**COSC 612/ AIT 624**

**PROJECT TITLE:** FABRIC INVENTORY MANAGEMENT SYSTEM

**GROUP NAME:** CODE CRAFTERS

**GROUP MEMBERS:**

HALEEMAH AMISU

JOSEPH FRISHKORN

AASTHA BHATT

BHUVAN ANGIRAA

SAI MANASWINI UTLA

**SPRING 2025**

**Section 1: Planning and Scheduling**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assignee Name** | **Email** | **Task** | **Duration (hours)** | **Dependency** | **Due date** |
| Haleemah Amisu | hamisu1@students.towson.edu | Build backend, database connectivity back and front end, setting up front end, update kanban board | 6 | Amazon Web Services | 4/21/25 |
| Joseph Frishkorn | jfrish1@students.towson.edu | Report format, assisting with system modeling as required | 3 | All sprint tasks | 4/21/25 |
| Aastha Bhatt | abhatt6@students.towson.edu | Updated two sequence diagrams based on feedback. Created 2 new sequence diagrams. Made diagrams for the Architectural view (Logical, process, developmental, physical) | 5 | Revise and Refine System | 4/21/25 |
| Bhuvan Angiraa | bangira1@students.towson.edu | Test Specifications and test cases for the two major use cases. | 5 | System requirements, use cases | 4/21/25 |
| Sai Manaswini Utla | sulta1@students.towson.edu | Implementation of Test cases and updated the use case diagrams according to the requirement | 3 | Test cases & Test Specifications | 4/21/25 |

**Section 2: Problem Statement**

**What is the Fabric Inventory Management System?**   
It is a smart tool that helps businesses in textiles, fashion, upholstery, and manufacturing keep track of their fabric stock. It ensures they always know what’s available, how much is being used, and when to restock, helping to prevent shortages and reduce waste.    
   
**Whom is it for?**   
It is designed for businesses and professionals involved in fabric-related industries. It is particularly useful for textile manufacturers, fashion designers, wholesale fabric distributors and for the businesses which work or deal with fabric.   
   
**What problem does it solve?**   
It solves problems and challenges faced by businesses handling fabric such as stock mismanagement, manual errors, slow & inefficient working environment, preventing wastage & loss of resources and many more.

**What alternatives are available?**   
There are many alternatives available to this such as manual tracking methods which include spreadsheets and physical logbooks. They also depend on the scale/size of the business and the needs like **Zoho**, which is a cloud-based inventory system used in various industries as well as in the fabric industries.

**Why is this project compelling and worth developing?**   
**Solves Real Industry Challenges** - stock mismanagement, wastage, and inefficiencies, Reduces Costs & Increases Efficiency, High Demand Across Multiple Industries, Potential for Business Growth & Monetization, Integration with Emerging Technologies **–** use of AI & ML to predict the demand of fabrics.

**Primary objectives** – Automating fabric tracking, reduce wastage, increasing efficiency, optimizing inventory, enhance decision-making (data driven/demand driven)   
**Target Customers** – Textile manufacturers, retailers, wholesale distributors,

**Scope -** Includes inventory management, Order & supplier management, multi-user access. 

**Who are the competitors, and what is novel about your approach?**

Fabric Inventory Competitors:

1.Fabric Manager

2.Stitch Lab

3.BlueCherry ERP

4.Fishbowl Inventory

The websites mentioned above follow a similar approach to fabric inventory management, where fabric is entered into the system based on key attributes such as color, type, and quantity. They also track fabric usage and maintain up-to-date information regarding stock levels. These features are designed to ensure efficient inventory management and seamless operations within fabric-related businesses.

However, our website offers several unique features that set it apart:

1. **Chatbot Assistance**: We provide a chatbot that offers immediate assistance to users, helping them address common queries in real time. This feature improves user experience by providing quick, automated support for frequent questions, making it easier for users to navigate the system and resolve issues without needing to wait for manual help.
2. **Restock Requests**: When stock reaches a predefined threshold, our platform automatically sends restock requests to suppliers. This proactive feature ensures that fabric inventory is always replenished on time, reducing the chances of stockouts and improving overall operational efficiency.
3. **Role-Based Profiles and Access Control**: Our system allows users to create profiles based on their roles within the organization, providing customized access and permissions depending on their responsibilities. This feature enhances security and streamlines workflows, as different users can access only the parts of the system that are relevant to their tasks.
4. **Common Features**: Like other platforms, we offer essential features such as:
5. **Easy Search and Filter**: Users can quickly search, and filter fabrics based on various attributes like type, color, and other criteria, making it easy to find specific materials in large inventories.
6. **Detailed Fabric Catalog**: A comprehensive catalog displays all the available fabrics, with detailed information about each one, allowing users to view their options at a glance.
7. **Simple Stock Management**: Adding, editing, or deleting fabric stock is a simple process. Users can update inventory by filling out a straightforward form, ensuring that the system stays current with minimal effort.

These unique and common features together enhance user convenience, improve operational efficiency, and provide better control over fabric inventory management, making our platform a more comprehensive and user-friendly solution.

**Clearly demonstrate that the system can be built effectively using available resources and technology:**

This website can be built using a combination of available tools and technologies, ensuring a robust, scalable, and responsive platform for managing fabric inventory.

**Frontend:**

* **HTML, CSS, JavaScript**: These core technologies will form the foundation of the website’s user interface, providing a responsive and user-friendly design for staff to easily navigate and manage the fabric inventory. The front end will be optimized for both desktop and mobile devices, ensuring accessibility for all users.

**Backend:**

* **Python**: The backend is built using Python for its versatility and power. Python's extensive libraries and frameworks are beneficial for integrating machine learning models and handling complex backend processes. The language is widely recognized for its scalability and maintainability, making it an excellent choice for building the website's core functionalities.
* **Machine Learning**: Python is used to integrate machine learning algorithms, especially for the chatbot feature. These algorithms enable the chatbot to improve its responses over time by learning from the data it processes.
* **Libraries**: We use popular libraries like **TensorFlow**, **PyTorch**, and **scikit-learn** to build and train machine learning models for the chatbot. Supervised learning and reinforcement learning are common approaches employed to improve chatbot performance through continuous training and data analysis.

**Database:**

* **MySQL**: For efficient database management, we rely on MySQL, which will store and manage all the fabric inventory data. It enables structured queries, ensuring that the system can easily handle large volumes of data and allow users to access, update, and delete fabric stock as needed.

**Cloud Hosting and Services:**

* **AWS (Amazon Web Services)**: To ensure high performance, scalability, and security, we use AWS for hosting and managing the website. AWS offers cloud-based services that are well-suited for websites with growing demands, ensuring that the platform can scale seamlessly as the user base and inventory size increase.
* AWS provides essential infrastructure such as virtual servers (EC2), managed databases (RDS), and storage (S3), all of which contribute to maintaining smooth website performance.

**Artificial Intelligence or Machine Learning Component:**

AI/ML models analyze sales data to forecast demand, with key components including the AI/ML prediction engine. Additionally, conversational AI can interpret user queries, and this capability can also be integrated into a chatbot.

**Machine Learning for Chatbot:**

* **Chatbot Functionality**: The chatbot, powered by Machine Learning, continuously learns from user interactions. With supervised learning, the chatbot is initially trained on labeled data, allowing it to handle basic queries. Over time, through reinforcement learning, the chatbot adapts and improves its ability to address more complex or context-dependent queries.
* **Libraries**: TensorFlow, PyTorch, and scikit-learn are used to build the models that allow the chatbot to learn and provide increasingly accurate responses.

By combining these technologies, the website will not only offer seamless fabric inventory management but also provide an interactive, intelligent user experience through the chatbot. The integration of machine learning ensures that the platform remains innovative, efficient, and adaptive to the needs of users.

**What makes this project interesting from a technical perspective?**

 This project is technically fascinating due to its seamless integration of modern web technologies, advanced machine learning, cloud computing, and database management. The frontend, built with HTML, CSS, and JavaScript, offers a responsive and user-friendly interface, while the backend leverages Python for scalability, machine learning integration, and efficient data handling with SQL databases. AWS cloud hosting ensures optimal performance, scalability, and security. Unique features like an AI-powered chatbot, automatic restock requests, and role-based access control enhance user experience and streamline fabric inventory management. The chatbot continuously improves through machine learning algorithms using libraries like TensorFlow and PyTorch, while real-time updates and proactive inventory management further elevate the platform's efficiency. Combining traditional web development with cutting-edge AI techniques, this project creates a highly adaptive, intelligent, and scalable system, making it both technically challenging and innovative.

* **Functional Requirements**: Features and functions that describe what the system should do.
* **Non-Functional Requirements**: Qualities and performance attributes of the system that define how the system should perform.

|  |  |
| --- | --- |
| **Functional Requirements** | **Non-Functional Requirements** |
| 1. The system should ensure secure user authentication with role-based access. | The system should be secure and support role-based access for different user levels. |
| 2. The system should allow authorized users to add new fabric entries. | The system should have an intuitive user interface for easy navigation. |
| 3. Users should be able to update fabric stock when receiving shipments. | The system should provide quick responses for inventory queries and updates. |
| 4. The system should allow authorized users to remove fabric records if no dependencies exist. | The system should ensure high availability with minimal downtime. |
| 5. Users should be able to quickly search and filter fabric inventory. | The system should scale easily to handle increasing fabric inventory and transactions. |
| 6. The system should generate alerts when stock levels fall below a predefined threshold. | The system should process data securely and comply with relevant data protection regulations. |
| 7. The system should facilitate placing restock orders with suppliers. | The system should be able to handle large datasets for analytical reports without significant performance degradation. |
| 8. The system should generate analytical reports for inventory and sales. |  |
| 9. User access should be role-based, limiting actions based on predefined permissions. |  |
| 10. The chatbot should assist users with inventory-related queries. |  |
| 11. The system should evaluate supplier performance based on historical transactions. |  |

**Section 3: Requirements**

**User Requirements (no update required)**

Use Cases

|  |  |
| --- | --- |
| **Use Case no:** 1 | **Use Case Name:** User Login & Authentication |
| **Actors:** Staff Member (User) | |
| **Description:** A staff member logs into the system using their credentials. Upon successful authentication, they gain access to relevant features based on their role. | |
| **Alternate Path:** If login fails, the system prompts the user to retry or reset the password. | |
| **Pre-Condition:** The user must be registered in the system. | |

|  |  |
| --- | --- |
| **Use Case no:** 2 | **Use Case Name:** Add New Fabric to Inventory |
| **Actors:** Inventory Manager | |
| **Description:** The inventory manager adds a new fabric item to the system, entering details such as type, color, quantity, supplier, and price. | |
| **Alternate Path:** If data is incomplete, the system prompts the user to fill in missing fields. | |
| **Pre-Condition:** The user must have proper access to modify inventory. | |

|  |  |
| --- | --- |
| **Use Case no:** 3 | **Use Case Name:** Update Fabric Stock |
| **Actors:** Inventory Manager | |
| **Description:** A user updates the stock level of an existing fabric item after receiving a new shipment. | |
| **Alternate Path:** If the fabric does not exist, the system prompts the user to add it as a new entry. | |
| **Pre-Condition:** The fabric item must exist in the database. | |

|  |  |
| --- | --- |
| **Use Case no:** 4 | **Use Case Name:** Remove Fabric from Inventory |
| **Actors:** Inventory Manager | |
| **Description:** A user deletes a fabric entry from the system if it is discontinued or no longer available. | |
| **Alternate Path:** If the fabric is linked to pending orders, the system restricts deletion and prompts an alternative action. | |
| **Pre-Condition:** The user must have permission to remove inventory items. | |

|  |  |
| --- | --- |
| **Use Case no:** 5 | **Use Case Name:** Search & Filter Fabric Inventory |
| **Actors:** Staff Member | |
| **Description:** A user searches for fabrics by various filters such as type, color, supplier, or stock level. | |
| **Alternate Path:** If no results are found, the system suggests modifying search criteria. | |
| **Pre-Condition:** The database must contain fabric entries. | |

|  |  |
| --- | --- |
| **Use Case no:** 6 | **Use Case Name:** Generate Restock Alerts & Requests |
| **Actors:** System, Inventory Manager | |
| **Description:** The system automatically generates a restock alert when a fabric's stock level falls below the defined threshold. A restock request is then sent to the supplier. | |
| **Alternate Path:** The inventory manager can manually override the system’s suggestion if stock levels are sufficient. | |
| **Pre-Condition:** A restock threshold must be set for each fabric. | |

|  |  |
| --- | --- |
| **Use Case no:** 7 | **Use Case Name:** Process Supplier Orders |
| **Actors:** Supplier, Inventory Manager | |
| **Description:** The system allows users to place orders with suppliers when restocking is needed. | |
| **Alternate Path:** If the supplier is out of stock, the system suggests alternative suppliers. | |
| **Pre-Condition:** The supplier must be registered in the system. | |

|  |  |
| --- | --- |
| **Use Case no:** 8 | **Use Case Name:** View Sales & Inventory Reports |
| **Actors:** Business Owner, Inventory Manager | |
| **Description:** The system generates reports on fabric sales, inventory status, and usage trends. | |
| **Alternate Path:** If no data is available for the selected timeframe, the system informs the user. | |
| **Pre-Condition:** The system must have recorded transactions. | |

|  |  |
| --- | --- |
| **Use Case no:** 9 | **Use Case Name:** Role-Based Access Control |
| **Actors:** Admin, Staff Member | |
| **Description:** The system grants different access levels based on the user's role, restricting or allowing actions such as adding/deleting fabrics. | |
| **Alternate Path:** If an unauthorized user tries to perform restricted actions, an error message is displayed. | |
| **Pre-Condition:** User roles must be predefined. | |

|  |  |
| --- | --- |
| **Use Case no:** 10 | **Use Case Name:** Chatbot Assistance |
| **Actors:** User, AI Chatbot | |
| **Description:** A user interacts with the chatbot to ask questions about fabric availability, restock alerts, and order processing. | |
| **Alternate Path:** If the chatbot cannot answer a query, it redirects the user to human support. | |
| **Pre-Condition:** The chatbot must have trained responses or access to relevant system data. | |

|  |  |
| --- | --- |
| **Use Case no:** 11 | **Use Case Name:** Predict Fabric Demand Using AI |
| **Actors:** System, Business Owner | |
| **Description:** The AI module analyzes sales data and predicts future demand for fabrics, helping businesses optimize restocking. | |
| **Alternate Path:** If insufficient historical data exists, the system notifies the user. | |
| **Pre-Condition:** The system must have stored transaction history. | |

|  |  |
| --- | --- |
| **Use Case no:** 12 | **Use Case Name:** Track Supplier Performance |
| **Actors:** Business Owner, Inventory Manager | |
| **Description:** The system evaluates supplier performance based on factors like delivery speed, order accuracy, and quality. | |
| **Alternate Path:** If data is incomplete, the system prompts users to manually enter performance records. | |
| **Pre-Condition:** The system must have stored data on past supplier interactions. | |

**System Requirements**

Use Case 1: Login: Requirement number: 1

Inputs:

* Username
* Password

Requirement Description:

* Username Validation:
* The username must be checked for existence in the user's database.
* The username must not be empty.
* Password Validation:
* The password must be validated against the stored password hash in the database.
* The system must ensure the password is correct and matches the one stored.
* Validation:
* If the username or password is incorrect, return a message: "Invalid credentials."
* The user must be redirected to the login page upon failed login.
* Success:
* If the login is successful, the user will be granted access to the dashboard and a session will be created indicating they are logged in.
* Error Handling:
* Display an error message if the login fails: "Invalid credentials."

Outputs:

* Success Output: "Login successful"
* Error Output: "Invalid credentials."

Use Case 2: Inventory Management

Inputs:

* None (GET request)

Requirement Description:

* Login Check
* The user must be logged in before they can access the inventory page.
* If the user is not logged in, they should be redirected to the login page with an appropriate error message: "Please log in first."
* Inventory Data:
* Upon successful login, the system should query the fabric inventory from the database.
* The inventory data should be returned as a JSON object, containing a list of all fabric items with their attributes.
* Validation:
* The system should ensure that only authorized users (e.g., "Inventory Manager", "Admin") can access this page.
* Error Handling:
* If an error occurs while retrieving inventory data, return an appropriate message: "Unable to retrieve inventory data."

Outputs:

* Success Output: "Login successful"

Error Output: "Invalid credentials."

Use Case 3: Orders: Requirement number: 3

Inputs:

- None (GET request)

Requirement Description:

* Login Check:
* The user must be logged in before they can access the orders page.
* If the user is not logged in, they should be redirected to the login page with a message: 'Please log in first.'
* Order Data:
* Upon successful login, the system should query the orders table in the database.
* The order data should be returned as a JSON object, listing all orders with details (e.g., order ID, status, quantity).
* Validation:
* Only users with proper roles (e.g., 'Admin', 'Inventory Manager') should have access to view orders.
* Error Handling:
* If an error occurs while retrieving order data, the system should return: 'Unable to retrieve order data.'
* Outputs:
* Success Output: A JSON array of orders.
* Error Output: 'Please log in first.' or 'Unable to retrieve order data.'

Use Case 4: Sales Reports Requirement number: 4

Inputs:

- None (GET request)

Requirement Description:

- Login Check:

- - The user must be logged in to access the sales reports.

- - If the user is not logged in, they will be redirected to the login page, with the message: 'Please log in first.'

- Sales Data:

- - The system should retrieve the sales reports from the database upon a successful login.

- - The sales report data will be returned as a JSON object, containing the report details (e.g., report ID, generated date, report data).

- Validation:

- - Only users with specific roles, such as 'Admin' or 'Inventory Manager', should be allowed to access this data.

- Error Handling:

- - If there is an issue fetching the sales reports, return: 'Unable to retrieve sales report data.'

Outputs:

- Success Output: A JSON array containing the sales report data.

- Error Output: 'Please log in first.' or 'Unable to retrieve sales report data.'

Use Case 5: Logout

Requirement number: 5

Inputs:

- None (POST request)

Requirement Description:

- Logout:

- - The user can log out by sending a POST request to the '/logout' endpoint.

- Session Handling:

- - When the user logs out, the session should be cleared.

- - The system should return a success message indicating the user has logged out.

- Error Handling:

- - If there is an issue with logging out, an appropriate error message should be returned.

Outputs:

- Success Output: 'Logged out successfully'

- Error Output: 'Error logging out.'

System Requirements for Use Case 6: User Logout

System Requirements for Use Case 6: User Logout

• Requirement number: 6

• Inputs:

• None (POST request)

• Requirement Description:

• Session Management: The system must clear the session upon logout.

• Redirect: The system must redirect the user to the login page.

• Error Handling: If the session cannot be cleared, return an error message: “Error logging out.”

• Outputs:

• Success Output: “Logged out successfully.”

• Error Output: “Error logging out.”

System Requirements for Use Case 7: Password Reset

• Requirement number: 7

• Inputs:

• User email address

• Requirement Description:

• Email Validation: The system must check if the provided email exists in the database.

• Email Notification: If the email is valid, the system must send a password reset link to the user’s email address.

• New Password: The system must allow the user to enter a new password after clicking the reset link.

• Security: The reset link must be time-sensitive and expire after a set time period (e.g., 24 hours).

• Error Handling: If the email is not found or an error occurs, return an appropriate error message: *“Email address not registered.”* or *“Error resetting password.”*

• Outputs:

• Success Output: *“Password reset link sent to your email address.”*

• Error Output: *“Email address not registered.”* or *“Error resetting password.”*

System Requirements for Use Case 8: User Permissions Management

• Requirement number: 8

• Inputs:

• User ID

• New permissions settings

• Requirement Description:

• Permissions Update: The system must update the user’s permissions in the database.

• Role Validation: The system must verify that the logged-in user has admin privileges.

• Error Handling: If the user cannot be found or if there is an issue updating the permissions, return: *“Error updating permissions.”*

• Outputs:

• Success Output: *“Permissions updated successfully.”*

• Error Output: *“User not found.”* or *“Error updating permissions.”*

System Requirements for Use Case 9: Role Assignment

• Requirement number: 9

• Inputs:

• User ID

• Role (Admin, Inventory Manager, etc.)

• Requirement Description:

• Role Assignment: The system must update the user’s role in the database.

• Role Validation: Only an admin can assign roles.

• Error Handling: If an invalid user is selected or the role cannot be assigned, return an error message: *“User not found.”* or *“Error assigning role.”*

• Outputs:

• Success Output: *“Role assigned successfully.”*

• Error Output: *“User not found.”* or *“Error assigning role.”*

System Requirements for Use Case 10: Fabric Order Placement

• Requirement number: 10

• Inputs:

• Fabric item ID

• Quantity

• Requirement Description:

• Inventory Update: The system must update the fabric inventory based on the quantity ordered.

• Order Creation: A new order record must be created in the orders table.

• Error Handling: If an error occurs during order placement, return: *“Error placing order.”*

• Outputs:

• Success Output: *“Order placed successfully.”*

• Error Output: *“This fabric is currently out of stock.”* or *“Error placing order.”*

Use Case 5: Account Creation

• Requirement number: 5

• Inputs:

• Username

• Password

• Confirm Password

• Email

• Requirement Description:

• Username:

• The username must be unique and not already exist in the users table.

• If the username is already taken, return the message: *“Username already exists.”*

• Email:

• The email must be unique.

• If the email is already in the database, return the message: *“An account already exists with this email address. Please log in.”*

• Password:

• The password must be at least 8 characters long.

• The password must contain at least one number, one letter, and one special character (e.g., !@#$%^&\*).

• The password must not be the same as the username.

• The password must not contain the username as part of the string.

• Confirm Password:

• The password and confirm password fields must match.

• If they don’t, return the message: *“Passwords do not match.”*

• Validation:

• The system must validate all inputs and ensure they are not empty.

• The system should verify the password meets the specified complexity criteria.

• The system should check if the username or email already exists in the database.

• Account Creation:

• After successful validation, the system should hash the password and store the username, password hash, and email in the database.

• The system should assign the default role as “Staff Member” during account creation.

• After successful account creation, redirect the user to the login page with the message: *“Account created successfully!”*

• Error Handling:

• If there is any issue with the account creation (e.g., username or email already exists), the system should display an appropriate error message.

• Outputs:

• Success Output: *“Account created successfully!”*

• Error Output: Specific error messages such as *“Username already exists.”*, *“An account already exists with this email address. Please log in.”*, *“Password must be at least 8 characters long.”*, etc

|  |  |
| --- | --- |
| Requirement number: 3 | Use Case Number: 3 |
| Introduction: Users should be able to update fabric stock when receiving shipments. | |
| Inputs: Fabric ID, updated quantity. | |
| Requirements Description: The system shall allow inventory managers to modify stock levels and maintain logs of changes. | |
| Outputs: Updated stock levels and confirmation message | |

|  |  |
| --- | --- |
| Requirement number: 4 | Use Case Number: 4 |
| Introduction: The system should allow authorized users to remove fabric records if no dependencies exist. | |
| Inputs: Fabric ID | |
| Requirements Description: If the fabric is linked to pending orders, deletion should be restricted with an error message. Otherwise, the record should be removed from the database. | |
| Outputs: Confirmation of deletion or restriction warning | |

|  |  |
| --- | --- |
| Requirement number: 5 | Use Case Number: 5 |
| Introduction: Users should be able to quickly search and filter fabric inventory. | |
| Inputs: Search criteria (type, color, supplier, stock level) | |
| Requirements Description: The system shall retrieve matching fabric entries and display them. If no matches exist, it shall prompt users to refine search criteria. | |
| Outputs: List of matching fabrics or "no matches found" message | |

|  |  |
| --- | --- |
| Requirement number: 6 | Use Case Number: 6 |
| Introduction: The system should generate alerts when stock levels fall below a predefined threshold. | |
| Inputs: Fabric stock levels | |
| Requirements Description: The system shall automatically notify users when restocking is required. | |
| Outputs: Restock alert notification | |

|  |  |
| --- | --- |
| Requirement number: 7 | Use Case Number: 7 |
| Introduction: The system should facilitate placing restock orders with suppliers. | |
| Inputs: Fabric ID, quantity, supplier details | |
| Requirements Description: The system shall generate and send order requests to suppliers. If a supplier is unavailable, alternative suppliers should be suggested. | |
| Outputs: Order confirmation message | |

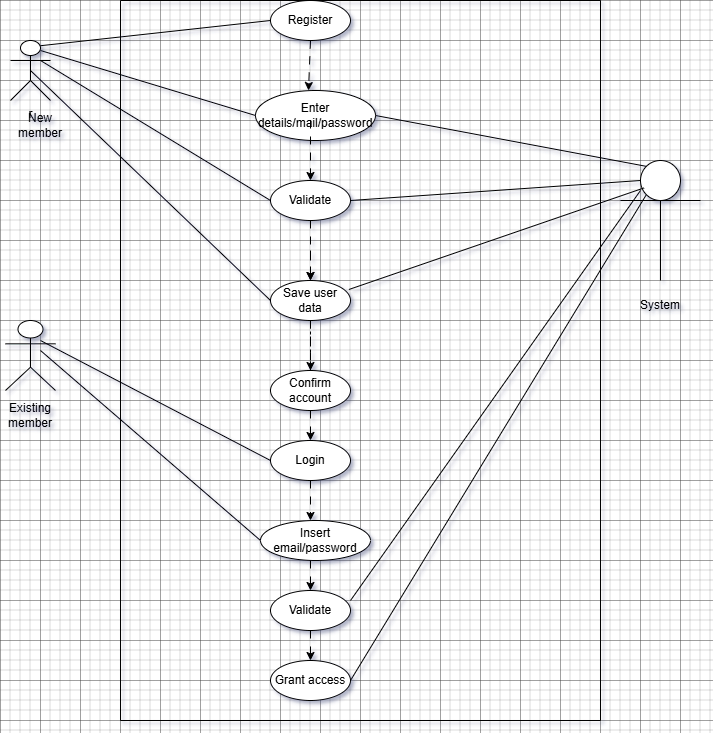
|  |  |
| --- | --- |
| Requirement number: 8 | Use Case Number: 8 |
| Introduction: The system should generate analytical reports for inventory and sales. | |
| Inputs: Report type, date range | |
| Requirements Description: The system shall compile sales and inventory data into structured reports. | |
| Outputs: Graphs, tables, or downloadable reports | |

|  |  |
| --- | --- |
| Requirement number: 9 | Use Case Number: 9 |
| Introduction: User access should be role-based, limiting actions based on predefined permissions. | |
| Inputs: User role, authentication credentials | |
| Requirements Description: The system shall grant or restrict access to functions based on user roles. Unauthorized actions shall trigger a warning message. | |
| Outputs: Access granted or restricted message | |

|  |  |
| --- | --- |
| Requirement number: 10 | Use Case Number: 10 |
| Introduction: The chatbot should assist users with inventory-related queries. | |
| Inputs: User query | |
| Requirements Description: The chatbot shall process input and provide relevant inventory data or suggest escalation to human support. | |
| Outputs: Chatbot response or escalation suggestion | |

|  |  |
| --- | --- |
| Requirement number: 11 | Use Case Number: 11 |
| Introduction: The system should evaluate supplier performance based on historical transactions. | |
| Inputs: Supplier transaction history, delivery speed, order accuracy | |
| Requirements Description: The system shall generate reports on supplier reliability and performance. | |
| Outputs: Supplier rating and performance analytics | |

**Use Case Diagrams**

****

**Section 4: System Modeling**

**Architecture Modeling**

Our system will use a combination of Layered Architecture, Repository Architecture, and Client-Server Architecture. Here’s why:

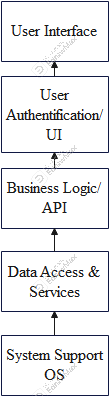
Layered Architecture provides clear separation of concerns, ensuring that changes in one layer do not cascade into other layers. This simplifies maintenance and allows for independent updates or replacements without affecting other layers.

Repository Architecture centrally manages data integrity and sharing across multiple functional modules, which is crucial for a system managing large volumes of inventory data.

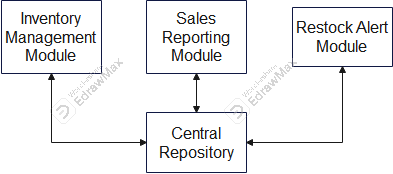
Client–Server Architecture ensures that the system is accessible from distributed locations and can handle concurrent user interactions, a key requirement for web-based applications. Organizing the system with clients interacting with a central server allows multiple users to access inventory data concurrently. This setup supports scalability and remote accessibility, crucial for users in different locations.

Together, these patterns offer a robust, maintainable, and scalable design that aligns well with both the functional and non-functional requirements of the Fabric Inventory Management System. Below are the diagrams that represent each of the three architectural patterns our system will use.

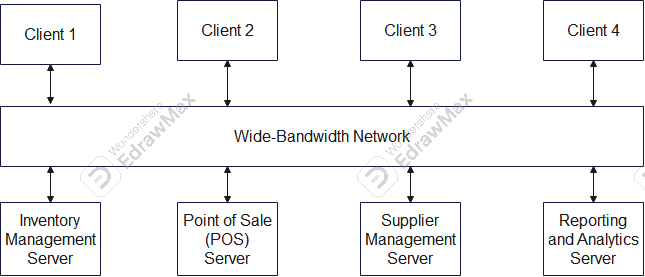
Layered Model



Repository Model

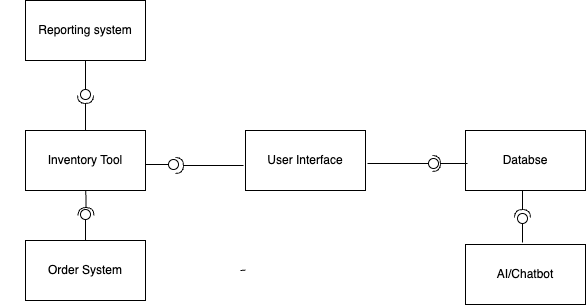


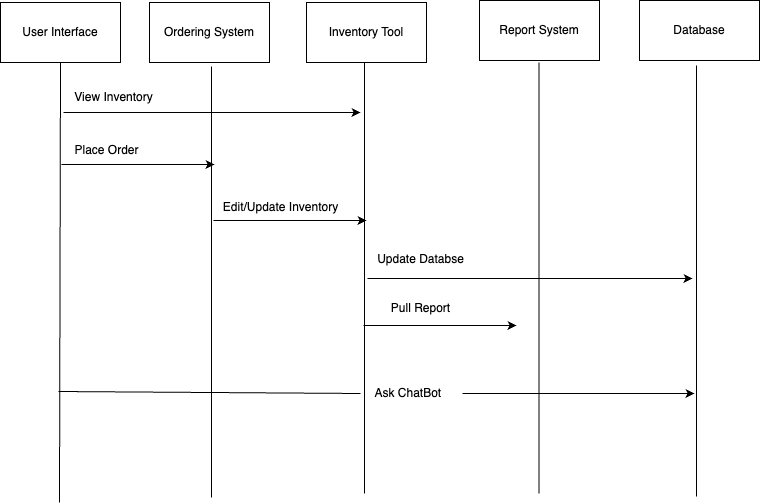
Client-Server Model



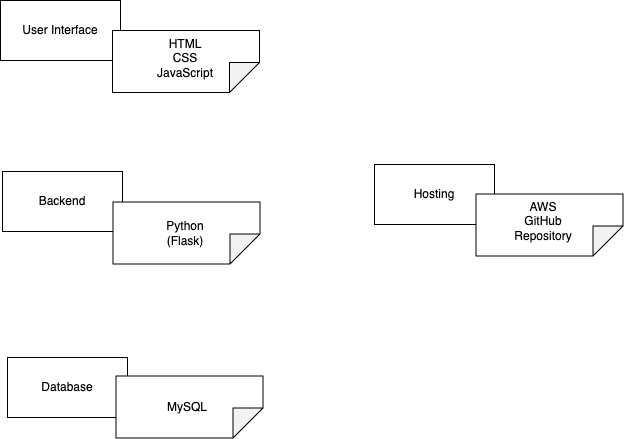
4 + 1 Architectural Design (Logical View, Process View, Developmental View, Physical View)

Logical View

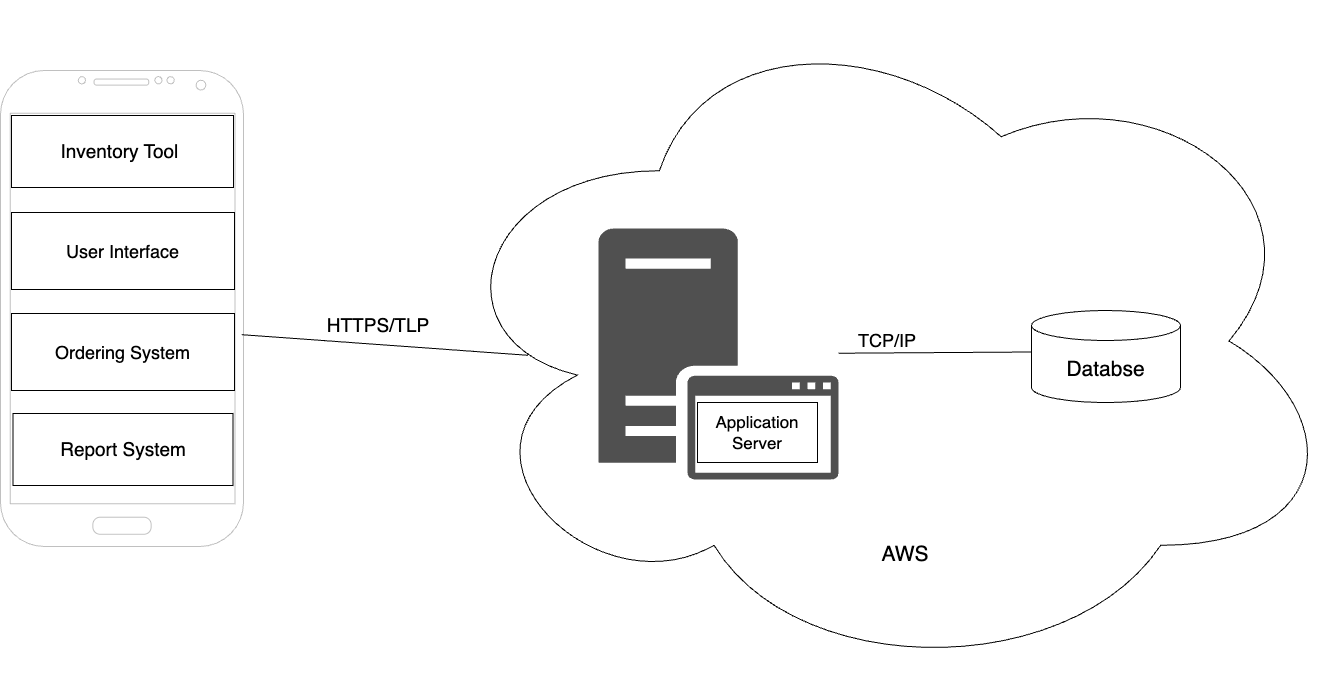


Process View   


Developmental View

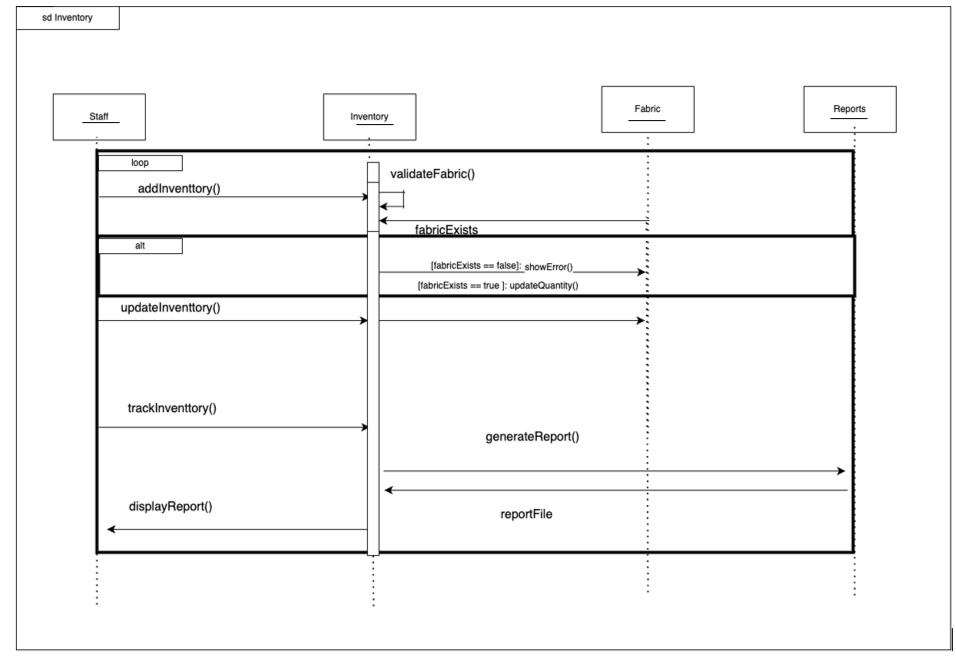


Physical View

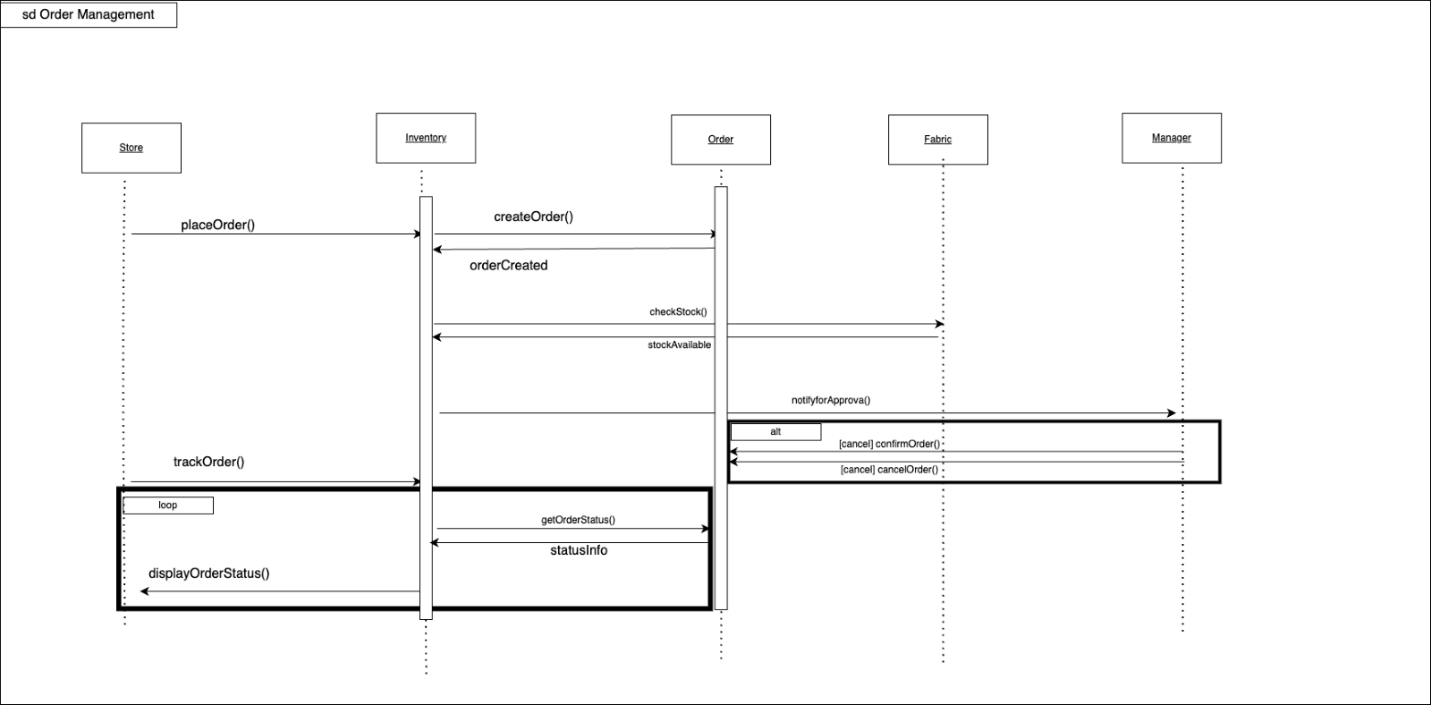


**Behavioral Modeling**

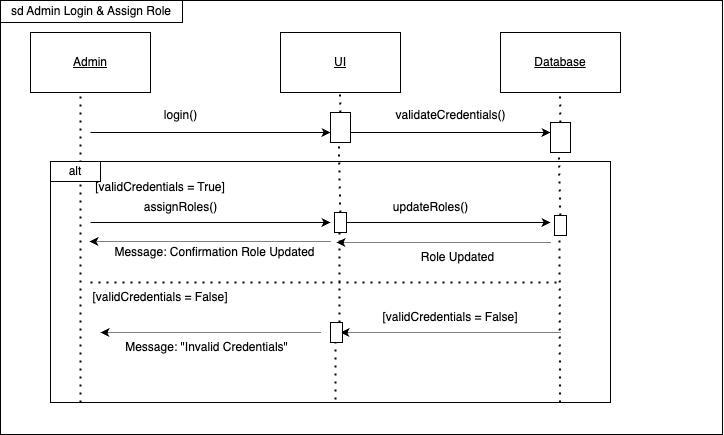
Sequence Diagram 1



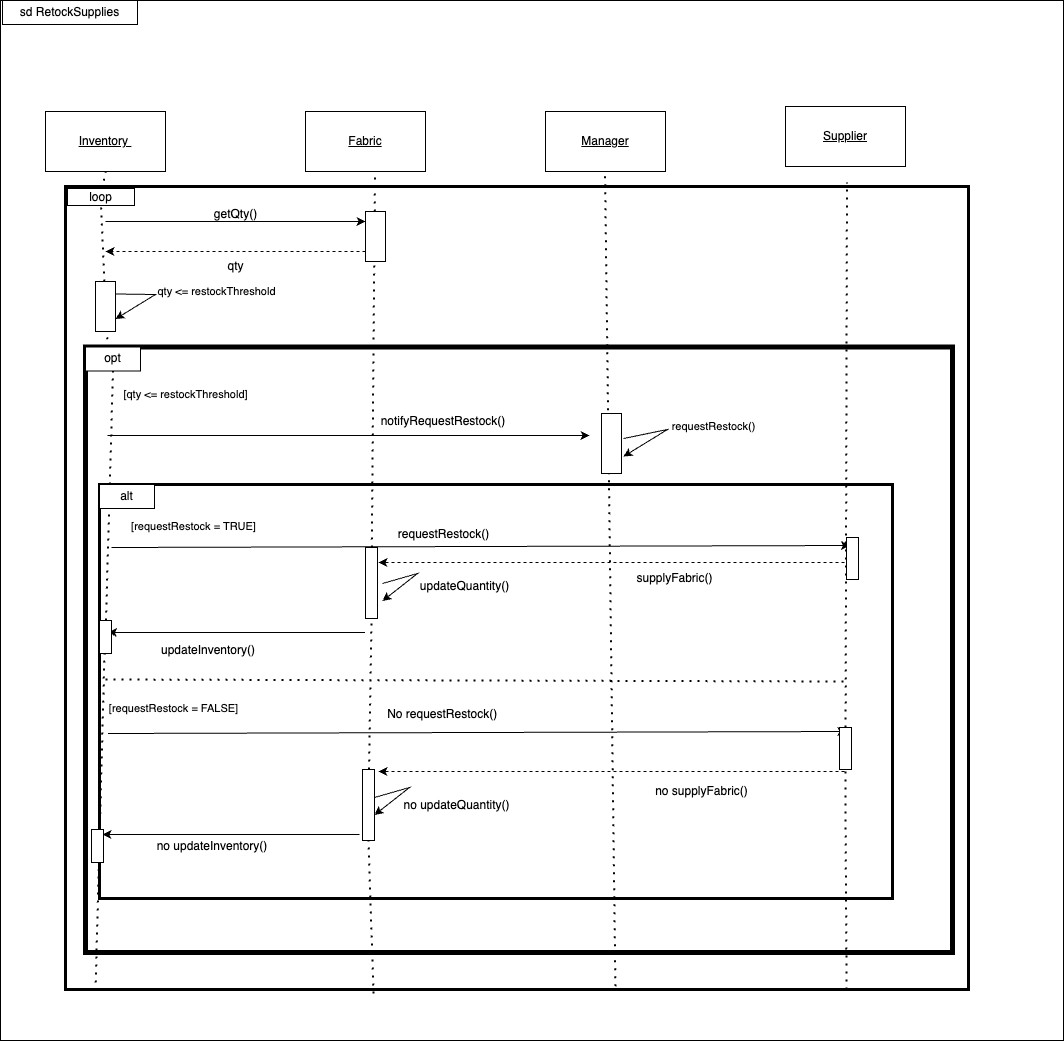
Sequence Diagram 2



Sequence Diagram 3



Sequence Diagram 4



**Database Specification and Analysis**

System Architecture:

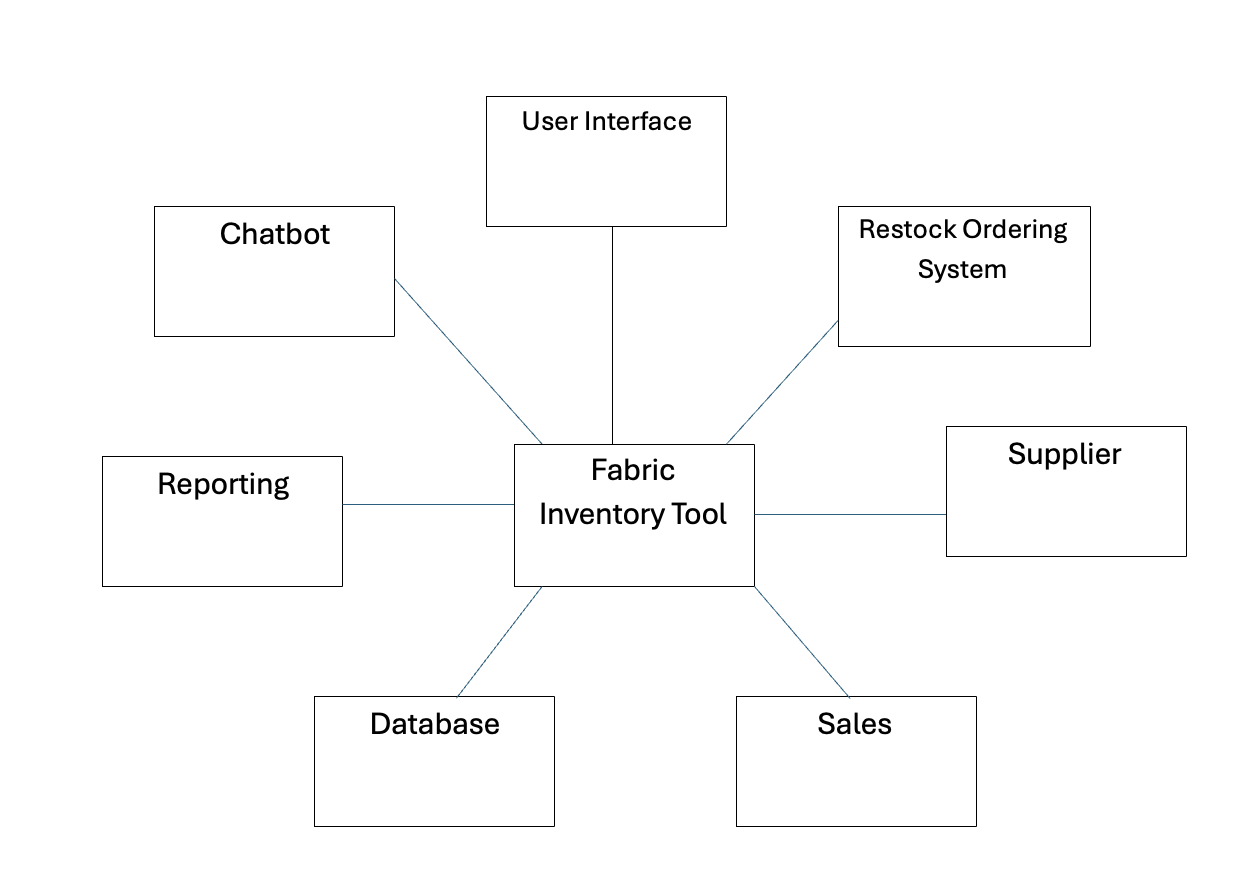
The context model below, Figure 3, visualizes at a high level the system architecture of the Fabric Inventory Tool. This shows the interaction of the tool with other components. Each component has a unique job/purpose, however, there is overlap as together all of them make the efficient and complete Fabric Inventory Tool.

Below are the various components/systems:

1. Fabric Inventory Tool:
2. This is the center piece in the context model as it is the primary system. It tracks incoming and outgoing inventory, manages stock levels, and generates restock alerts. The database and sales tracking system connect/interact with this.
3. User Interface:
4. This component helps staff members of the fabric supply interact with the system and perform tasks such as tracking and updating inventory. The basic inventory update features include adding, editing, and deleting items. This component interacts with all other system components.
5. Database:
6. This component stores the data in an organized manner. Such as data regarding the type of fabric, the quantity, various suppliers' details, the user data, etc. It interacts with all other components.

1. Restock Ordering System:
2. When inventory levels reach a set threshold, this system alerts the user about which fabric needs restocking and identifies the supplier that provided it.
3. Supplier
4. This is the third-party supplier that produces the various fabrics and sells it to the user. This is whom we look towards to purchase supplies, thus making it important that the inventory interacts with it.
5. Sales System
6. This system keeps track of the financial aspect of the user business. How many stocks are being purchased and at what price. As well as the rate at which the stock the user ordered and the need to restock. The profit or loss the user is making.
7. Reporting System
8. This system interacts with the inventory making reports that display the stocks, supplying, ordering and purchase transactions.
9. Chatbot
10. Helps the user gain information about the inventory quickly and accurately through the help of AI.

Context Model



**Database Tables**

#Note- PK stands for Primary key & FK stands for Foreign Key.

1. Users Table

• user\_id (INT, PK, AUTO\_INCREMENT)

• username (VARCHAR(50), UNIQUE, NOT NULL)

• password\_hash (VARCHAR(255), NOT NULL)

• role (ENUM('Admin', 'Inventory Manager', 'Staff Member', 'Business Owner'),

NOT NULL)

2. Fabric Inventory Table

• fabric\_id (INT, PK, AUTO\_INCREMENT)

• fabric\_type (VARCHAR(100), NOT NULL)

• color (VARCHAR(50), NOT NULL)

• quantity (INT, NOT NULL)

• price\_per\_unit (DECIMAL(10,2), NOT NULL)

• supplier\_id (INT, FK → Suppliers.supplier\_id)

• restock\_threshold (INT, NOT NULL)

3. Suppliers Table

• supplier\_id (INT, PK, AUTO\_INCREMENT)

30

• supplier\_name (VARCHAR(100), NOT NULL)

• contact\_info (VARCHAR(255), NOT NULL)

• address (TEXT, NOT NULL)

• rating (DECIMAL(3,2), DEFAULT NULL)

4. Stock Transactions Table

• transaction\_id (INT, PK, AUTO\_INCREMENT)

• fabric\_id (INT, FK → Fabric Inventory.fabric\_id)

• transaction\_type (ENUM('Addition', 'Removal', 'Update'), NOT NULL)

• quantity\_changed (INT, NOT NULL)

• user\_id (INT, FK → Users.user\_id)

• transaction\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

5. Restock Alerts Table

• alert\_id (INT, PK, AUTO\_INCREMENT)

• fabric\_id (INT, FK → Fabric Inventory.fabric\_id)

• current\_stock (INT, NOT NULL)

• alert\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

• status (ENUM('Pending', 'Resolved'), DEFAULT 'Pending')

6. Orders Table

• order\_id (INT, PK, AUTO\_INCREMENT)

• supplier\_id (INT, FK → Suppliers.supplier\_id)

• fabric\_id (INT, FK → Fabric Inventory.fabric\_id)

• quantity (INT, NOT NULL)

• order\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

• status (ENUM('Pending', 'Completed', 'Cancelled'), DEFAULT 'Pending')

7. Sales Reports Table

• report\_id (INT, PK, AUTO\_INCREMENT)

• generated\_by (INT, FK → Users.user\_id)

• report\_date (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP)

• report\_data (TEXT, NOT NULL) -- (Can store JSON or structured data)

8. Role-Based Access Table

• role\_id (INT, PK, AUTO\_INCREMENT)

• role\_name (VARCHAR(50), UNIQUE, NOT NULL)

• permissions (TEXT, NOT NULL) -- (Stores role-based access rules)

Relationships:

1. Users manage Fabric Inventory and process transactions.

2. Fabric Inventory belongs to Suppliers.

3. Stock Transactions track fabric changes by Users.

4. Restock Alerts are generated based on Fabric Inventory stock levels.

31

5. Orders are placed with Suppliers for Fabric Inventory restocking.

6. Sales Reports are generated by Users.

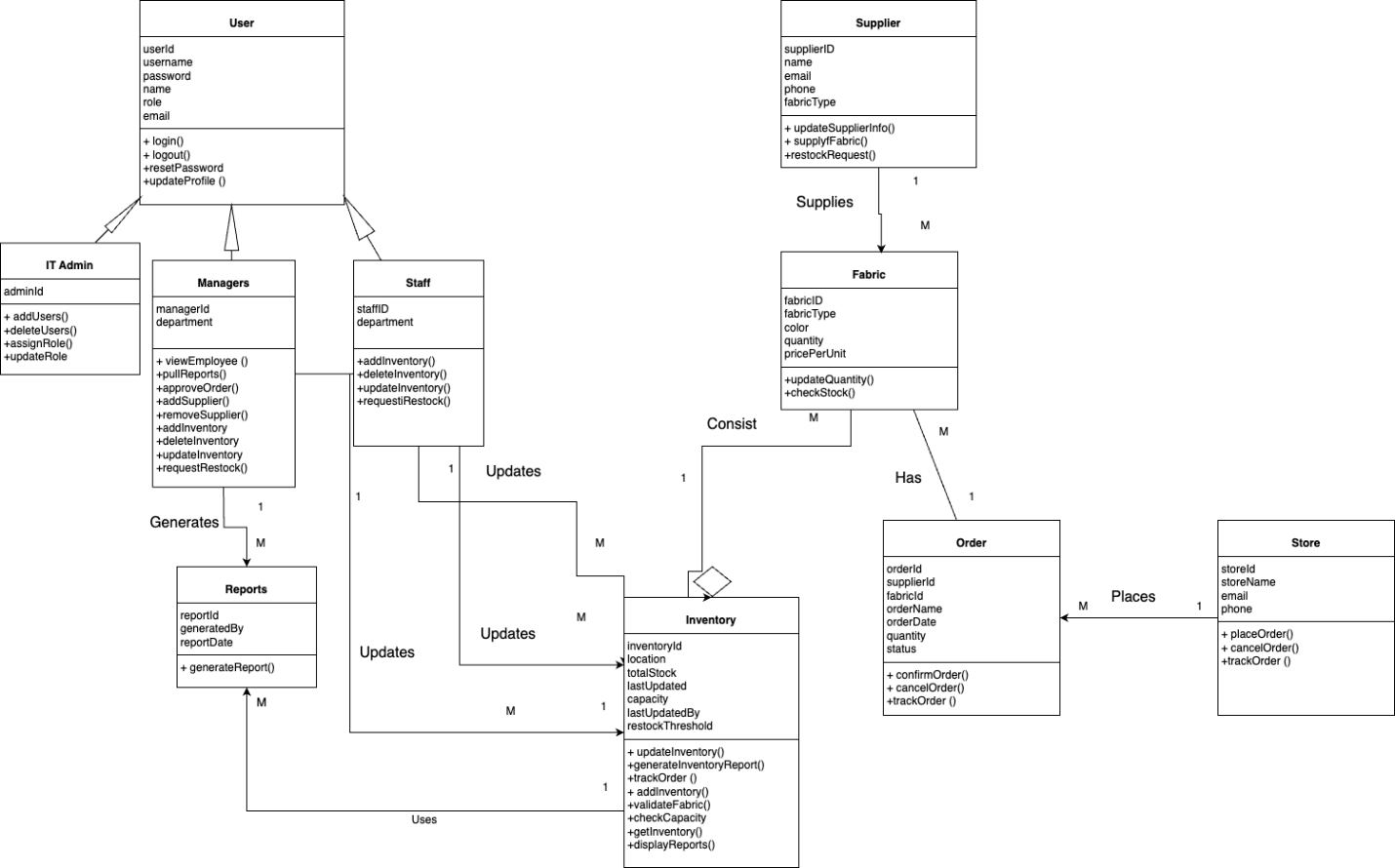
7. Role-Based Access defines access for Users.

Database Management System,

We will be using MySQL as our DBMS and will be using MySQL Workbench to host the

system.

**Updated Class Diagrams**



**Section 5: Implementation**

Overview:

The goal of this task was to implement a user authentication system, allowing users to log in, manage their accounts, and access certain pages (like the inventory, orders, and reports) based on their login status. This was achieved by integrating both frontend (HTML, CSS, and JavaScript) and backend (Flask, MySQL on AWS RDS) technologies. The core functionality included:

• User Registration (Create Account)

• Login with Validation

• Role-based Access Control (with login\_required decorator)

Implementation Details:

Frontend (HTML, CSS, JavaScript):

1. Login Page (login.html):

• The login page allows users to input their username and password.

• A POST request is sent to the Flask backend to validate the credentials.

• The frontend uses the Fetch API to send the credentials as JSON to the backend and handle the response.

• If the login is successful, the user is redirected to the dashboard page (dashboard.html). If not, an error message is displayed.

2. Create Account Page (CreateAccount.html):

• This page allows users to create a new account by entering a username, password, and email.

• Password validation is implemented on the frontend, including checks for length, digits, letters, and special characters.

• A POST request is sent to the backend to store the user’s information in the database.

3. Role-based Access:

• Links in the navigation bar are conditionally rendered based on whether the user is logged in and their role (e.g., “Admin”, “Inventory Manager”).

• Unauthorized users are redirected to the login page if they attempt to access certain pages.

Backend (Flask, AWS RDS MySQL, python):

1. User Registration:

• The backend handles user registration through the /CreateAccount route.

• Before adding a user to the database, checks are made to ensure that the username and email are unique.

• Passwords are hashed using werkzeug.security to ensure security.

2. Login:

• The /login route accepts the username and password as POST requests.

• The backend validates the credentials by comparing the entered password (after hashing) with the stored password in the database.

• If the credentials are valid, a session is created, and the user is granted access to the dashboard.

• The session is stored using Flask’s session management system, with the logged\_in key indicating the user’s authentication status.

3. Inventory Management:

• The /inventory route fetches and displays all fabric inventory items after validating that the user is logged in.

• This page is only accessible to logged-in users, enforced by the login\_required decorator.

5. Database (AWS RDS MySQL):

• AWS RDS is used to host the MySQL database, ensuring scalability, reliability, and remote access.

• The users’ table stores information like username, password hash, role, and email.

• The fabric inventory table stores the data related to fabrics, including type, color, quantity, and price per unit.

• Connection to AWS RDS: The database connection is made via get\_connection() that connects to the AWS RDS instance securely using the environment variables stored in .env.

4. Testing and Validation:

• Test Case 1: Login with Valid Credentials:

• Tested logging in with valid credentials (username and password). The user was successfully redirected to the dashboard after authentication.

• Test Case 2: Create Account:

• Tested account creation with valid inputs and ensured that error messages were displayed for invalid or missing fields (e.g., missing email, password too short).

• Test Case 3: Inventory Management Access:

• Tested that the inventory page could only be accessed by logged-in users. Users who were not logged in were redirected to the login page.

5. Compilation Instructions: - Clone the github repository https://github.com/Amisu-Tayo/SWE-CodeCrafters

- Run the run.py strict located in the SWE-CodeCrafters folder (use python run.py in your command interface.

- This automatically installs all dependencies from requirements.txt, sets up a virtual environment, starts the flask development server and opens the app in a web browser.

**Section 6: Testing**

**Test Specifications:**

**USE CASE 1: LOGIN**   
   
**Description**: This use case lets users sign in with their username and password. If the details are correct, they can access their account. If not, they’ll see an error and be asked to try again.

**Feature Identification:**

* **Feature Name**: User Login
* **Feature Type**: Authentication
* **Related Modules**: User Management, Security
* **Purpose**: To authenticate users and provide secure access to the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Case Name** | **Input** | **Expected Result** |
| **1.1** | Login with valid credentials | login (FakeUser2, Abcdefghijk1@) | Login Successful |
| **1.2** | Login with Invalid Username | login(Tayo1,Tes@tPassword1) | Invalid Credentials |
| **1.3** | Login with incorrect password | login(FakeUser2,Wr0ng@Password) | Invalid Credentials |
| **1.4** | Login with empty username and password | login(‘ ’, ‘ ’) | Sytem will display the following message: Username and password required. |

**Test Case 1.1: Login with Valid Credentials:** Test Case ID: login(username, password)

Test Value:  login (FakeUser2, Abcdefghijk1@) (This user was created in an earlier test of CreateAccount)

**Pre-condition:**

* The user is already registered with a valid username and password in the system.
* The user is on the login page (login.html).

**Test Steps:**

1. Open the login page (login.html).
2. Enter a valid username and the corresponding password.
3. Click on the login button.

**Expected Outcome :**

* The system should validate the credentials.
* If the credentials are correct, the user is redirected to the dashboard page (dashboard.html).
* A success message (“Login successful”) should be shown.

**Actual Outcome:**

* Login Successful is outputted
* User is redirected to inventory page
* After successful redirection, user is redirected once more to login page with message “Invalid Credentials”

**Pass Criteria:**

• User successfully logs in and is redirected to the dashboard page without any errors.

**Test Outcome: Failed- Debugging necessary**

**Test Case 1.2: Login with Invalid Username:** Test Case ID: login(username, password)

Test Value: login(Tayo1, Tes@tPassword1)

**Pre-condition / Partition:**

* The entered username does not exist in the system (e.g., InvalidUser).
* The user is on the login page (login.html).

**Test Steps:**

1. Open the login page (login.html).
2. Enter an valid username and a password.
3. Click on the login button.

**Expected Outcome :**

* The system should return an error message: “Invalid credentials.”
* The user should stay on the login page.

**Actual Outcome:**

* The system returned error message: “Invalid credentials.”
* The user stayed on the login page.

**Pass Criteria:**

• The system shows an error message and prevents login.

**Test Outcome: Passed**

**Test Case 1.3: Login with Incorrect Password:** Test Case ID: login(username, password)

Test Values: login(FakeUser2, Wr0ng@Password)

**Pre-condition:**

* The user is already registered with the correct username and password in the system
* The user is on the login page (login.html).
* User enters correct name but incorrect password

**Test Steps:**

1. Open the login page (login.html).
2. Enter a valid username (FakeUser2)  and an invalid  password.
3. Click on the login button.

**Expected Outcome:**

* The system should return an error message: “Invalid credentials.”
* The user should remain on the login page

**Actual Outcome:**

* The system returned an error message: “Invalid credentials.”
* The user stayed on the login page.

**Pass Criteria:**

• The system shows an error message and prevents login.

**Test Outcome: Passed**

**Test Case 1.4: Login with Empty Fields** Test Case ID: login(username, password)

Test Values: login(‘’, ‘’)

**Pre-condition:**

* The user is on the login page (login.html).
* Username and password fields are left empty

**Test Steps:**

1. Open the login page (login.html).
2. Leave the username and password fields empty.
3. Click on the login button.

**Expected Outcome:**

* The system should display a validation message: “Both username and password are required.”
* The user should not be able to proceed with the login

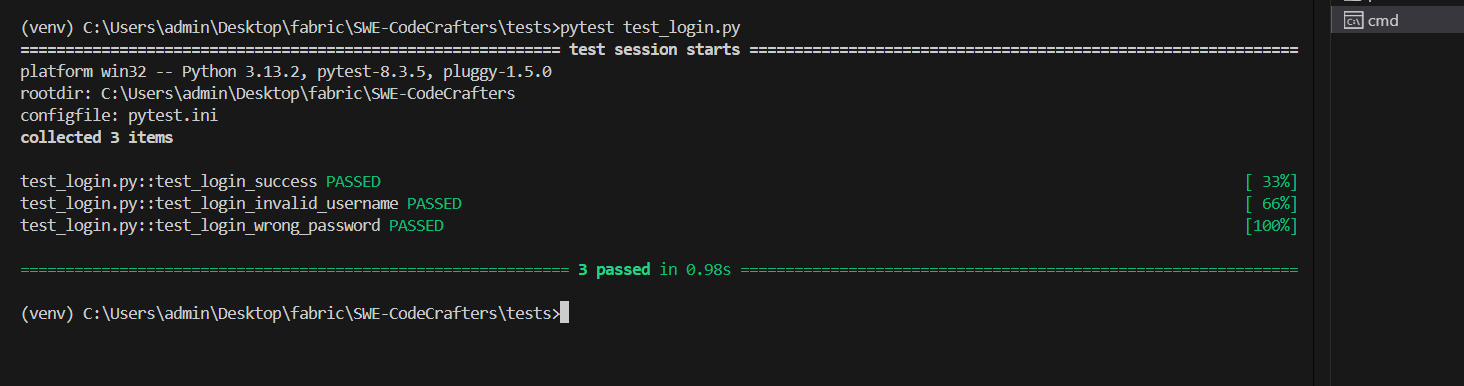
**Actual Outcome:**

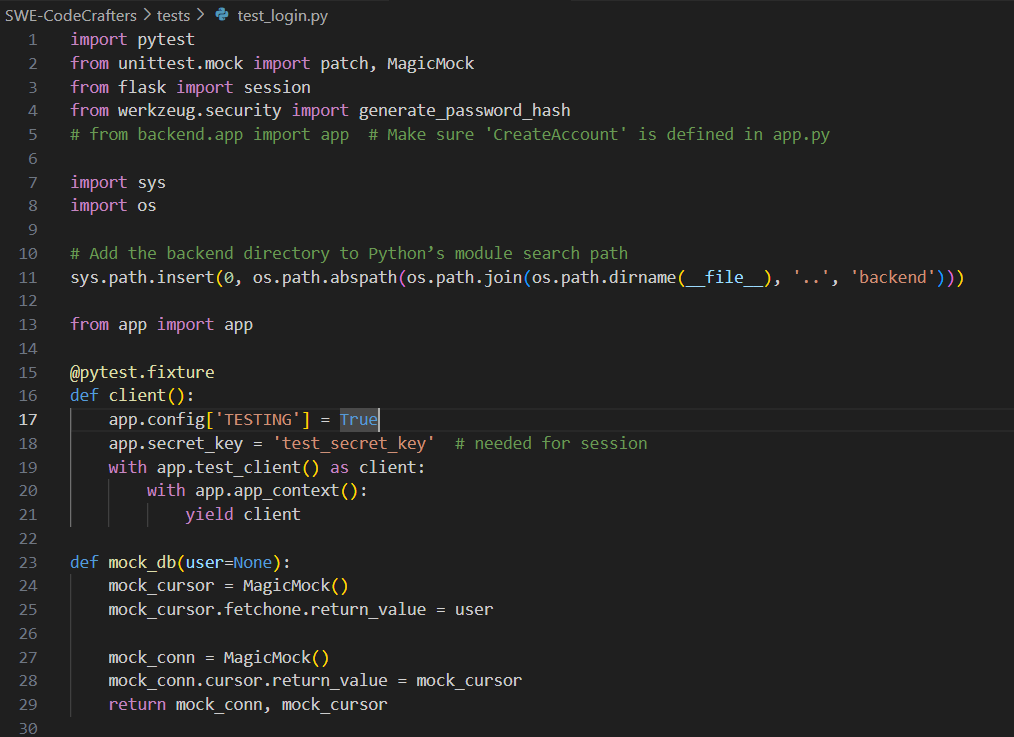
* The system displayed a validation message: “Both username and password are required.”
* The user stayed on the login page.

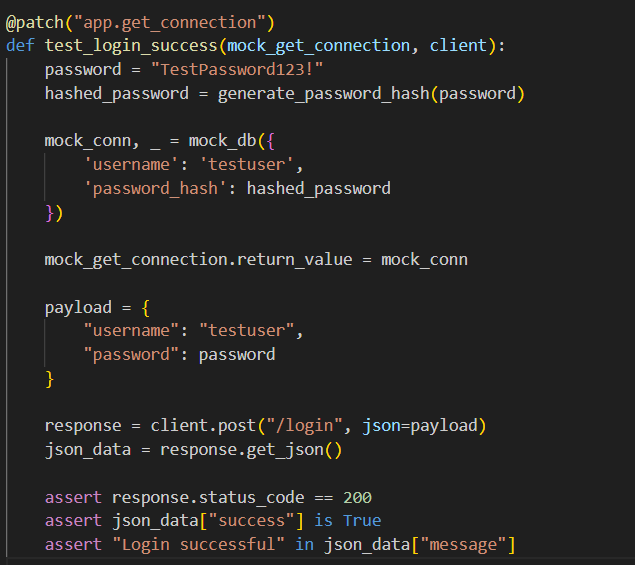
**Pass Criteria:**

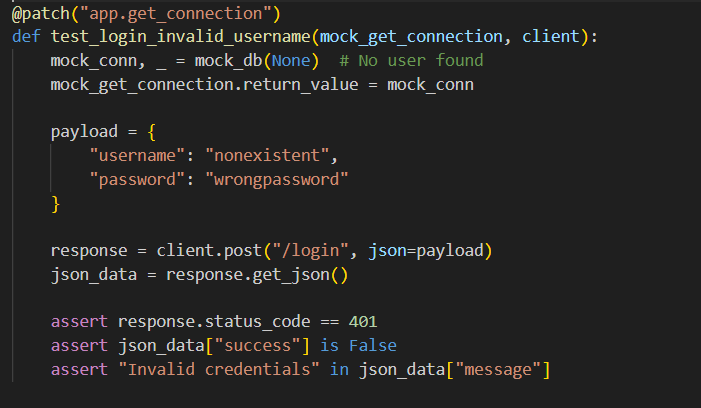
• The system shows a  validation message and prevents login.

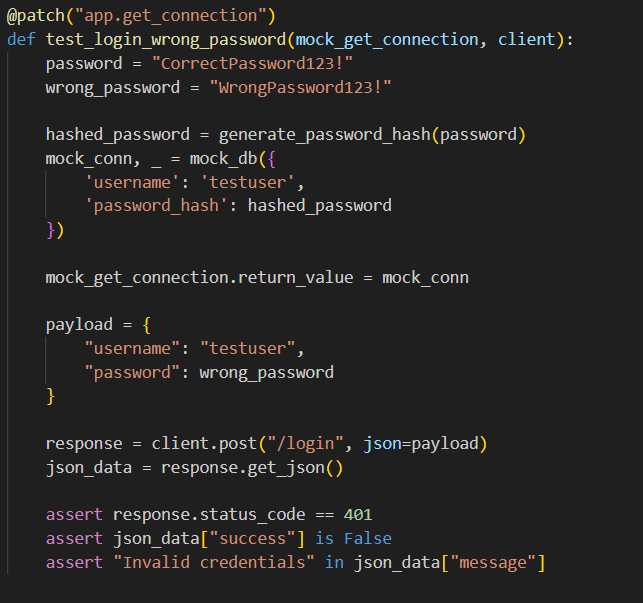
**Test Outcome: Passed**

Shape 







**TEST CASE 2: ACCOUNT CREATION**  
**Description:-** This use case helps new users get started by signing up with a unique username, email, and a confirmed password. Once everything checks out, their account is created and they’re ready to log in.

**Feature Identification:**

* **Feature Name**: User Registration
* **Feature Type**: User Management
* **Related Modules**: Authentication, User Database, Input Validation
* **Purpose**: To allow new users to create an account and gain access to the system

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Case Name** | **Input** | **Expected Result** |
| **2.1** | Create account with valid input | login(FakeUser2, Abcdefghij1@, Abcdefghij1@, user2@user2.com) | The account should be created successfully |
| **2.2** | Create an account with an existing Username. | login(FakeUser2, Abcdefghij1@, Abcdefghij1@, userTest@user2.com) | The system should display an error message: “Username already exists.” |
| **2.3** | Create Account with Existing Email | login(FakeUser3, Abcdefghij1@, Abcdefghij1@, user2@user2.com) | The system should display an error message: “An account already exists with this email address. Please log in.” |
| **2.4** | Create Account with Existing Email | login(FakeUser3, Abcdefghij1@, Abcdefghij1@, user2user2.com) | The system should display an error message: “Invalid email format.” |

**Test Case 2.1 Create Account with Valid Data:** Test Case ID: CreateAccount(username, password, ConfirmPassword, email)

Test Values: login(FakeUser2, Abcdefghij1@, Abcdefghij1@, user2@user2.com)

**Pre-condition:**

* The user is not previously registered in the system
* The user is on the create account page (CreateAccount.html).
* The user enters valid details (username, password, confirm password, email)

**Test Steps:**

1. Open the create account page (CreateAccount.html)..
2. Enter a valid username (e.g., NewUser), a password (e.g., TestPassword1!), confirm the password, and provide a valid email (e.g., newuser@test.com).
3. Click on the “Create Account” button

**Expected Outcome:**

* The account should be created successfully
* The system should redirect the user to the login page (login.html) after successful account creation.

**Actual Outcome:**

* The account was created successfully
* The system redirected the user to the login page (login.html) after successful account creation.

**Pass Criteria:**

• The system successfully creates an account and is redirected to the loginn page

**Test Outcome: Passed**

Shape

**Test Case 2.2: Create Account with Existing Username:** Test Case ID: CreateAccount(username, password, ConfirmPassword, email)

Test Values: login(FakeUser2, Abcdefghij1@, Abcdefghij1@, userTest@user2.com)

**Pre-condition:**

* The user is already registered in the system
* The user is on the create account  page (CreateAccount.html).

**Test Steps:**

1. Open the create account page (CreateAccount.html)..
2. Enter an already existing username (e.g.FakeUser2, ), a password (e.g., TestPassword1!), confirm the password, and provide a valid email (e.g., newuser@test.com).
3. Click on the “Create Account” button

**Expected Outcome:**

* The system should display an error message: “Username already exists.”
* The user should not be able to create an account with the existing username.

**Actual Outcome:**

* The system displayed an error message: “Username already exists.”

**Pass Criteria:**

• The system prevents account creation with an existing username and shows an appropriate error message

**Test Outcome: Passed**

Shape

**Test Case 2.3: Create Account with Existing Email:** Test Case ID: CreateAccount(username, password, ConfirmPassword, email)

Test Values: login(FakeUser3, Abcdefghij1@, Abcdefghij1@, user2@user2.com)

**Pre-condition:**

* The email entered already exists in the database
* The user is on the create account  page (CreateAccount.html).

**Test Steps:**

1. Open the create account page (CreateAccount.html)..
2. Enter a valid username (e.g., NewUser), password, confirm password, and an email that already exists in the system (e.g., user1@example.com).
3. Click on the “Create Account” button

**Expected Outcome:**

* The system should display an error message: “An account already exists with this email address. Please log in.”
* The user should not be able to create an account with the existing email.
* User should be redirected to login

**Actual Outcome:**

* The system displayed an error message: “An account already exists with this email”

**Pass Criteria:**

* The system prevents account creation with an existing email and shows an appropriate error message

**Test Outcome: Passed**

Shape**Test Case 2.4: Create Account with Existing Email:** Test Case ID: CreateAccount(username, password, ConfirmPassword, email)

Test Values: login(FakeUser3, Abcdefghij1@, Abcdefghij1@, user2user2.com)

**Pre-condition:**

* The user enters an invalid email format (e.g., invalidemail.com).
* The user is on the create account  page (CreateAccount.html).

**Test Steps:**

1. Open the create account page (CreateAccount.html)..
2. Enter a valid username (e.g., NewUser), password, confirm password, and an invalid email (e.g., invalidemail.com).
3. Click on the “Create Account” button

**Expected Outcome:**

* The system should display an error message: “Invalid email format.”
* The user should not be able to create an account with an invalid email.

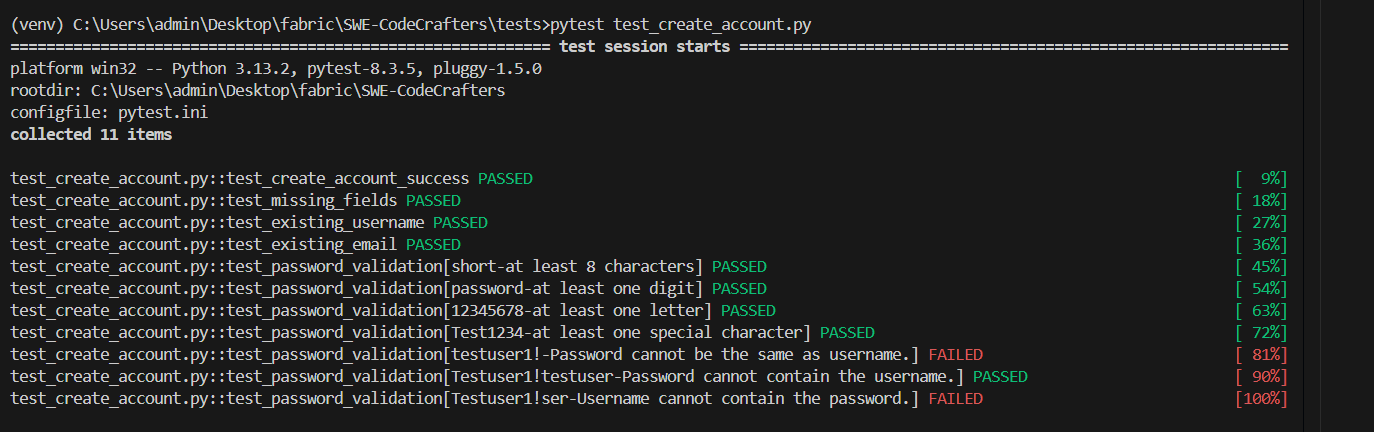
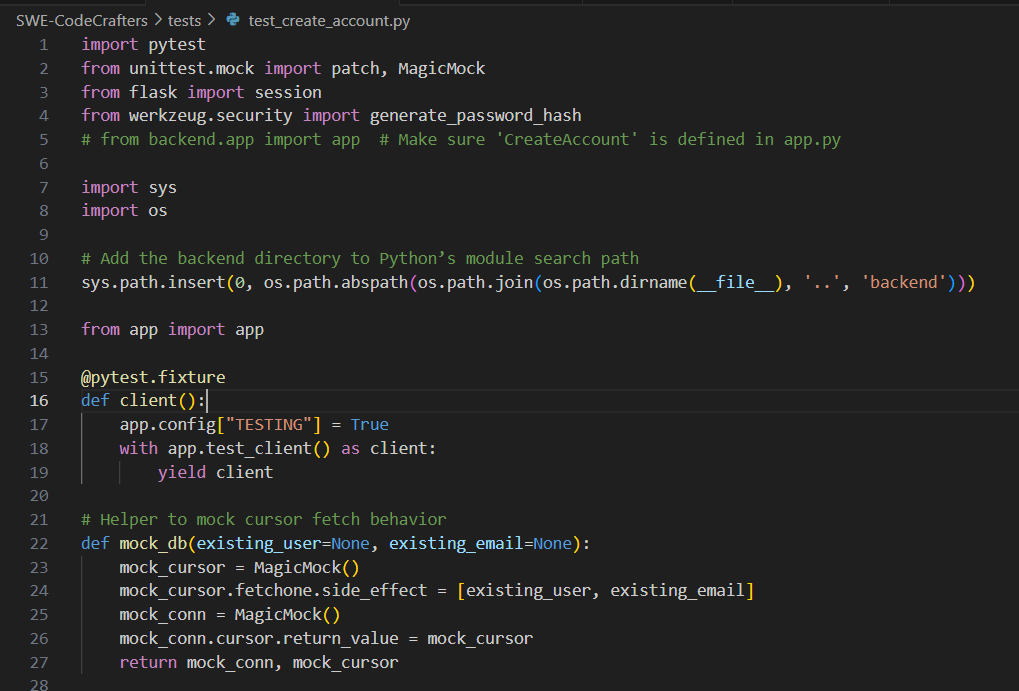
**Actual Outcome:**

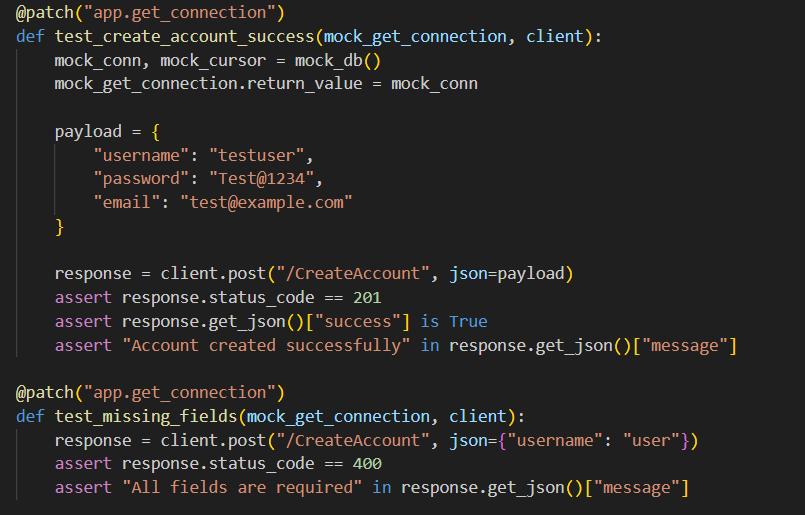
* The system displayed an error message: “Invalid email format.”

**Pass Criteria:**

* The system prevents account creation with an invalid email format and shows an appropriate error message.

**Test Outcome: Passed**







Part 2

**Create account**

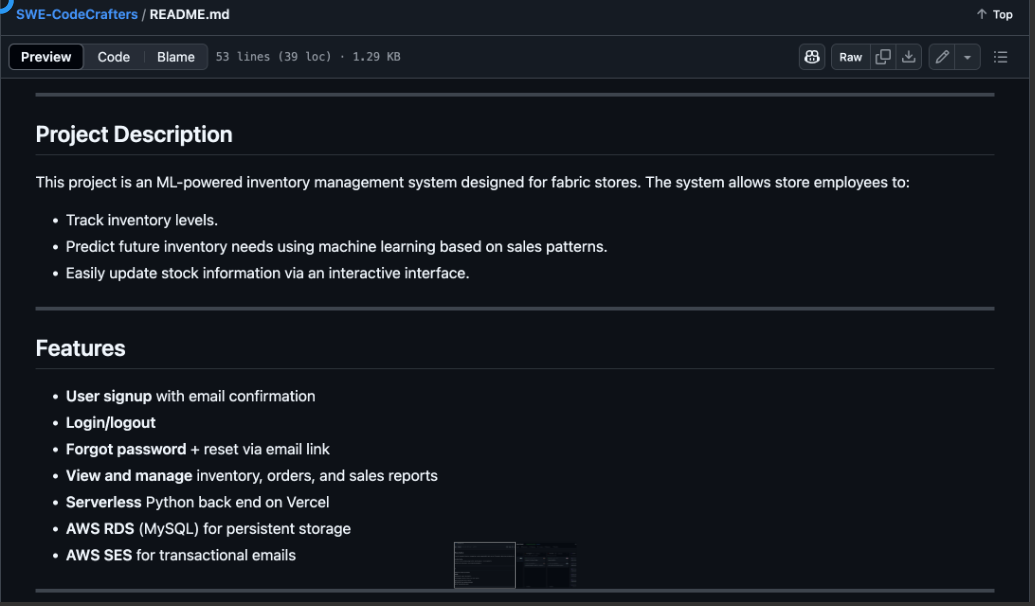
|  |  |
| --- | --- |
| Test ID | TC\_CREATE\_001 |
| Purpose of Test | Verify that a new user can successfully create an account with valid data. Covers REQ-REG-01. |
| Test Environment | OS: Windows 11   Browser: Chrome  Backend: Flask API  DB: MySQL  Test user must not already exist |
| Test Steps | 1.Navigate to the registration page  2.Enter valid username, email, and strong password  3.Click "Create Account” |
| Test Input | Username: newuser  Email: [newuser@example.com](mailto:newuser@example.com)  Password: Passw0rd! |
| Expected Result | Account is created successfully; success message or redirect to dashboard shown. |
| Likely Problems/Bugs Revealed | Email/username conflict not handled, weak password accepted, DB errors, form submission fails. |

**Login**

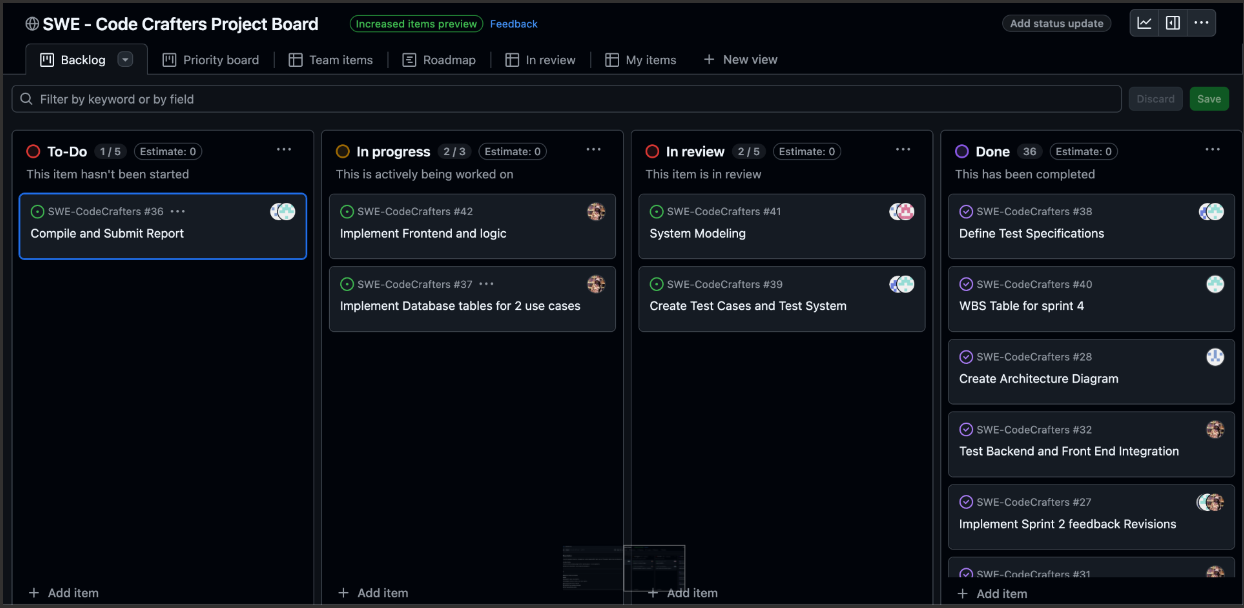
|  |  |
| --- | --- |
| Test ID | TC\_LOGIN\_001 |
| Purpose of Test | Verify that a registered user can successfully log in using correct credentials (REQ-LOGIN-01). |
| Test Environment | OS: Windows 11   Backend: Flask API  DB: User must be registered beforehand |
| Test Steps | 1.Go to login page  2.Enter valid username and password  3.Click “Login”  4.Navigate to the registration page  5.Enter valid username, email, and strong password  6.Click "Log In” |
| Test Input | Username: newuser  Password: Passw0rd! |
| Expected Result | |  | | --- | | User is logged in and redirected to dashboard. | |  | |
| Likely Problems/Bugs Revealed | Wrong redirect, session not created incorrect credential validation. |

**Section 7: Appendix**

**Appendix A: GitHub README Screenshot**



**Appendix B: GitHub Project Board/ KanBan Screenshot**



**Appendix C: Direct Links**

**GitHub Repository:** [**https://github.com/Amisu-Tayo/SWE-CodeCrafters**](https://github.com/Amisu-Tayo/SWE-CodeCrafters)

**Project Board/KanBan Link:** [**https://github.com/Amisu-Tayo/SWE-CodeCrafters/projects?query=is%3Aopen**](https://github.com/Amisu-Tayo/SWE-CodeCrafters/projects?query=is%3Aopen)