2024. [Online]. Available: https://reactjs.org/.

[8] Tailwind CSS, "Tailwind CSS Official Documentation," 2024. [Online]. Available: https://tailwindcss.com/.

[9] Zod, "Zod Schema Validation Documentation," 2024. [Online]. Available: https://zod.dev/.

[10] MongoDB Atlas, "MongoDB Atlas Documentation," 2024. [Online]. Available: https://www.mongodb.com/cloud/atlas.

[11] Postman, "Postman Official Documentation," 2024. [Online]. Available: https://www.postman.com/.

[12] Vaul, "Vaul Security Documentation," 2024. [Online]. Available: https://www.vaultproject.io/.

[13] Tailwind Merge, "Tailwind Merge Documentation," 2024. [Online]. Available: https://tailwind-merge.com/.

[14] JavaScript Info, "JavaScript Guide," 2024. [Online]. Available: https://javascript.info/.

[15] Amazon Web Services (AWS), "AWS Documentation," 2024. [Online]. Available: https://aws.amazon.com/.

[16] D. Johnson, *API Design Patterns and Best Practices*, 1st ed., O'Reilly Media, 2024.