Software Engineering & Agile Development

Project Name:

Stock Maintenance System

Team Members:

Arpitha - 2022BCSE07AED852

Anand Theertha - 2022BCSE07AED854

Stock Maintenance System

Stock Maintenance System

Aim: To design and implement a Stock Maintenance System that automates inventory tracking, minimizes manual errors, monitors stock levels, generates alerts for low inventory, and maintains accurate supplier information.

Requirements:

- **Operating System:** Windows 7/10
- **Front End:** J2EE or any equivalent software
- **Back End:** MySQL Server or any equivalent
- **IDE Used:** NetBeans or any equivalent
- **Hardware Requirements:**
    - Processor: i3 or higher
    - RAM: 4 GB
    - Hard Disk: 500 GB

SUGGESTED READING (Theoretical Background):

The theoretical background for a **Stock Maintenance System** includes understanding inventory management concepts, database design, and software engineering practices. Key topics include:

- **Inventory Management Principles:** Techniques like Just-In-Time (JIT) inventory, Economic Order Quantity (EOQ), and safety stock calculation.
- **Software Development Life Cycle (SDLC):** Phases including requirement gathering, system design, coding, testing, and deployment.
- **Database Design:** Concepts like normalization, indexing, ER diagrams, and efficient data querying for handling inventory datasets.
- **UML Diagrams:** Understanding system interactions using use-case, class, activity, sequence, and state diagrams.
- **Testing Techniques:** Applying white box and black box testing methods to validate system functionality and reliability.

Procedure/Step-by-Step Instructions:

1.Problem Statement:

Manual inventory tracking often leads to errors, mismanagement of stock, delayed restocking, and financial losses. Businesses face challenges in maintaining accurate inventory records, tracking supplier transactions, and generating timely reports. The proposed Stock Maintenance System aims to address these issues by offering a digital solution that monitors stock inflow, outflow, and supplier information.

2. **Preparation of Software Requirement Specification (SRS) Document:**

- **User Requirements:**
  - **Admin:** Full access to stock management, supplier details, and report generation.
  - **Staff:** Limited access to view stock and generate basic reports.
- **Functional Requirements:**
  - Secure login with role-based access control.
  - Add, update, delete, and view stock details.
  - Manage supplier information, including contact and transaction history.
  - Monitor stock levels, reorder points, and generate alerts.
  - Generate various reports: stock status, reorder lists, and transaction history.
- **Non-functional Requirements:**
  - **Scalability:** Adapt to growing inventory and business needs.
  - **Data Security:** User authentication and secure data storage.
  - **Usability:** Intuitive and user-friendly interface.
  - **Reliability:** Consistent performance with minimal downtime.

3. **Preparation of Software Configuration Management (SCM) Software Requirements:**

- **Version Control:**
  - Use GitHub or GitLab to track code changes and maintain different versions.
  - Create branches for development, testing, and deployment.

- **Backup and Recovery:**
    - Schedule regular backups to secure data against loss.
    - Implement recovery mechanisms for system failures.
- **Change Management:**
    - Document and track modifications with version history.
    - Conduct code reviews before implementing major changes.
- **Development Environment:**
    - IDE: NetBeans or Eclipse.
    - Programming Language: Java/J2EE.
    - Database: MySQL Server or Oracle.

4. **Study and usage of any Design phase CASE tool** : STARUML or Any Equivalent

How to Install StarUML on Windows 10

Star UML is a UML (**Unified Modeling Language**) tool, introduced by MKLab. It is an open- source modeling tool that supports the UML framework for system and software modeling. StarUML is based on UML version 1.4, it provides 11 different types of diagrams and it accepts UML 2.0 notation. Version 2.0 was released for beta testing under a property license.

StarUML is actively supporting the **MDA (Model Driven Architecture).** It approaches by supporting the UML profile concept and allowing it to generate code for multiple languages. It also provides a number of bug fixes and improved compatibility with the modern versions of the Windows Operating System.

Diagram Types in StarUML

1. Use Case Diagram
2. Activity Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. Class Diagram
6. State chart Diagram

7. Component Diagram
8. Deployment Diagram

Features of StarUML

1. It supports multi-platform such as macOS, Windows, and Linux.
2. It involves UML 2.x. standard compliant.
3. Includes Entity-Relationship diagram (ERD), Data-flow diagram (DFD), and Flowchart diagram.
4. It creates multiple windows.
5. It has modern UX and dark and light themes.
6. Featured with retina (High-DPI) display support.
7. Includes model-driven development.
8. It has open APIs.
9. Supports various third-party extensions.
10. Asynchronous model validation.
11. It can export to HTML docs.

Steps to Download and Install StarUML

Step 1: Go on the browser, type in the URL "StarUML"

Step 2: Click on the very first search "Download-StarUML".

Step 3: There will be 3 Operating Systems (OS) options, click on the option as per the devise OS.

Step 4: Now, right-click on the downloaded file, select "Show in Folder" option. Step 5: Click on the open file, a popup window opens, click on the "Yes" button.

Step 6: Installation gets started. After installation popup opens to ask to buy a license. If you want to click on the “Buy Now” button or else close that window. StarUML is ready to use.

Performing the Design by using any Design phase CASE tools

CASE Tool: StarUML

Use Case Diagram: The Stock Maintenance System use cases are:

Actors:

1. Administrator
2. Customer
3. Supplier

Use Cases:

1. Product details
2. Purchase details
3. Sales details
4. Stock details
5. Purchase the product
6. Supply the product

Actors:

1. **Administrator:**
    - Manages overall stock maintenance activities.

- Responsible for viewing and updating product, purchase, sales, and stock details.
- Oversees supplier and customer interactions.

2. **Customer:**
    - Requests product details.
    - Purchases products.
    - Checks product availability through stock details.

3. **Supplier:**
    - Supplies products to the system.
    - Updates the administrator about the delivery status.
    - Can view purchase orders from the administrator.

Use Cases:

- **Product Details:**
    - Initiated by Admin to add or update product information.
    - **Precondition:** Admin is authenticated.
    - **Normal Flow:** Admin enters product name, category, and price.
    - **Post Condition:** Product details are successfully saved.
- **Purchase Details:**
    - Initiated by Admin to document product purchases.
    - **Precondition:** Purchase order is valid.
    - **Normal Flow:** Admin records product name, quantity, and supplier.
    - **Post Condition:** Purchase record is updated in the system.
- **Sales Details:**
    - Initiated by Admin to record sales transactions.
    - **Precondition:** Sufficient stock is available.
    - **Normal Flow:** Admin records product sale details.
    - **Post Condition:** Stock quantity is reduced.
- **Stock Details:**

- Accessed by Admin or Staff to check stock levels.
- **Precondition:** User has viewing access.
- **Normal Flow:** View product quantity, reorder points, and stock status.
- **Post Condition:** Current stock information is displayed.

- **Purchase the Product:**
  - Performed by Customer to buy products.
  - **Precondition:** Product is available in stock.
  - **Normal Flow:** Customer selects product and makes payment.
  - **Post Condition:** Transaction is recorded.

- **Supply the Product:**
  - Handled by Supplier to deliver ordered products.
  - **Precondition:** Supplier receives purchase order.
  - **Normal Flow:** Supplier delivers products to the Admin.
  - **Post Condition:** Stock is updated upon receipt.

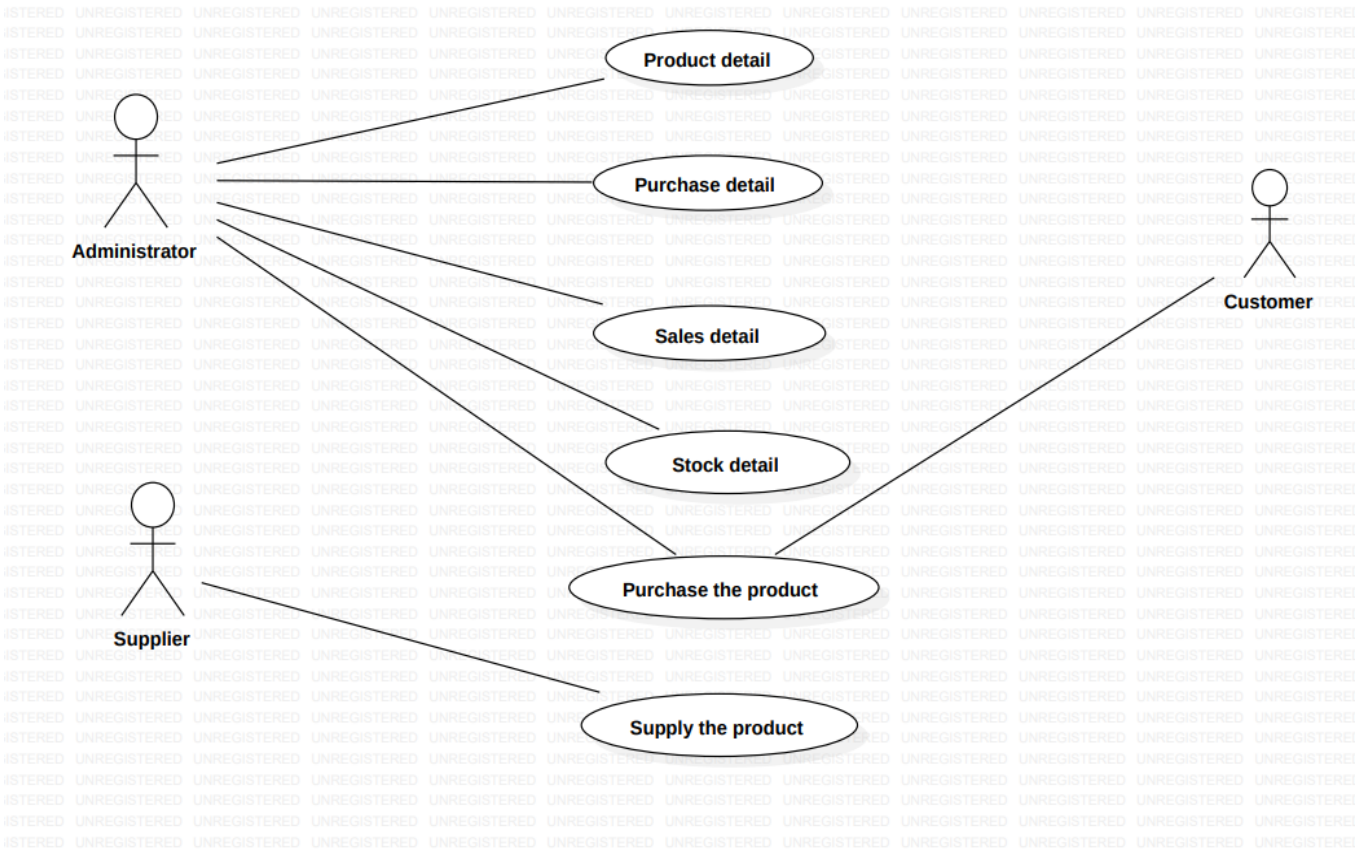


Figure 1: Use Case Diagram for Stock Maintenance System.

Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions, supporting choice, iteration, and concurrency. In the Unified Modeling Language (UML), they describe business and operational workflows. An activity is represented by a rounded box containing the name of the operation. These diagrams demonstrate the overall flow of control and system behaviour.

Activity Diagram for Inventory Management:

- Initiate inventory check.
- Update new items to inventory.
- Detect expired or low-stock products.
- Add information to the user catalog.

Activity Diagram for Purchasing:

- Initiate a purchase order.
- Verify supplier details.
- Process the purchase and update inventory.

Activity Diagram for Sales:

- Verify stock availability.

- Process customer order.
- Update inventory and record sales.

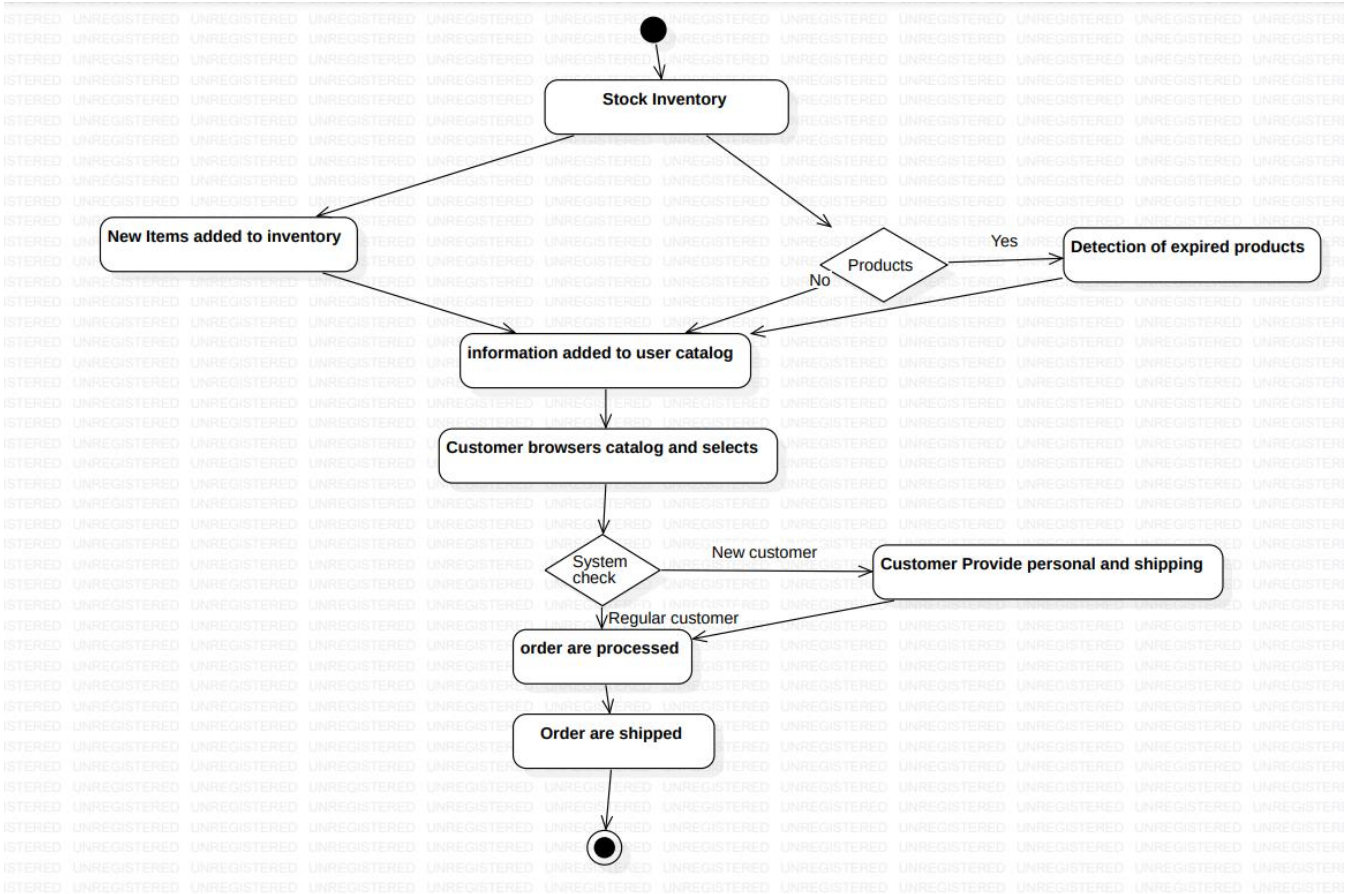


Figure 2: Activity Diagram for Stock Maintenance System.

Sequence Diagram :

A sequence diagram depicts the interaction and flow of messages between various system components during a specific scenario. Below is the explanation based on the sequence diagram displayed:

1. **Login Process:**
    - The user (Admin or Staff) enters their login details.
    - The system verifies these details through the Verifier component.
    - The Verifier checks the login validity with the Database.
    - If valid, the system grants access.
2. **Stock Availability Check:**
    - The system sends a request to check stock availability to the Verifier.
    - The Verifier verifies the request and consults the Database.

- The Database returns the availability status to the Verifier.
- The Verifier sends the status back to the system.

3. **Stock Management:**
   - If the user has appropriate access, they enter stock details (addition, update, deletion).
   - The system processes these details and maintains updated stock information in the Database.

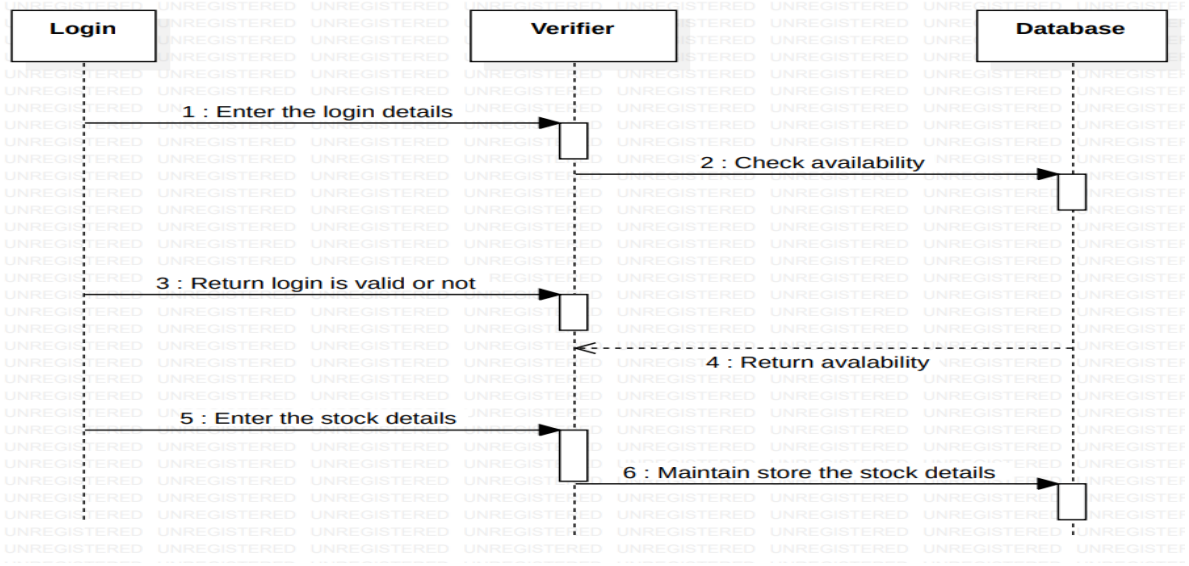


Figure 3: Sequence Diagram for Stock Maintenance System.

Collaboration Diagram:

A collaboration diagram, also known as a communication diagram, emphasizes the structural organization of objects that send and receive messages. It showcases the interaction between objects and the sequence of message exchanges for a particular scenario.

1. **Login Interaction:**
   - The user enters login details to access the system.
   - The login details are sent to the Verifier for validation.
   - The Verifier checks the details with the Database to confirm authenticity.
   - A response is sent back indicating whether the login is valid or not.

2. **Stock Availability Check:**
   - After a successful login, the user can enter stock details.
   - The system requests the Verifier to check the availability of the stock.
   - The Verifier consults the Database and returns the stock availability status.

3. **Stock Management:**
   - If the stock is available, the user can update or enter new stock details.
   - The system maintains and stores these updated stock details in the Database for future reference.

1. **Enter the login details** - Initiated by the user to the system.

2. **Check availability** - The Verifier checks the Database.
3. **Return availability** - The Database sends back the result.
4. **Return login is valid or not** - The Verifier sends the validation status to the user.
5. **Enter the stock details** - The user inputs the stock details.
6. **Maintain and store the stock details** - The Database updates and stores the information.

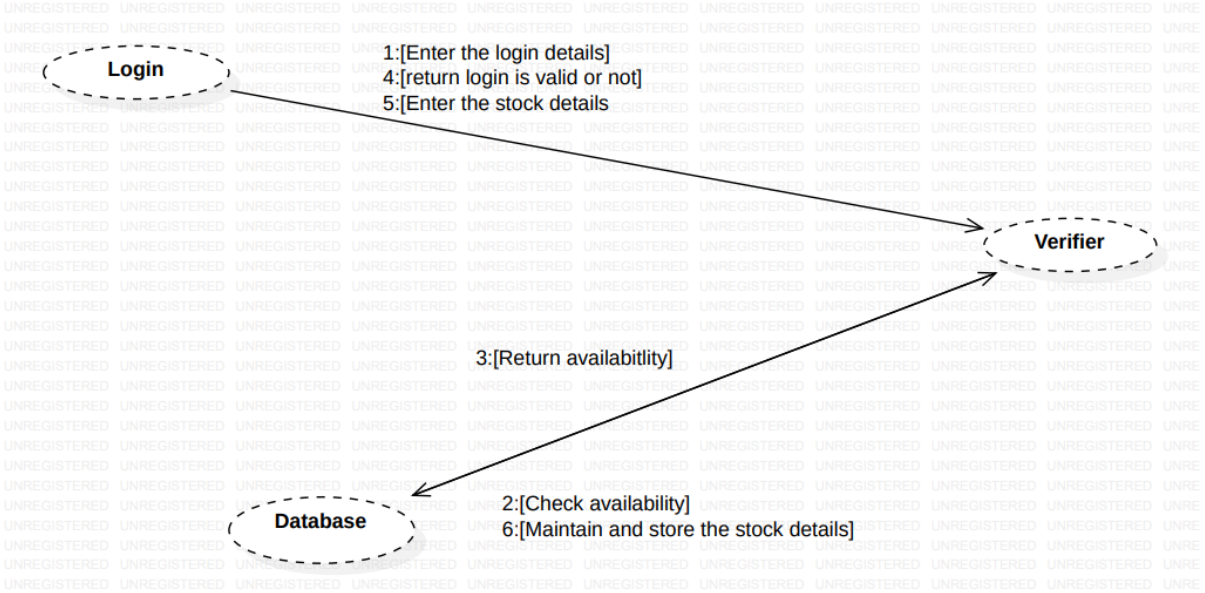


Figure 4: Collaboration Diagram for Stock Maintenance System.

Class Diagram:

A class diagram is a visual representation of the system's structure, showing its classes, attributes, methods, and relationships. It helps understand the system's overall design and interactions between different components.

Classes and Their Details:

1. **Product Details:**
   - **Attributes:**
     - Pcode: int — Product Code
     - Price: int — Price of the product
     - Pname: string — Product Name
     - OpeningStock: int — Initial stock available
   - **Methods:**
     - Add() — Adds new products to the inventory.
     - Delete() — Deletes a product from the inventory.
     - Update() — Updates product information.
     - Clear() — Clears the product details.
     - Exit() — Exits from the product module.

2. **Purchase Details:**
   - **Attributes:**
     - Pdate: date — Date of purchase
     - Price: int — Purchase price

        - Pcode: int — Product code
        - Sname: string — Supplier name
        - Pqty: int — Quantity purchased
    - **Methods:**
        - Purchase save() — Saves purchase information.
        - Purchase delete() — Deletes a purchase record.
        - Pedit() — Edits purchase details.
        - Exit() — Exits from the purchase module.

3. **Sales Details:**
    - **Attributes:**
        - Sdate: date — Sales date
        - Cname: string — Customer name
        - Pcode: int — Product code
        - Price: int — Selling price
        - Qty: int — Quantity sold
    - **Methods:**
        - Add sales() — Adds sales details to the system.
        - Exit() — Exits from the sales module.
        -

4. **Stock Details:**
    - **Attributes:**
        - Date: date — Start date for stock check
        - Date2: date — End date for stock check
    - **Methods:**
        - Viewstock() — Views the current stock level.
        - Cancel() — Cancels the current operation.

Relationships Between Classes:

- The Product Details class is associated with both Purchase Details and Sales Details. This relationship is crucial as it maintains the stock balance.
- Purchase Details and Sales Details interact with Stock Details to ensure real-time updates of inventory.
- The use of arrows indicates the direction of interaction between the classes.

- The Product Details class serves as the core for stock management.
- Purchase Details update the inventory by adding stock, while Sales Details decrease the inventory.

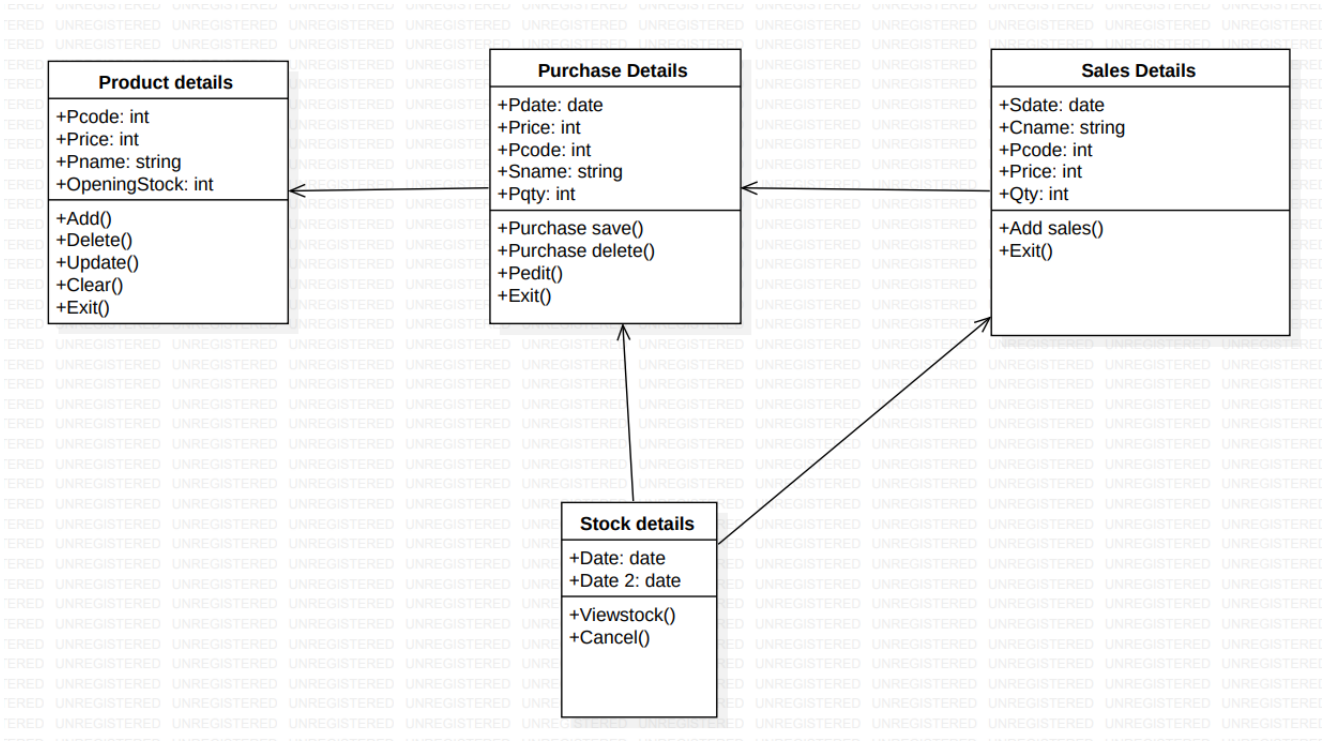


Figure 5: Class Diagram for Stock Maintenance System.

State Chart :

A State Chart Diagram (or State Machine Diagram) is used to represent the dynamic behavior of a system. It shows various states of an object and transitions between these states based on events.

1. **Initial State:**
   - The solid black circle at the top indicates the starting point of the state diagram.
   - The system begins with checking the **product quantity**.
2. **Search State:**
   - The system first searches the inventory to check the available stock quantity.
   - Based on the result, there are two possible transitions:
     - If the quantity is **less**, the system moves to the **Add** state.
     - If the quantity is **enough**, it directly reaches the **final state**.
3. **Add State:**
   - When the stock is insufficient, the system enters the **Add** state.
   - Here, additional stock is added to ensure availability.
4. **Final State:**
   - The filled black circle at the end represents the final state of the process, indicating the completion of stock verification and management.

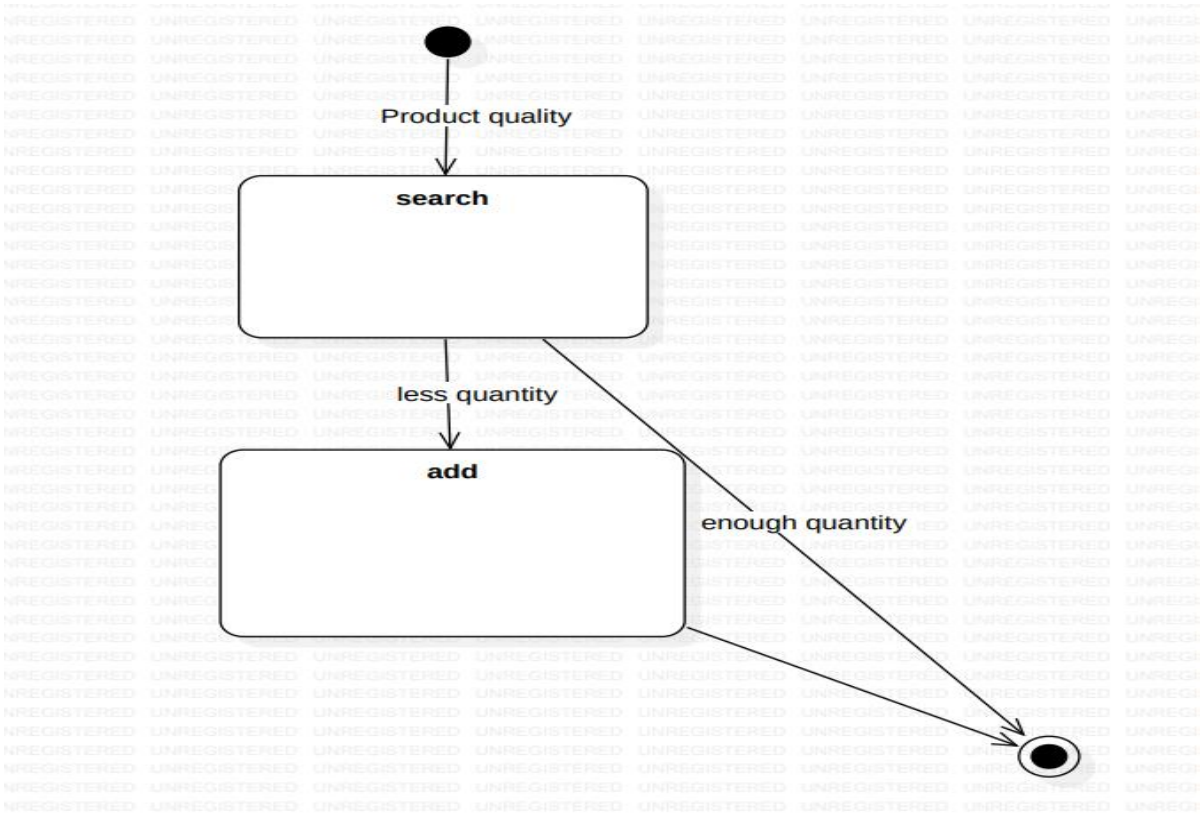


Figure 6: State chart Diagram for Stock Maintenance System

Component Diagram:

A Component Diagram is used to visualize the organization and relationships between various software components in a system. It shows how components interact with each other to perform a particular function.

1. **Components Involved:**
    - **Login:** Represents the user authentication module, where authorized users can access the system.
    - **Stock Details:** Handles all functionalities related to product stock, such as adding, updating, and deleting product information.
    - **Database:** Stores and manages all data related to the products, purchases, and sales.
2. **Interactions:**
    - The **Login** component connects to the **Stock Details** component, ensuring only authenticated users can access the stock information.
    - The **Stock Details** component interacts with the **Database** for retrieving and updating stock data.
    - The arrow direction shows the data flow and dependency between components.

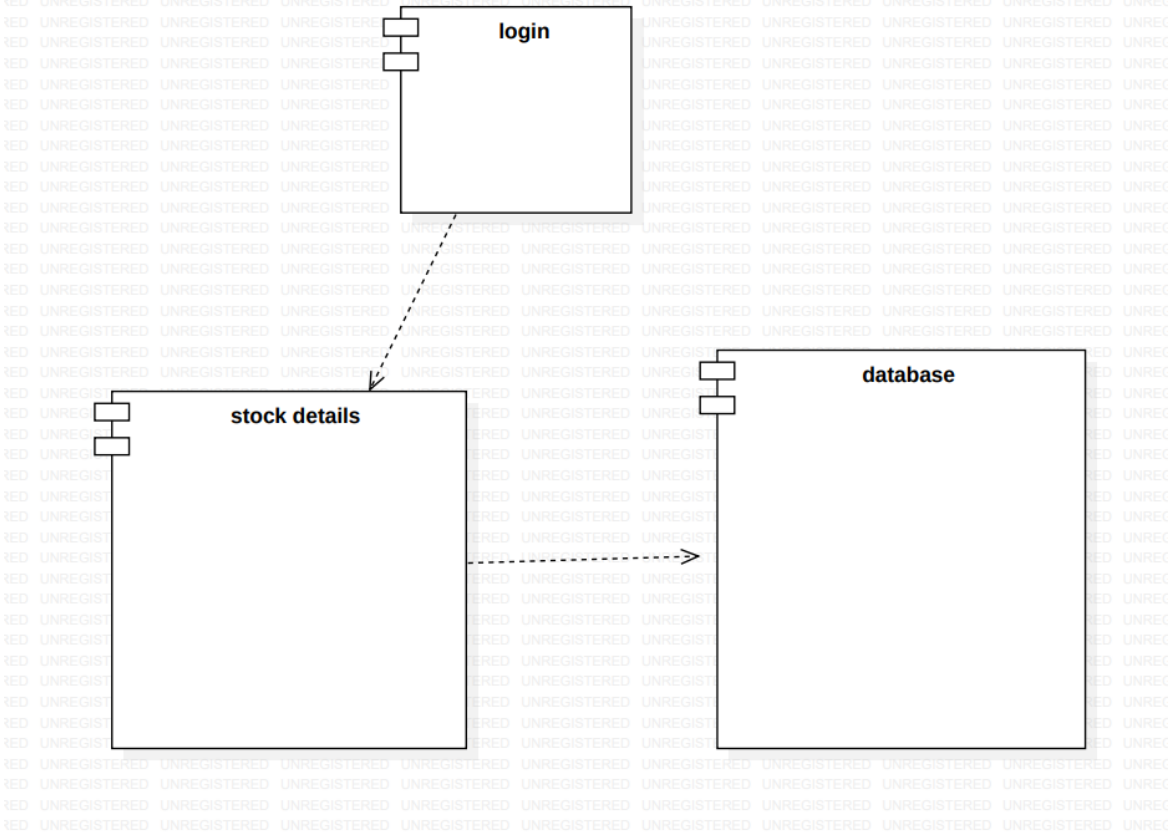


Figure 7 : Component Diagram for Stock Maintenance System.

Deployment Diagram :

A Deployment Diagram visualizes the physical architecture of a system. It shows the hardware components (nodes), software components (artifacts), and their relationships.

1. **Nodes Involved:**
    - **Product Report:** Represents the component that handles the generation of product-related reports.
    - **Purchase:** Manages the purchase process and related transactions.
    - **Sales Report:** Generates sales-related reports and tracks sales data.
    - **Stock Report:** Creates and manages stock-level reports.
    - **Printer:** Used to print the generated reports.
2. **Connections:**
    - All components (**Product Report**, **Purchase**, **Sales Report**, and **Stock Report**) are connected to the **Printer** node.
    - This connection shows that all these components can generate reports and send them to the printer for hard copies.

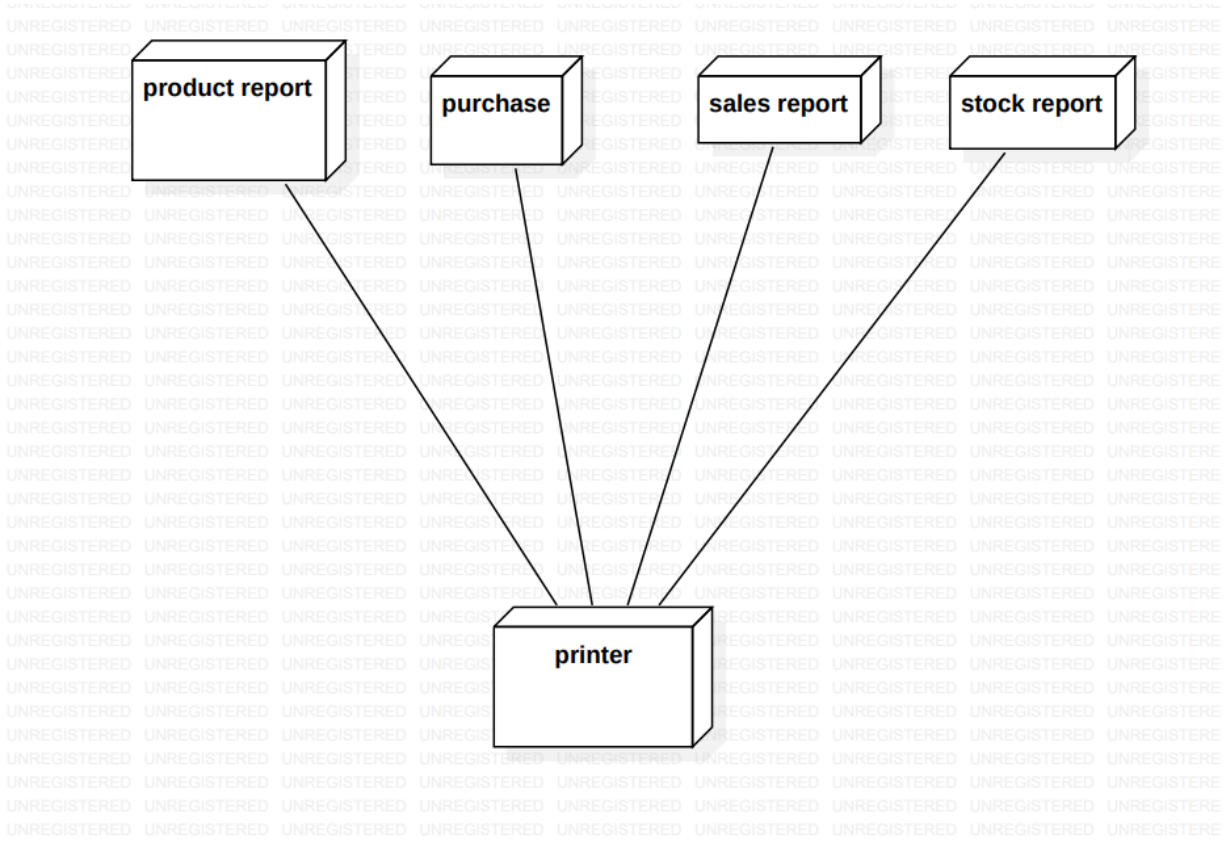


Figure 8: Deployment Diagram for Stock Maintenance System.

Test Cases:

Test Cases for Login Page:

Sl. No	Test Case Description	Input Data	Expected Result	Test Result
1	Valid Login Credentials	Valid username and password	User should be successfully logged in and redirected to the dashboard	Successful
2	Invalid Username	Invalid username with valid password	Error message should be displayed indicating incorrect credentials	Successful
3	Invalid Password	Valid username with invalid password	Error message should be displayed indicating incorrect credentials	Successful
4	Empty Username or Password	Leave username or password field blank	Error message should prompt to fill all required fields	Successful
5	SQL Injection Test	Enter SQL code in username or password field	System should block and prevent unauthorized access	Successful

Test Cases for Stock Maintenance System:

Sl. No	Test Case Description	Input Data	Expected Result	Test Result
1	Add New Stock Item with Valid Data	All required fields with valid item details	Stock item should be successfully added to the database	Successful
2	Add Stock Item with Missing Fields	Leave required fields blank	System should display an error message to fill all required fields	Successful
3	Update Existing Stock Item	Modify item details with valid data	Updated item details should be saved and displayed correctly	Successful
4	Delete Stock Item	Select an existing item and click delete	Stock item should be removed from the database with a confirmation message	Successful
5	Search for Stock Item	Search using valid item name or ID	Relevant search results should be displayed	Successful
6	Invalid Stock Search	Search using invalid or non-existent item details	System should display a message like "No results found"	Successful
7	View Low Stock Items	Access the low stock alert section	System should list items with stock below the threshold	Successful
8	Stock Transaction Log	View transaction history for an item	System should display accurate and complete transaction history	Successful

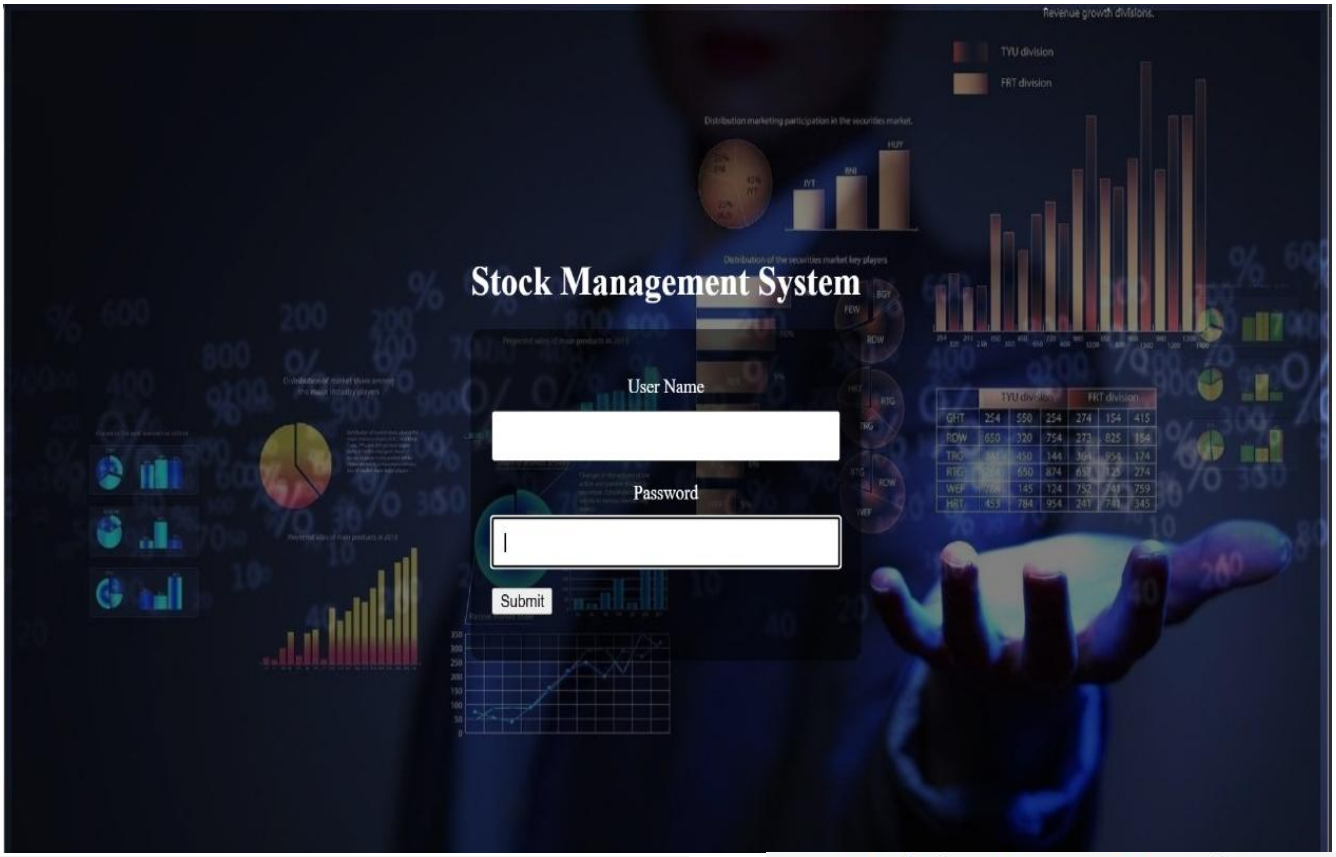
Inferences:

This experiment identifies the specific requirements for managing inventory operations, including tracking product details, purchase records, and sales transactions. The Stock Maintenance System is designed to handle stock efficiently, prevent overstocking or shortages, and ensure data accuracy. The use of UML diagrams helps visualize the system's flow, while proper testing techniques validate its functionality. The result is a scalable, efficient, and user-friendly system with accurate data management and cost-effective resource utilization.

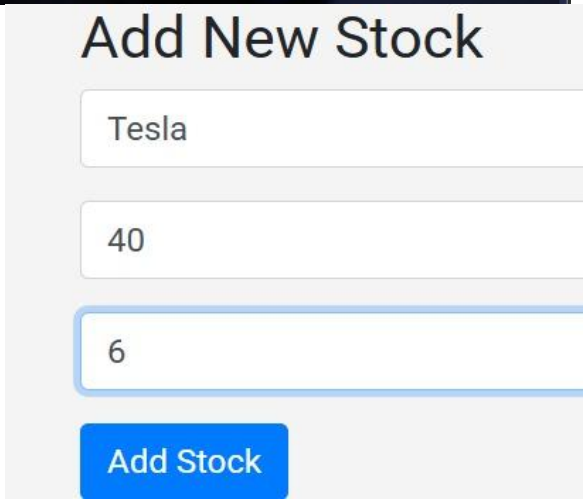
Sample Output/Result:

- Successful login with valid credentials.
- Accurate addition, update, and deletion of stock items.
- Proper calculation of stock levels and reorder points.
- Seamless generation of purchase and sales reports.
- Error messages for invalid login attempts or insufficient stock.

This approach ensures the effective performance of the Stock Maintenance System, reducing manual errors and supporting business operations efficiently.


Stock Management System
User Name
Password
Submit


Add New Stock
Stock Name
Stock Price
Stock Change
Add Stock


Add New Stock
Tesla
40
6
Add Stock

Hindustan Unilever	₹2422.30	0.34% ▲	Delete
Bajaj Auto	₹3901.22	0.25% ▲	Delete
Larsen & Toubro	₹1658.37	-0.63% ▼	Delete
Dr Reddy's Laboratories	₹4505.33	0.52% ▲	Delete
Bharti Airtel	₹541.11	0.8% ▲	Delete
Axis Bank	₹868.75	0.6% ▲	Delete
Tesla	₹41.27	0.91% ▲	Delete

Add New Stock

Tesla

40

6

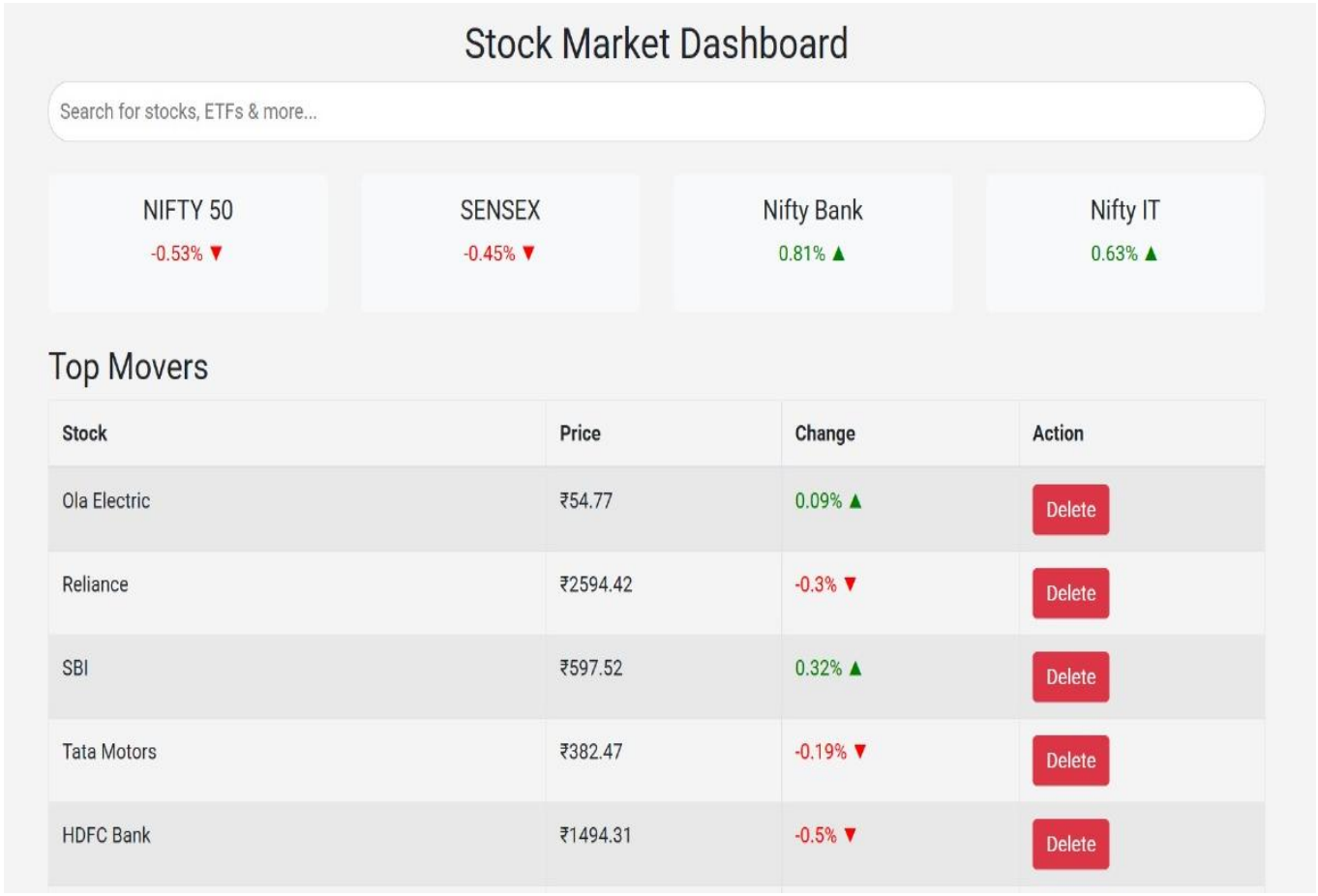
Stock Market Dashboard

axis

NIFTY 50	SENSEX	Nifty Bank	Nifty IT
-0.04% ▼	0.4% ▲	0.78% ▲	-0.47% ▼

Top Movers

Stock	Price	Change	Action
Axis Bank	₹863.63	0.09% ▲	Delete


Stock Market Dashboard
Search for stocks, ETFs & more...
NIFTY 50
-0.53% ▼
SENSEX
-0.45% ▼
Nifty Bank
0.81% ▲
Nifty IT
0.63% ▲
Top Movers
Stock
Price
Change
Action
Ola Electric
₹54.77
0.09% ▲
Delete
Reliance
₹2594.42
-0.3% ▼
Delete
SBI
₹597.52
0.32% ▲
Delete
Tata Motors
₹382.47
-0.19% ▼
Delete
HDFC Bank
₹1494.31
-0.5% ▼
Delete