SENIOR DESIGN PROJECT
KIET/SE/SDP/Fall-2021/2025

Supply Chain Management with AI Chatbot

Group Members

Student Name	Enrollment#
Student-1 Hashir Ahmed Buriro	BE-3-21-13948
Student-2 M Irteza Waseem	BE-3-21-13952
Student-3 Abdullah Sikandar	BE-3-21-13694
Student-4 Ubaid Rashid	BE-3-21-14229

Supervised by

Engr. Ghulam Qasim

College of Engineering (COE)

Acknowledgments

In the name of Allah, the Most Merciful, the Most Gracious, we begin by expressing our deepest gratitude to the Almighty, who has blessed us with the gift of knowledge and the strength to pursue our final year project. It is through His guidance and mercy that we were able to embark on and complete this important academic journey. We also pay our highest respects to the last and most beloved messenger, Prophet Muhammad (Peace be upon him), whose life and teachings continue to inspire and guide humanity.

We are sincerely thankful to the **Karachi Institute of Economics and Technology** for providing us with the platform, resources, and supportive environment that have been essential to the success of our project. This achievement marks a significant milestone in our academic and professional growth.

Our heartfelt appreciation goes to our supervisor, **Engr. Sir Ghulam Qasim**, for his unwavering support, insightful guidance, and continuous encouragement throughout the course of this project. His mentorship played a vital role in shaping our ideas and bringing our vision to life.

We would also like to thank our classmates and team members whose encouragement, feedback, and collaboration kept us motivated and helped us improve every step of the way. Their support made the process more meaningful and enjoyable.

A special note of thanks goes to our co-leader, **Engineer Asad Ali Rahimo**, for generously sharing his time, experience, and support. His contributions were instrumental in helping us reach our goals, and we are truly grateful for his involvement.

Finally, we extend our sincere thanks to everyone who supported us—directly or indirectly—throughout this journey. Your help, kindness, and prayers have meant a great deal to us. May Allah continue to guide and bless us all with success in our future endeavors.

Abstract

In today's globalized business landscape, efficient supply chain management is essential for staying competitive and meeting customer expectations. It involves streamlining procurement, production, inventory, and distribution to improve efficiency and reduce costs. However, challenges like manual workflows, disconnected systems, and limited real-time visibility continue to hinder performance. To overcome these issues, this project aims to develop an integrated supply chain management platform that unifies processes, stakeholders, and data sources to deliver end-to-end visibility and control. The proposed solution leverages IoT, artificial intelligence, and big data analytics to offer real-time shipment tracking, automated inventory updates, predictive insights, supplier performance monitoring, and smart contract-based transactions. Its user-friendly interface empowers stakeholders to make data-driven decisions, ultimately reducing lead times, minimizing stockouts, cutting costs, and improving agility. Additionally, the project offers students valuable hands-on experience in building enterprise-level software, enhancing their skills and employability in IT and supply chain domains.

TABLE OF CONTENTS

TABLE OF FIGURES

List of Tables

1. INTRODUCTION

1.1 Introduction

This AI-based Integrated Supply Chain Management System with an embedded chatbot is designed to transform the supply chain industry by leveraging artificial intelligence and automation. The system provides a user-friendly and seamless experience tailored to the specific needs of supply chain operations, addressing common challenges such as inventory mismanagement, inefficient workflows, and communication delays. It supports key processes like procurement, inventory control, and distribution, helping businesses streamline operations and improve efficiency. Powered by AI, the system enables real-time data access, smart automation, and intelligent decision-making. A robust database management system ensures data security, accessibility, and integrity, preventing issues like stock shortages or overstocking. By automating routine tasks such as report generation and billing, it reduces manual errors and increases productivity. Designed especially for industries like manufacturing, retail, and agriculture, this solution enhances transparency, reduces operational costs, and empowers businesses to respond more flexibly to market changes—making it an essential tool for maintaining a competitive edge.

1.2 Problem Statement

Well-managed supply chain efficiently meets customer demand in a cost-effective manner and gives business a competitive advantage. But conventional supply chain systems have a lot of problems suppressing operational efficiency:

1.2.1 Poor Inventory Management

Challenges there are such as overstocking/understocking, and lack of real-time updates that often result in financial loss and missed opportunities.

1.2.2 Manual Processes and Errors

Billing, report generation, communication, etc., are all tedious work that is vulnerable to human error and takes a lot of time and resources.

1.2.3 Communication Gaps

The coordination between suppliers, retailers, and other stakeholders is often poor leading to delays and improper management of resources.

1.2.4 Limited Accessibility

Many existing systems are complex and not user-friendly, making them inaccessible to users with limited technical expertise.

1.2.5 Absence of Intelligent Assistance

There are no smart components available in conventional systems that can help users with decision-making, data analysis, or recommendations so businesses are left with manual analysis.

1.3 Background / Literature Review

1.3.1 Forecasting Supply Chain Demand Approach Using Knowledge Management Processes and Supervised Learning Techniques

1.3.1.1 REFRENCE: Menaouer Brahami, LABAB Laboratory, National Polytechnic School of Oran - M. Audin, Algeria

1.3.1.2 Published Year: January 2022

In this **Article**, today's context (competition and knowledge economy), ML and KM on the supply chain level have received increased attention aiming to determine long and short-term success of many companies. However, demand forecasting with maximum accuracy is absolutely critical to invest in various fields, which places the knowledge extract process in high demand. In this paper, the authors propose a hybrid approach of prediction into a demand forecasting process in supply chain based on the one hand, on the processes analysis for best professional knowledge for required competencies. And on the other hand, the use of different data sources by supervised learning to improve the process of acquiring explicit knowledge, maximizing the efficiency of the demand forecasting, and comparing the obtained efficiency results. Therefore, the results reveal that the practices of KM should be considered as the most important factors affecting the demand forecasting process in supply chain. The classifier performance is examined by using a confusion matrix based on their accuracy and Kappa value.

1.3.2 Machine Learning Demand Forecasting and Supply Chain Performance

1.3.2.1 REFERENCE: Javad Feizabadi (2020): Machine learning demand forecasting and supply chain performance, International Journal of Logistics Research and Applications, DOI: 10.1080/13675567.2020.1803246

1.3.2.2 Published Year: 04 Aug 2020

In many supply chains, companies operating at the upstream stages often struggle with inefficiencies caused by demand information getting distorted as it moves through different levels of the chain. This distortion leads to variance amplification, where even small changes in demand at the customer level result in larger fluctuations further up the chain. While earlier research suggests that advanced forecasting methods—especially those using machine learning—can help reduce this effect and improve overall supply chain performance, there is still limited understanding of the actual scale of improvement in measurable terms. This study develops a hybrid forecasting approach that combines machine learning techniques, specifically ARIMAX and Neural Networks, using both time-series data and explanatory variables. The model was tested and validated in a real-world setting involving a functional product and a steel manufacturing firm. The performance of traditional forecasting methods was compared to the machine learning-based approach, and the differences in supply chain performance were analyzed statistically. The findings offer valuable insights for both practical applications and theoretical development in supply chain forecasting.

1.4 Problem in existing system

Traditional supply chain management systems cannot cope with the dynamic supply and demand of modern business. Here are the main problems with existing systems:

1.4.1 Lack of Real-Time Data

Coordination slows down because many systems do not provide Realtime update of inventory levels, ordered statuses and resource availability.

1.4.2 Manual Dependency

Dependence on manual processes like generating invoices and creating reports leads to a significant error rate, increased effort, and slow operations.

1.4.3 Improper Communication

Poor communication among suppliers, outlets, and different stakeholders hampers coordination and reasons delays in choice-making.

1.4.4 Constrained Automation

Existing systems lack sufficient automation abilities, leaving essential duties at risk of human errors and inefficiencies.

1.4.5 Complex Interfaces

Many systems aren't consumer-friendly and require widespread training, making them inaccessible for customers with restricted technical expertise.

1.4.6 Excessive Operational expenses

Inefficiencies in existing systems frequently result in better operational costs due to wasted resources, delays and mistakes

1.4.7 Lack of intelligent assistance

Still using traditional methods does not provide intelligent tools to provide insights, suggestions o r decisions, which makes the process too time consuming and unproductive.

1.5 Proposed Solution

1.5.1 Features of the project

1.5.1.1 Aim:

Develop an integrated web-based Supply chain management system that used Artificial intelligence to enhances efficiency, reduces costs, makes customer satisfy, manage time and improves transparency.

1.5.1.2 Objectives:

Implement real-time tracking of inventory and shipments. Use of artificial intelligence to enhance communication and listing. Make order management efficient and less time consuming. Prioritize the items after analysis from real time data. Stock management updating through artificial intelligence. AI Chatbot to help or assist customer and user queries.

1.5.1.3 Forecasting for prediction in sales:

Implement forecasting algorithms to predict the sales of future.

1.5.1.4 Real –time inventory automation:

Monitor the stock availability driven from real time data.

1.5.1.5 Order Management Automation:

Automates order processing from suppliers to customers.

1.5.1.6 Supplier and Vendor Collaboration:

1.5.1.6.1 Facilitates seamless communication between suppliers, manufacturers, and distributors.

1.5.1.6.2 Enables automated replenishment and coordination for just-in-time delivery.

1.5.1.7 AI Chatbot:

Solve customer and seller queries.

1.5.1.8 Analytics and Reporting:

1.5.1.8.1 Provides data-driven insights for demand forecasting, supply chain optimization, and performance analysis.

1.5.1.8.2 Customizable dashboards for stakeholders to monitor key metrics.

1.5.2 Methodology/Algorithm

1.5.2.1 Block Diagram:

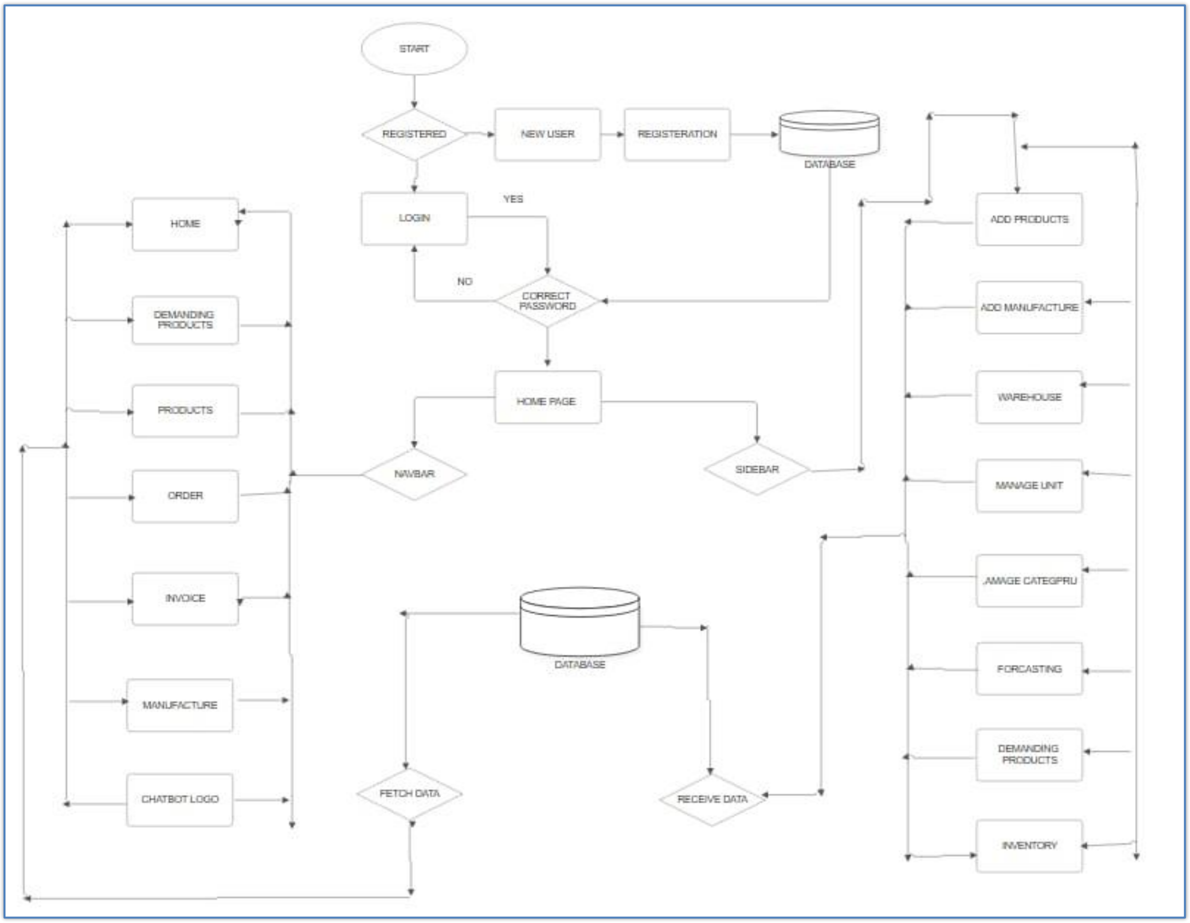


Figure 1 Block Diagram

1.5.2.1.1 Figure 1 Block Diagram Explain:

The Block diagram outlines an SCM system's workflow, starting with user login or registration. Users can browse products, place orders, and access invoices. The system integrates with a database to manage product listings and inventory, ensuring a smooth shopping experience.

1.5.2.2 Algorithm:

XGBoost is an open-source, high-performance, machine learning library applied to various supervised learning problems like regression and classification. It is based on the gradient boosting construction of models such that they provide improvements over existing errors. Key features of XGBoost are regularization against overfitting, parallel processing with faster results, scalability for processing large datasets, and flexibility on objectives. Such applications include its use in predictive modeling in the fields of finance, healthcare, and marketing.

1.5.3 Technologies to be used

1.5.3.1 Frontend Framework

1.5.3.1.1 HTML/CSS:

Utilizing Bootstrap: A responsive CSS framework for creating clean and user-friendly designs.

1.5.3.1.2 JavaScript:

Use jQuery: For simpler DOM manipulation and user notification.

1.5.3.2 Backend Framework

1.5.3.2.1 PHP:

Utilizing PHP: A popular PHP framework for building robust backend systems with built-in tools for authentication, database migrations, and routing.

1.5.3.2.2 Database MySQL:

Use phpMyAdmin: For database management via a web interface. Integrate with PHP through PDO (PHP Data Objects) for secure and flexible database interaction.

1.5.3.2.3 Scheduling AI

The system uses artificial intelligence to streamline supply chain tasks like inventory restocking by analyzing historical and real-time data to generate efficient schedules, reduce conflicts, and improve overall operational performance.

1.5.3.2.4 Python:

We Utilizes Python as the programming language, and it complements its robust ecosystem of libraries and tools in preprocessing data, training the model, and evaluation. Therefore, accuracy and efficiency in demand prediction are assured.

1.5.3.2.5 Algorithm:

XGBoost is an open-source, high-performance, machine learning library applied to various supervised learning problems like regression and classification.

1.5.3.2.6 Hosting:

XAMPP is an all-in-one local development environment for Apache, MySQL, and PHP. It allows you to run and test your PHP applications locally. PhpMyAdmin is the tool for the management of MySQLdatabases using a web interface and comes with the package in XAMPP. Tasks such as creating databases, executing SQL queries, and exporting/importing data can be carried out. With XAMPP, you are able to develop and test your PHP applications locally before you move it live to the server.

1.6 Project Scope / Deliverables

1.6.1 Project Goals and Objectives

1. The project aims to achieve a milestone of 100% completion by the end of the semester.
2. Seamless integration of various supply chain components, including inventory, suppliers, orders, and logistics, is a priority.
3. Customer satisfaction is a key objective, achieved through improved delivery timed personalized service.
4. Real-time monitoring ensures accurate stock levels.
5. AI algorithms predict future sales patterns.
6. AI chatbot assists users with queries and updates
7. Efficient order processing saves time and effort
8. AI analysis prioritizes items based on real-time data.

1.6.2 Manage Areas to be Covered

1. Add Products
2. View Category
3. Manage Category
4. Products
5. Retailer
6. Add Ware House
7. Manage Unit
8. Demanding Products
9. Manufacture
10. Add Manufacture
11. Manage Area
12. Inventory
13. Orders
14. Invoice

1.6.3 Specific Project Deliverables and Tasks

1.6.3.1 Document Deliverables:

1. Project Proposal Document
2. System Requirements Specification
3. Design Documentation
4. Implementation Documentation
5. Testing and Quality Assurance Documentation
6. User Manual

1.6.4 Source code and Executables

1. Supply Chain Management System with AI Chat Bot Source Code.
2. Compiled Executable for Testing

1.6.5 Libraries and Dependencies:

1. Python
2. XGBoost

1.6.6 MY SQL Database

2. SOFTWARE PROJECT MANAGEMENT PLAN

2.1 Project Organization

2.1.1 Project Manager:

2.1.1.1 Responsibilities:

1. Overall project oversight and coordination
2. Ensuring project goals align with client expectations
3. Resource allocation and scheduling
4. **Name:** Hashir Ahmed

2.1.2 Developer Team:

2.1.2.1 Responsibilities:

1. System implementation and coding.
2. Integration of AI chatbot features.
3. Adhering to coding standards and best practices.
4. **Team Members:** Hashir Ahmed, Ubaid Rashid, Abdullah, M. Irteza Waseem.

2.1.3 Testing Team:

2.1.3.1 Responsibilities:

1. Creating and executing test cases.
2. Verifying chatbot functionality and supply chain system reliability.
3. Collaborating with developers for issue resolution.
4. **Team Members:** Hashir Ahmed, Ubaid Rashid, Abdullah, M. Irteza Waseem.

2.1.4 Design Team:

2.1.4.1 Responsibilities:

1. Designing user interfaces for supply chain modules and chatbot.
2. Creating system design documents
3. Collaborating with developers for design implementation
4. **Team Members:** Hashir Ahmed, Ubaid Rashid, Abdullah, M. Irteza Waseem

2.1.5 Stakeholders:

2.1.5.1 Responsibilities:

1. Providing input on supply chain and chatbot requirements.
2. Offering feedback on project deliverables
3. Ensuring project alignment with business objectives

2.1.6 Communication Plan:

1. Regular team meetings and updates
2. Client meetings for feedback and clarification
3. Email and project management tools for asynchronous updates

2.1.7 Meeting Schedule:

1. Weekly team meetings on **Monday**
2. Bi-weekly client meetings on **Sunday**

2.2 Software Process Model

For our Final Year Project, Supply Chain Management System with AI Chatbot, we will adopt an **Agile software development process model**, specifically following the Scrum framework. This choice is motivated by the project's academic context, iterative requirements, and the importance of continuous collaboration with project stakeholders, including university faculty and administrators.

Identify and describe the software process model you will use in terms of the following:

2.2.1 Flow of Information and Work Products:

2.2.1.1 Backlog Refinement:

Information flows collaboratively between the project team, including developers and academic stakeholders, during backlog refinement sessions. This ensures that academic requirements are accurately captured. Work products include a refined academic feature backlog.

2.2.1.2 Sprint Planning:

Development team members, in collaboration with academic advisors, receive information about selected academic features for the upcoming sprint. Work products include a sprint backlog, detailing academic tasks and estimates.

2.2.1.3 Daily Stand-ups:

Daily communication within the development team, including updates on academic feature progress and addressing any academic considerations raised during the development process. Work products include academic insights gained from team collaboration.

2.2.1.4 Sprint Review:

Academic stakeholders actively participate in the collaborative review of the academic features produced during the sprint. Feedback from academic stakeholders is actively sought to enhance the academic components of the system.

2.2.1.5 Sprint Retrospective:

The project team engages in retrospectives to review academic processes, identifying successful practices, and areas for improvement. Work products include actionable items for refining academic processes.

2.2.2 Reviews To Be Conducted

2.2.2.1 Sprint Planning Review

1. Objective: Agree on the work items to be done during the next sprint, including user stories and tasks.
2. Participants: Scrum team, Product Owner, and optional stakeholders, for example, faculty.
3. Frequency: At the beginning of each sprint.

2.2.2.2 Daily Scrum (Daily Stand-up)

1. Objective: Track progress, identify blockers, and plan work for the day.
2. Participants: Scrum team.
3. Frequency: Daily, during the sprint.

2.2.2.3 Sprint Review

1. Objective: Present the work done during the sprint, get feedback, and confirm that the increment of the product meets the requirements.
2. Participants: Scrum team, Product Owner, and stakeholders.
3. Frequency: After the end of every sprint.

2.2.3 Major Milestones to Be Achieved

2.2.3.1 Requirement Analysis and Specification (Month 1)

1. Finalize functional and non-functional requirements.
2. Solicit inputs through stakeholder interviews.
3. Create a comprehensive project plan and timeline.

2.2.3.2 System Architecture and Design (Month 2)

1. Define system architecture, including AI chatbot integration.
2. Design the database schema and API structure.
3. Create wireframes and UI/UX prototypes of the system.

2.2.3.3 Development Phase 1: Core Features (Months 3-4)

1. Implement product demand forecasting using XGBoost.
2. Implementation of the backend supply chain workflows in terms of inventory management and order tracking.
3. Basic conversational flow of the chatbot.

2.2.3.4 Development Phase 2: Advanced Features (Months 5-6)

1. Develop AI chatbot with Machine Learning Algorithm capabilities for advanced queries
2. Integrate data visualization tools for generating forecasting reports
3. Implement role-based access control for different stakeholders (admin, user).

2.2.3.5 System Integration and Testing (Month 7)

Unit, integration, and system testing User acceptance testing with stakeholders

Bug fixing and optimization of the system

2.2.3.6 Deployment and Documentation (Month 8)

Implement the system on a selected platform, which may be either a local server.

Technical and User Documentation

2.2.3.7 Final Appraisal and Presentation (Month 9)

Presentation of the system to the evaluation panels of the university and the stakeholder groups.

2.2.4 Versions To Be Established

2.2.4.1 Version 1.0 (Prototype)

1. Basic supply chain workflows.
2. Initial implementation of demand forecasting.
3. Simple chatbot functionality for FAQs.

2.2.4.2 Version 1.5 (Beta)

1. Enhanced chatbot with NLP for contextual responses.
2. Refined forecasting model with performance tuning.
3. User-friendly interface with interactive dashboards.

2.2.4.3 Version 2.0 (Final Release)

1. Fully integrated system with all features.
2. Role-based access and security protocols.
3. Comprehensive reporting and analytics.

2.2.5 Project Deliverables to Be Completed

2.2.5.1 Documentation

1. Requirement Specification Document (Month 1).
2. System Design Document, including architecture diagrams and wireframes (Month 2).
3. Test Plan and Test Cases (Month 7).
4. Final Technical and User Manuals (Month 8).

2.2.5.2 Software Deliverables

1. Source Code Repository (with version control).
2. Executable Application or Deployed System (Month 8).
3. API Documentation for system integration.

2.2.5.3 Presentation Materials

1. Mid-term Project Presentation Slides (Month 5).
2. Final Year Project Presentation Slides and Demonstration Video (Month 9).

2.2.5.4 Reports

1. Progress Reports for Sprints (Monthly).
2. Final Project Report (Month 9)

2.3 Roles and Responsibilities

2.3.1 Project Team Structure:

2.3.1.1 M Irteza Waseem (AI, Backend Developer and Project Leader):

2.3.1.1.1 Responsibilities:

1. Develop and get the AI chatbot integrated into the system, keeping in mind that all functionalities must be aligned with supply chain management requirements.
2. Lead on backend development, and server-side logic.
3. Coordinate with Ubaid Rashid and Hashir Ahmed to have data flow smoothly between frontend, backend, and database.
4. Keep a watchful eye for overall troubleshooting and resolving issues with AI or backend functionality.

2.3.1.2 Ubaid Rashid (Frontend Developer and Backend Developer)

2.3.1.2.1 Responsibilities:

1. Lead the frontend development for web platforms using HTML CSS Java Script and PHP.
2. Work with M Irteza Waseem and Abdullah to make sure that the user interface is visually appealing and user-friendly.
3. Make sure that proper communication is done with the backend so that data is fetched and displayed correctly.
4. Deal with UI-related issues in the project life cycle.

2.3.1.3 Abdullah (SQA and Frontend Developer)

2.3.1.3.1 Tasks:

1. Develop detailed project documentation covering design, implementation, testing, and deployment reports.
2. Help Ubaid Rashid with frontend development tasks when required.
3. Collaborate with Hashir Ahmed to ensure documentation is aligned with the project's overall goals and communicates the technical information involved.
4. Be a contact for all questions regarding project documentation.

2.3.1.4 Hashir Ahmed (Database Developer and Documentation):

2.3.1.4.1 Responsibilities:

1. Develop and maintain a database that will be scalable, secure, and performance-optimized.
2. Collaborate with M Irteza Waseem to ensure that the database will integrate well with the backend and AI chatbot.
3. Help Abdullah in writing project documentation that is detailed.

4. Resolve any database-related problems and ensure data integrity all through the project.

2.3.1.5 Lines of Communication:

1. Communication will flow between team members regularly through daily stand-ups and collaborative sessions.
2. we closely communicate to ensure a unified user experience and efficient project execution.
3. Regular updates on project progress, challenges, and achievements will be shared among the team members.

2.4 Tools and Techniques

2.4.1 Development Methodologies:

2.4.1.1 Agile (Scrum Framework):

Utilize the Agile software development methodology, specifically following the Scrum framework, to accommodate iterative requirements and foster continuous collaboration.

2.4.1.2 Notations:

2.4.1.2.1 UML (Unified Modeling Language):

Employ UML for system modeling, analysis, and design, providing a standardized visual representation of the project's architecture.

2.4.1.2.2 Programming Languages:

2.4.1.2.2.1 Web Application:

Employ Bootstrap, utilizing HTML, CSS, JavaScript, and J Query for web application development, ensuring responsiveness and a user-friendly interface.

2.4.1.2.2.2 PHP For Backend Development:

Utilize PHP for backend development, managing server requests and data handling for the web application.

2.4.1.3 Database Management:

2.4.1.3.1 MySQL (PhpMyAdmin):

Choose MariaDB with PhpMyAdmin for database management, ensuring flexibility, scalability, and ease of integration.

2.4.1.4 Coding Standards:

2.4.1.4.1 Industry Best Practices:

Adhere to industry best practices for coding standards, emphasizing clarity, maintainability, and efficient coding practices.

2.4.1.5 Testing Frameworks:

2.4.1.5.1 Frontend Testing:

1. Usability Testing
2. Functionality Testing
3. Responsiveness Testing

2.4.1.5.2 PHP Unit Testing for Backend:

Implement PHP unit testing for backend components, validating the functionality of server-side code.

2.4.1.6 Collaboration Tools:

2.4.1.6.1 WhatsApp:

Utilize WhatsApp as the primary communication platform for real-time collaboration and team communication. The instant messaging features will facilitate quick updates and discussions among team members.

2.4.1.6.2 Google Meet:

Use Google Meet for virtual meetings and discussions. The video conferencing capabilities will enhance communication, allowing for face-to-face interactions when needed

2.5 Project Management Plan

2.5.1 Tasks

2.5.1.1 Requirements Clarification:

2.5.1.1.1 SPMPT001:

1. Description: Discuss and clarify project requirements with stakeholders.
2. Deliverables/Milestones: Refined project requirements document.
3. Resources Needed: Access to stakeholders and project documentation.
4. Dependencies/Constraints: None.
5. Risks/Contingencies: Potential changes in stakeholder expectations.

2.5.1.2 Exploratory Prototype Development:

2.5.1.2.1 SPMPT002:

1. Description: Develop an exploratory prototype for initial feedback.
2. Deliverables/Milestones: Prototype, user feedback.
3. Resources Needed: Development environment and user testing capabilities.
4. Dependencies/Constraints: Clear understanding of initial requirements.
5. Risks/Contingencies: Limited functionality in the prototype.

2.5.1.3 SRS Writing:

2.5.1.3.1 SPMPT003:

1. Description: Draft and finalize the Software Requirements Specification (SRS).
2. Deliverables/Milestones: Completed SRS document.
3. Resources Needed: Documentation tools, collaboration platforms.
4. Dependencies/Constraints: Clear project scope and requirements.
5. Risks/Contingencies: Changes in requirements during documentation.

2.5.1.4 System Design:

2.5.1.4.1 SPMPT004:

1. Description: Develop the system using Html CSS and Java Script.
2. Deliverables/Milestones: System design documents.
3. Resources Needed: Development environment, design tools.
4. Dependencies/Constraints: Finalized SRS, design standards.
5. Risks/Contingencies: Design complexity leading to delays.

2.5.1.5 SDD Writing:

2.5.1.5.1 SPMPT005:

1. Description: Write the Software Design Description (SDD) based on the system design.
2. Deliverables/Milestones: Completed SDD document.
3. Resources Needed: Documentation tools, design documents.
4. Dependencies/Constraints: Finalized system design.
5. Risks/Contingencies: Changes in design during documentation.

2.5.1.6 Development (Version 1):

2.5.1.6.1 SPMPT006:

1. Description: Develop the first version of the SCMS.
2. Deliverables/Milestones: Initial working version.
3. Resources Needed: Development environment, coding tools.
4. Dependencies/Constraints: Completed SDD, coding standards.
5. Risks/Contingencies: Coding challenges leading to revisions.

2.5.1.7 System Test (Version 1):

2.5.1.7.1 SPMPT007:

1. Description: Test the first version of the system.
2. Deliverables/Milestones: Test results, bug reports.
3. Resources Needed: Testing environment, test cases.
4. Dependencies/Constraints: Completed development.

5. Risks/Contingencies: Identification of critical bugs.

2.5.2 Description

2.5.2.1 Requirements Clarification (SPMPT001):

Description: Collaborative refinement of project requirements through stakeholder discussions. This task ensures a clear understanding of the project's scope and objectives.

2.5.2.2 Exploratory Prototype Development (SPMPT002):

Description: Creation of an initial prototype to gather user feedback and validate conceptual ideas. This task aims to provide a tangible representation of the project's early concepts.

2.5.2.3 SRS Writing (SPMPT003):

Description: Drafting and finalizing the Software Requirements Specification (SRS). This task involves comprehensive documentation of project requirements, guiding the development process.

2.5.2.4 System Design (SPMPT004):

Description: Development of the system design using HTML CSS and JAVA SCRIPT. This task focuses on creating a robust and scalable architecture for the Supply Chain Management System (SCMS).

2.5.2.5 SDD Writing (SPMPT005):

Description: Writing the Software Design Description (SDD) based on the finalized system design. This task involves documenting detailed design specifications for the development team.

2.5.2.6 Development (Version 1) (SPMPT006):

Description: Coding the first version of the SCMS. This task is the implementation phase, translating design specifications into a working software version.

2.5.2.7 System Test (Version 1) (SPMPT007):

Description: Testing the first version of the system to identify functionality issues and report bugs. This task ensures the initial software version meets specified requirements.

2.5.3 Deliverable and Milestone

This section outlines the key project output such as code and documentation and their associated milestones. For instance, Milestone 1 involves submitting the final versions of the SRS, STD, and SPMP; Milestone 2 covers the delivery of the completed STD; and Milestone 3 marks the release of the first tested development version.

2.5.4 Resources Needed

2.5.4.1 Requirements Clarification (SPMPT001):

1. Equipment: Meeting platform, documentation tools.
2. Access: Stakeholder availability, project documentation.

2.5.4.2 Exploratory Prototype Development (SPMPT002):

1. Equipment: Development environment, prototyping tools.
2. Access: User testing capabilities.

2.5.4.3 SRS Writing (SPMPT003):

1. Tools: Documentation platforms, collaboration tools.
2. Access: Project documentation, stakeholder input.

2.5.4.4 System Design (SPMPT004):

1. Tools: HTML CSS and Java Script.
2. Access: Finalized requirements, design standards.

2.5.4.5 SDD Writing (SPMPT005):

1. Tools: Documentation tools, design documents.
2. Access: Finalized system design.

2.5.4.6 Development (Version 1) (SPMPT006):

1. Tools: Development environment, coding tools.
2. Access: Completed SDD, coding standards.

2.5.4.7 System Test (Version 1) (SPMPT007):

1. Tools: Testing environment, test cases.
2. Access: Completed development, test plans.

2.5.5 Dependencies and Constraints

2.5.5.1 Requirements Clarification (SPMPT001):

2.5.5.1.1 Dependencies:

1. Stakeholder availability for discussions.
2. Access to existing project documentation.

2.5.5.1.2 Constraints:

The task initiation is contingent upon the availability of stakeholders for clarification sessions.

2.5.5.2 Exploratory Prototype Development (SPMPT002):

2.5.5.2.1 Dependencies:

1. Clear understanding of initial project requirements.
2. Availability of resources for prototype development.

2.5.5.2.2 Constraints:

The development of the prototype is dependent on having a comprehensive understanding of the initial project requirements.

2.5.5.3 SRS Writing (SPMPT003):

2.5.5.3.1 Dependencies:

1. Finalized project scope and requirements.
2. Collaboration with stakeholders for input.

2.5.5.3.2 Constraints:

SRS writing is contingent upon a clear and agreed upon project scope.

2.5.5.4 System Design (SPMPT004):

2.5.5.4.1 Dependencies:

1. Finalized and clarified requirements.
2. Design standards and guidelines.

2.5.5.4.2 Constraints:

System design initiation is subject to the availability of well-defined project requirements.

2.5.5.5 SDD Writing (SPMPT005):

2.5.5.5.1 Dependencies:

1. Completed system design.
2. Documentation standards and guidelines.

2.5.5.5.2 Constraints:

SDD writing is dependent on the completion of the system design phase.

2.5.5.6 Development (Version 1) (SPMPT006):

2.5.5.6.1 Dependencies:

1. Completed SDD and design standards.
2. Adequate resources for development.

2.5.5.6.2 Constraints:

Development initiation is subject to the availability of a finalized SDD and required resources.

2.5.5.7 System Test (Version 1) (SPMPT007):

2.5.5.7.1 Dependencies:

1. Completed development of Version 1.
2. Test plans and cases.

2.5.5.7.2 Constraints:

System testing relies on the completion of the initial development phase.

2.5.5.8 Website Hosting (SPMPT008):

2.5.5.8.1 Dependencies:

1. Completed website development.
2. Selection of a suitable hosting service.

2.5.5.8.2 Constraints:

Hosting setup is contingent on the completion of website development and the availability of a chosen hosting service.

2.6 Assignments

2.6.1 Requirements Clarification (SPMPT001):

Assignment: M. Irteza Waseem (Backend Developer, AI, and Project Leader), Ubaid Rashid (Backend Developer, Frontend Developer), Hashir Ahmed (Documentation and Database Developer) Abdullah (SQA and Frontend Developer)

2.6.2 Exploratory Prototype Development (SPMPT002):

Assignment: Ubaid Rashid, Abdullah

2.6.3 SRS Writing (SPMPT003):

Assignment: Hashir Ahmed

2.6.4 System Design (SPMPT004):

Assignment: M Irteza Waseem (Backend Developer and AI)

2.6.5 SDD Writing (SPMPT005):

Assignment: Hashir Ahmed, Abdullah

2.6.6 Development (Version 1) (SPMPT006):

Assignment: Ubaid Rashid, Abdullah

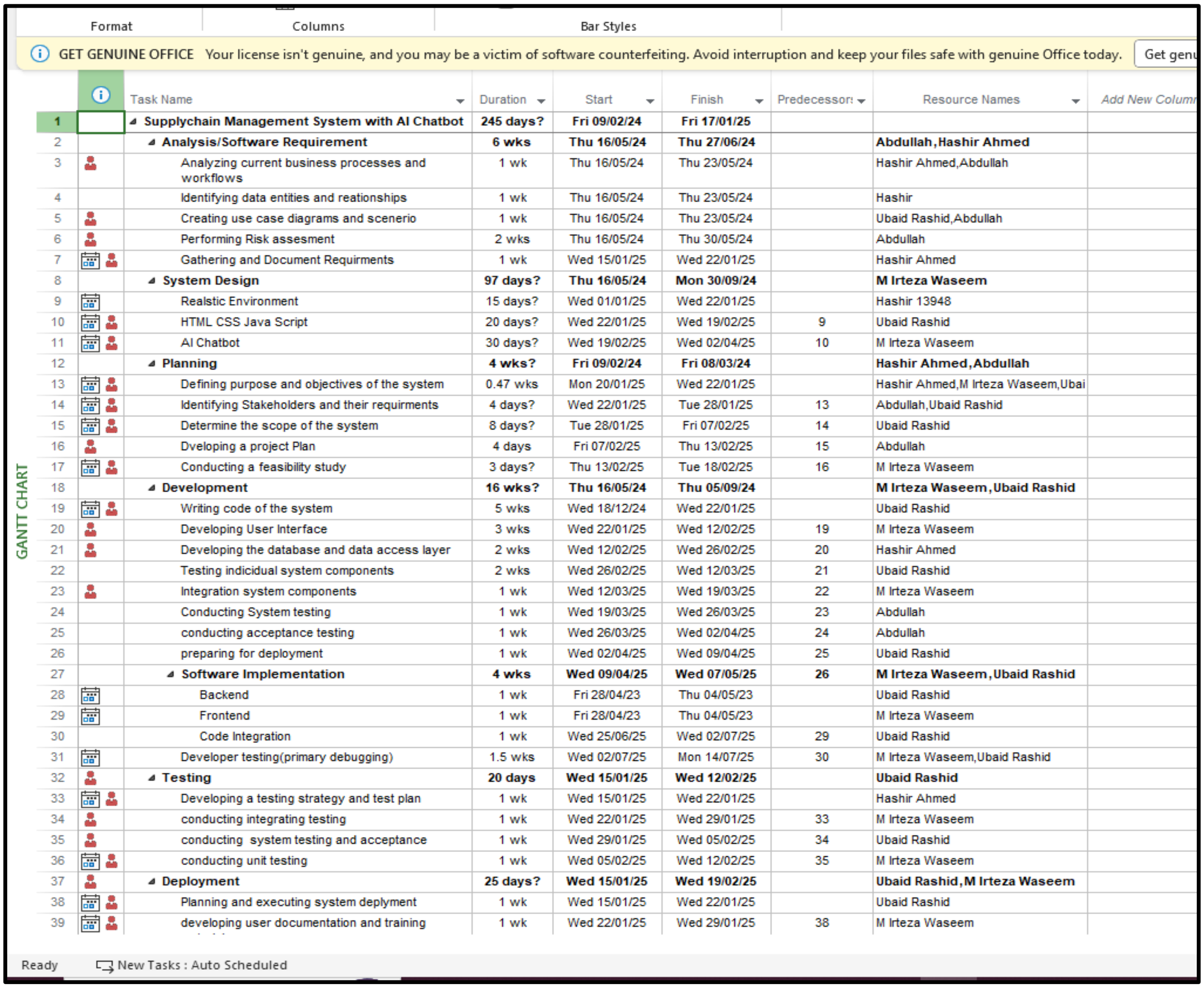
2.6.7 System Test (Version 1) (SPMPT007):

Assignment: Hashir Ahmed and Abdullah

2.6.8 Website Hosting (SPMPT008):

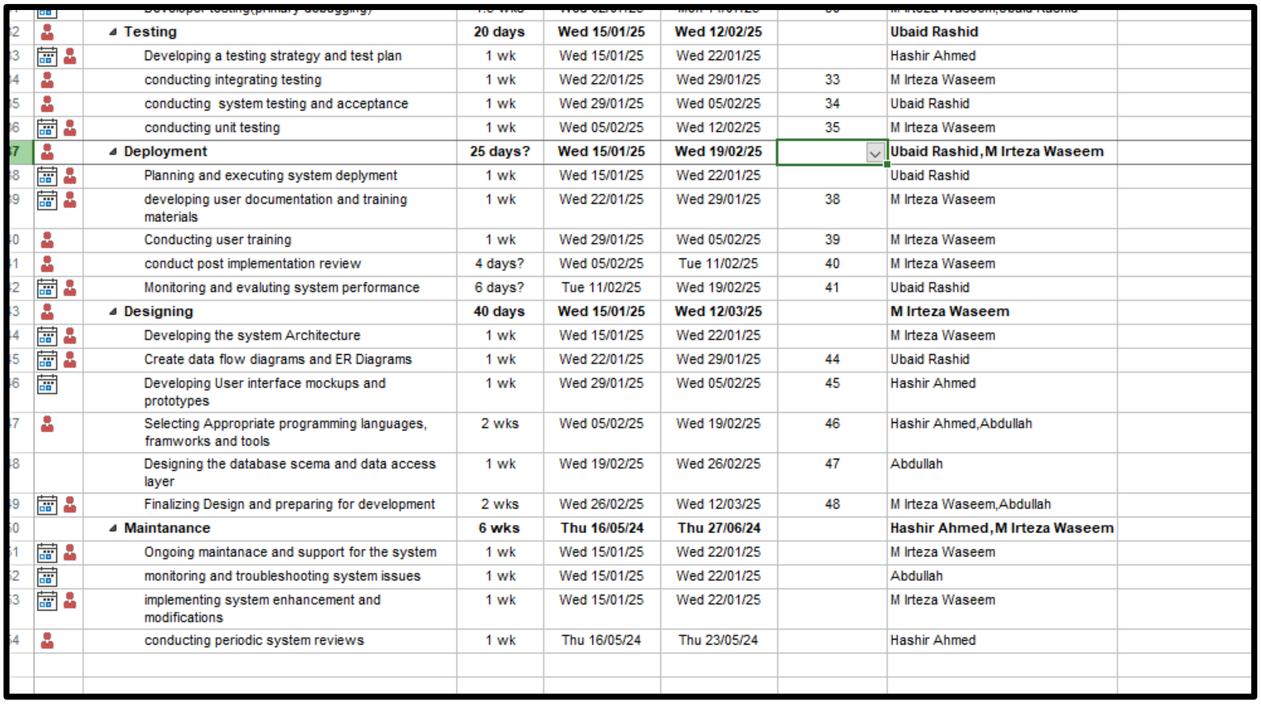
Assignment: Ubaid Rashid, M Irteza Waseem

2.7 Timetable

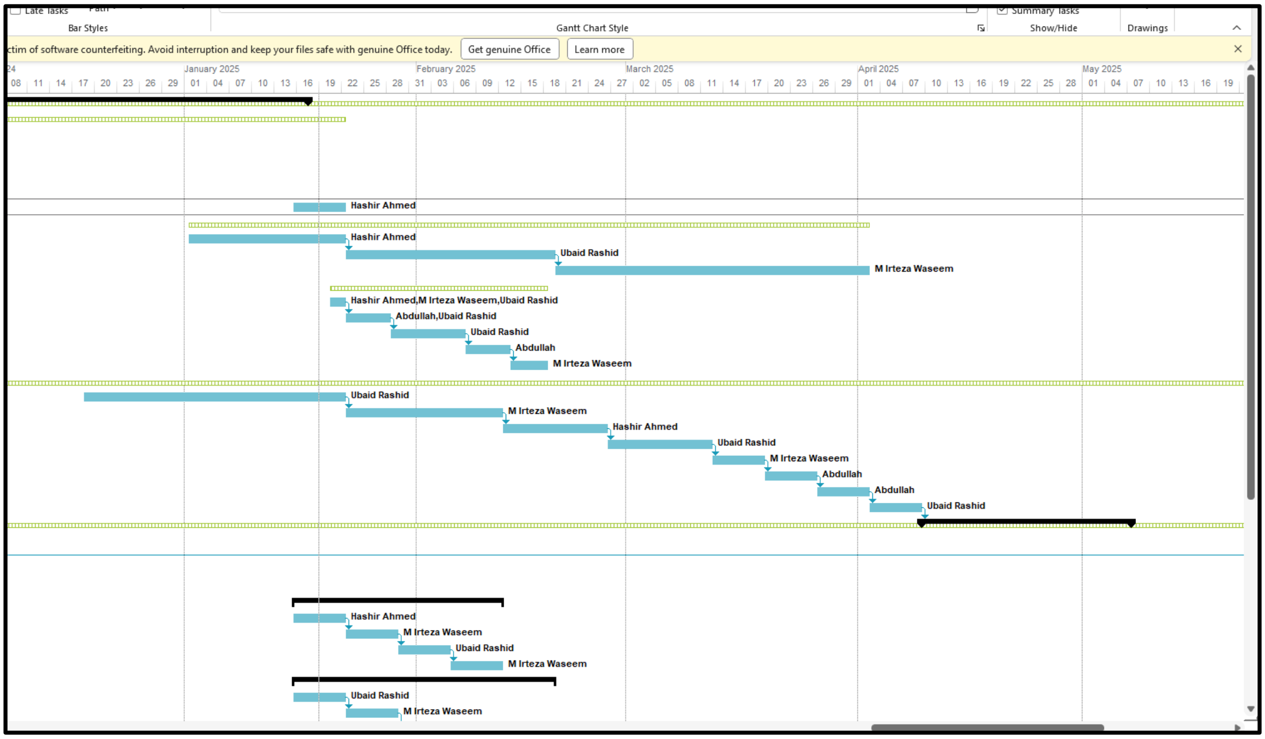

Format
Columns
Bar Styles
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today.
Get genu
GANTT CHART
Ready
New Tasks : Auto Scheduled

	ⓘ	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Add New Colum
1		**Supplychain Management System with AI Chatbot**	**245 days?**	**Fri 09/02/24**	**Fri 17/01/25**			
2		**Analysis/Software Requirement**	**6 wks**	**Thu 16/05/24**	**Thu 27/06/24**		**Abdullah,Hashir Ahmed**	
3	👤	Analyzing current business processes and workflows	1 wk	Thu 16/05/24	Thu 23/05/24		Hashir Ahmed,Abdullah	
4		Identifying data entities and reationships	1 wk	Thu 16/05/24	Thu 23/05/24		Hashir	
5	👤	Creating use case diagrams and scenerio	1 wk	Thu 16/05/24	Thu 23/05/24		Ubaid Rashid,Abdullah	
6	👤	Performing Risk assesment	2 wks	Thu 16/05/24	Thu 30/05/24		Abdullah	
7	📅 👤	Gathering and Document Requirments	1 wk	Wed 15/01/25	Wed 22/01/25		Hashir Ahmed	
8		**System Design**	**97 days?**	**Thu 16/05/24**	**Mon 30/09/24**		**M Irteza Waseem**	
9	📅	Realstic Environment	15 days?	Wed 01/01/25	Wed 22/01/25		Hashir 13948	
10	📅 👤	HTML CSS Java Script	20 days?	Wed 22/01/25	Wed 19/02/25	9	Ubaid Rashid	
11	📅 👤	AI Chatbot	30 days?	Wed 19/02/25	Wed 02/04/25	10	M Irteza Waseem	
12		**Planning**	**4 wks?**	**Fri 09/02/24**	**Fri 08/03/24**		**Hashir Ahmed,Abdullah**	
13	📅 👤	Defining purpose and objectives of the system	0.47 wks	Mon 20/01/25	Wed 22/01/25		Hashir Ahmed,M Irteza Waseem,Ubai	
14	📅 👤	Identifying Stakeholders and their requirments	4 days?	Wed 22/01/25	Tue 28/01/25	13	Abdullah,Ubaid Rashid	
15	📅 👤	Determine the scope of the system	8 days?	Tue 28/01/25	Fri 07/02/25	14	Ubaid Rashid	
16	👤	Dveloping a project Plan	4 days	Fri 07/02/25	Thu 13/02/25	15	Abdullah	
17	📅 👤	Conducting a feasibility study	3 days?	Thu 13/02/25	Tue 18/02/25	16	M Irteza Waseem	
18		**Development**	**16 wks?**	**Thu 16/05/24**	**Thu 05/09/24**		**M Irteza Waseem,Ubaid Rashid**	
19	📅 👤	Writing code of the system	5 wks	Wed 18/12/24	Wed 22/01/25		Ubaid Rashid	
20	👤	Developing User Interface	3 wks	Wed 22/01/25	Wed 12/02/25	19	M Irteza Waseem	
21	👤	Developing the database and data access layer	2 wks	Wed 12/02/25	Wed 26/02/25	20	Hashir Ahmed	
22		Testing indicidual system components	2 wks	Wed 26/02/25	Wed 12/03/25	21	Ubaid Rashid	
23	👤	Integration system components	1 wk	Wed 12/03/25	Wed 19/03/25	22	M Irteza Waseem	
24		Conducting System testing	1 wk	Wed 19/03/25	Wed 26/03/25	23	Abdullah	
25		conducting acceptance testing	1 wk	Wed 26/03/25	Wed 02/04/25	24	Abdullah	
26		preparing for deployment	1 wk	Wed 02/04/25	Wed 09/04/25	25	Ubaid Rashid	
27		**Software Implementation**	**4 wks**	**Wed 09/04/25**	**Wed 07/05/25**	**26**	**M Irteza Waseem,Ubaid Rashid**	
28	📅	Backend	1 wk	Fri 28/04/23	Thu 04/05/23		Ubaid Rashid	
29	📅	Frontend	1 wk	Fri 28/04/23	Thu 04/05/23		M Irteza Waseem	
30		Code Integration	1 wk	Wed 25/06/25	Wed 02/07/25	29	Ubaid Rashid	
31	📅	Developer testing(primary debugging)	1.5 wks	Wed 02/07/25	Mon 14/07/25	30	M Irteza Waseem,Ubaid Rashid	
32	👤	**Testing**	**20 days**	**Wed 15/01/25**	**Wed 12/02/25**		**Ubaid Rashid**	
33	📅 👤	Developing a testing strategy and test plan	1 wk	Wed 15/01/25	Wed 22/01/25		Hashir Ahmed	
34	👤	conducting integrating testing	1 wk	Wed 22/01/25	Wed 29/01/25	33	M Irteza Waseem	
35	👤	conducting system testing and acceptance	1 wk	Wed 29/01/25	Wed 05/02/25	34	Ubaid Rashid	
36	📅 👤	conducting unit testing	1 wk	Wed 05/02/25	Wed 12/02/25	35	M Irteza Waseem	
37	👤	**Deployment**	**25 days?**	**Wed 15/01/25**	**Wed 19/02/25**		**Ubaid Rashid,M Irteza Waseem**	
38	📅 👤	Planning and executing system deplyment	1 wk	Wed 15/01/25	Wed 22/01/25		Ubaid Rashid	
39	📅 👤	developing user documentation and training	1 wk	Wed 22/01/25	Wed 29/01/25	38	M Irteza Waseem	

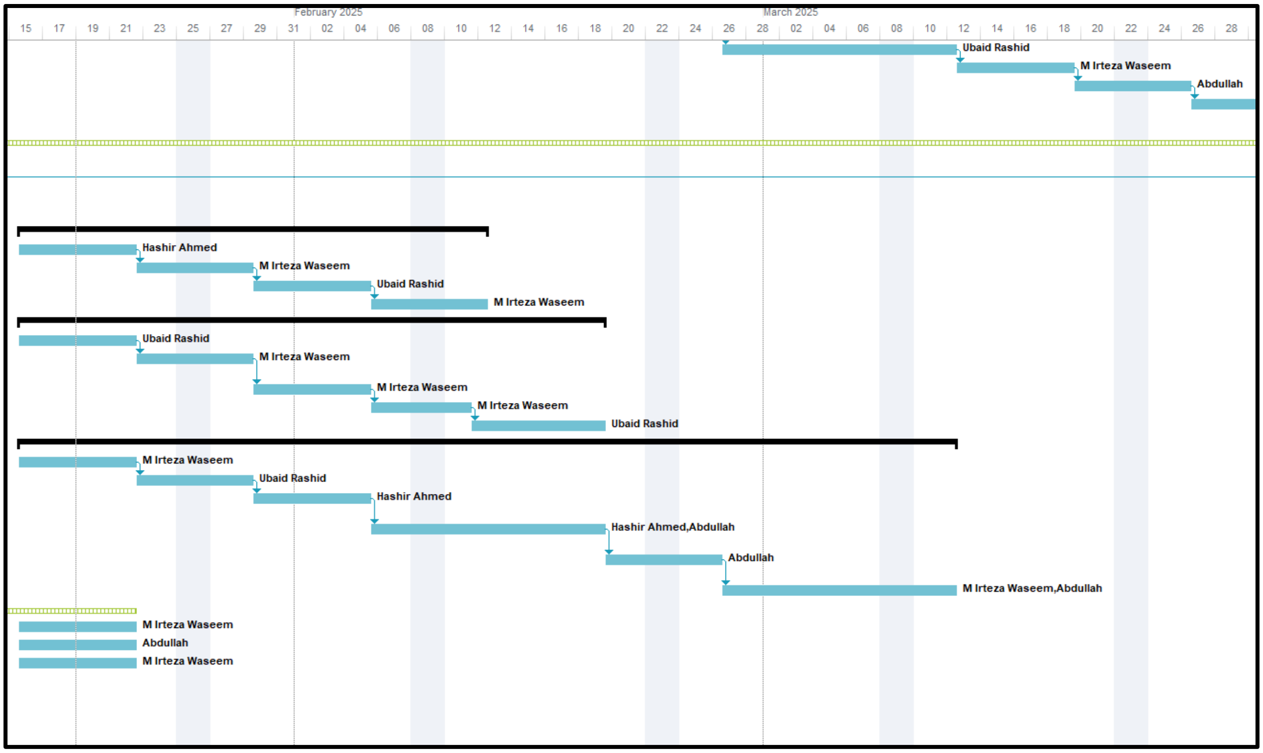
Figure 2 Time Table

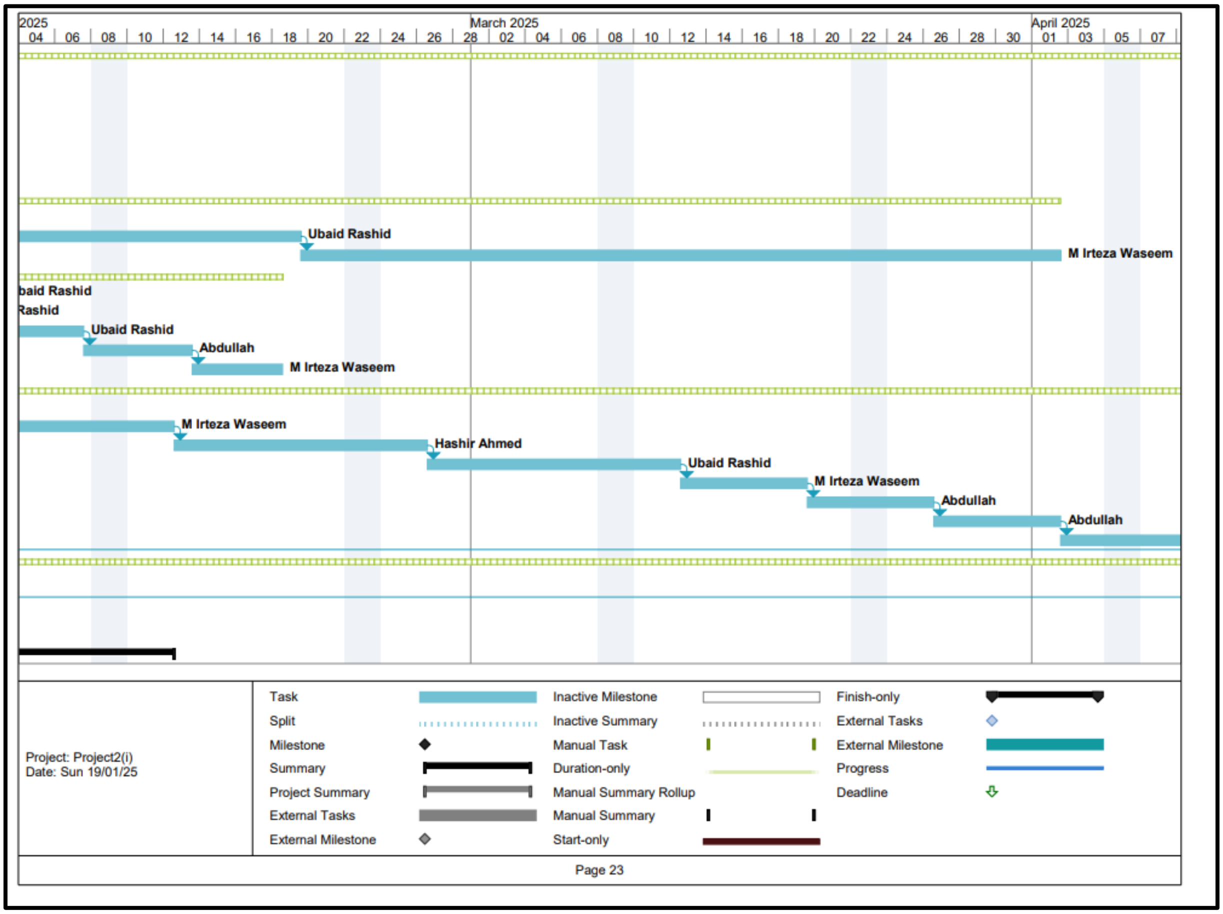


2	**Testing**	**20 days**	**Wed 15/01/25**	**Wed 12/02/25**		**Ubaid Rashid**
3	Developing a testing strategy and test plan	1 wk	Wed 15/01/25	Wed 22/01/25		Hashir Ahmed
4	conducting integrating testing	1 wk	Wed 22/01/25	Wed 29/01/25	33	M Irteza Waseem
5	conducting system testing and acceptance	1 wk	Wed 29/01/25	Wed 05/02/25	34	Ubaid Rashid
6	conducting unit testing	1 wk	Wed 05/02/25	Wed 12/02/25	35	M Irteza Waseem
7	**Deployment**	**25 days?**	**Wed 15/01/25**	**Wed 19/02/25**		**Ubaid Rashid,M Irteza Waseem**
8	Planning and executing system deplyment	1 wk	Wed 15/01/25	Wed 22/01/25		Ubaid Rashid
9	developing user documentation and training materials	1 wk	Wed 22/01/25	Wed 29/01/25	38	M Irteza Waseem
0	Conducting user training	1 wk	Wed 29/01/25	Wed 05/02/25	39	M Irteza Waseem
1	conduct post implementation review	4 days?	Wed 05/02/25	Tue 11/02/25	40	M Irteza Waseem
2	Monitoring and evaluting system performance	6 days?	Tue 11/02/25	Wed 19/02/25	41	Ubaid Rashid
3	**Designing**	**40 days**	**Wed 15/01/25**	**Wed 12/03/25**		**M Irteza Waseem**
4	Developing the system Architecture	1 wk	Wed 15/01/25	Wed 22/01/25		M Irteza Waseem
5	Create data flow diagrams and ER Diagrams	1 wk	Wed 22/01/25	Wed 29/01/25	44	Ubaid Rashid
6	Developing User interface mockups and prototypes	1 wk	Wed 29/01/25	Wed 05/02/25	45	Hashir Ahmed
7	Selecting Appropriate programming languages, framworks and tools	2 wks	Wed 05/02/25	Wed 19/02/25	46	Hashir Ahmed,Abdullah
8	Designing the database scema and data access layer	1 wk	Wed 19/02/25	Wed 26/02/25	47	Abdullah
9	Finalizing Design and preparing for development	2 wks	Wed 26/02/25	Wed 12/03/25	48	M Irteza Waseem,Abdullah
0	**Maintanance**	**6 wks**	**Thu 16/05/24**	**Thu 27/06/24**		**Hashir Ahmed,M Irteza Waseem**
1	Ongoing maintanace and support for the system	1 wk	Wed 15/01/25	Wed 22/01/25		M Irteza Waseem
2	monitoring and troubleshooting system issues	1 wk	Wed 15/01/25	Wed 22/01/25		Abdullah
3	implementing system enhancement and modifications	1 wk	Wed 15/01/25	Wed 22/01/25		M Irteza Waseem
4	conducting periodic system reviews	1 wk	Thu 16/05/24	Thu 23/05/24		Hashir Ahmed



ctim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today. Get genuine Office Learn more


February 2025
March 2025
15 17 19 21 23 25 27 29 31 02 04 06 08 10 12 14 16 18 20 22 24 26 28 02 04 06 08 10 12 14 16 18 20 22 24 26 28
Ubaid Rashid
M Irteza Waseem
Abdullah
Hashir Ahmed
M Irteza Waseem
Ubaid Rashid
M Irteza Waseem
Ubaid Rashid
M Irteza Waseem
M Irteza Waseem
M Irteza Waseem
Ubaid Rashid
M Irteza Waseem
Ubaid Rashid
Hashir Ahmed
Hashir Ahmed,Abdullah
Abdullah
M Irteza Waseem,Abdullah
M Irteza Waseem
Abdullah
M Irteza Waseem


2025
March 2025
April 2025
04 06 08 10 12 14 16 18 20 22 24 26 28 02 04 06 08 10 12 14 16 18 20 22 24 26 28 30 01 03 05 07
Ubaid Rashid
M Irteza Waseem
baid Rashid
Rashid
Ubaid Rashid
Abdullah
M Irteza Waseem
M Irteza Waseem
Hashir Ahmed
Ubaid Rashid
M Irteza Waseem
Abdullah
Abdullah
Project: Project2(i)
Date: Sun 19/01/25
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Progress
Deadline
Page 23

2.7.1 Figure 2 Timetable Explanation:

The project management overview details a software development initiative, outlining tasks like requirements analysis, system design, testing, deployment, and maintenance. Each task is assigned to team members, clearly indicating responsibilities.

The Gantt charts visually represent the project timeline, showing task start and end dates, as well as overlaps. This helps the team stay organized and manage resources effectively. Key milestones mark important stages, while dependencies clarify which tasks need completion before others can begin.

Overall, this documentation functions as a roadmap for the team, enhancing collaboration and ensuring smooth project progression.

3. SOFTWARE REQUIREMENTS SPECIFICATIONS

3.1 Introduction

Supply chain management provides the means of competitiveness and customer satisfaction. It is more interested in coordination of procurement, production, inventory, and distribution for gaining efficiency and reducing costs.

1. Automation of manual process.
2. The facility of real-time visibility to get data of all the functions.
3. Artificial Intelligence is the primary factor for analysis.
4. Data security and reliability is another major factor.
5. Flexibility according to business growth.

3.2 Product Overview

In the current globalized business environment, effective Supply Chain Management is important in competitiveness and satisfaction of customers. It encompasses coordinating procurement, production, inventory, and distribution to boost efficiency and cut costs. Yet, challenges persist due to manual processes, fragmented systems, and limited real-time visibility.

3.3 Specific Requirements

3.3.1 Functional Requirements

Functional requirements are the definition of specific behaviors and functionalities that must be carried out by the Supply Chain Management System. Each requirement must be testable and observable for users or other systems.

3.3.2 User Management:

1. The system will allow administrators to create, edit, and delete user accounts, assigning specific roles such as Supplier, Distributor, or Retailer to manage access and responsibilities effectively.
2. The system shall authenticate users by using their credentials to access the particular functionalities.

3.3.3 Inventory Management:

1. The system will monitor real-time stock levels across multiple warehouses, ensuring up-to-date and accurate inventory tracking
2. Users will receive automatic alerts whenever inventory levels drop below a set minimum threshold, helping prevent stockouts.
3. The system will allow users to easily view and update product details, including the product name, SKU, and available quantity, for efficient inventory management.

3.3.4 Order Management:

1. The system shall enable retailer to create and manage purchase orders.
2. The system shall enable customers to place orders.
3. The system will inform the user on updates regarding their orders (i.e., order placed).

3.3.5 Non-Functional Requirement

These define the quality attributes of the system such as performance, security, and usability.

3.3.5.1 Performance

1. The system shall handle up to 10,000 concurrent users without degrading in performance.
2. The average time taken by the system to process data queries and generate the results is within 2 seconds.

3.3.5.2 Scalability

The system shall have the ability to scale with an increase in the number of users, warehouses, and transactions.

3.3.5.3 Security

1. The system shall store the passwords of the users using one-way hashing algorithm.
2. The system shall have a multi-factor authentication mechanism for the administrative accounts.

3.3.5.4 Usability

1. The system shall be user-friendly with an interface accessible on web and mobile devices.
2. The system shall offer tooltips and an exhaustive user guide.

3.3.5.5 Reliability

1. The system shall have an uptime of at least 99.9%.
2. The system shall have automated data backup and recovery mechanisms.

3.3.5.6 Maintainability

1. The system codebase shall adhere to industry-standard coding practices for easy updates and maintenance.
2. The system shall provide logs in detail for debugging and error tracking.

3.3.5.7 Compliance

The system shall be in compliance with all relevant regulatory standards, including quality management with ISO 9001 and data protection with GDPR.

3.3.4 User Interfaces:

3.3.4.1 Administrator Interface

Provide administrators with all the tools required to supervise an effective supply chain; track inventory easily; manage suppliers from a database perspective; and measure performances. Provide actionable insight into bottlenecks and key performance indicators.

3.3.4.2 Retailer Interface:

Tightly interacting user-friendly application for retailers so an intuitive interface would result while processing their orders, managing inventories, and responding to procurement requests.

3.3.4.3 Customer Interface:

Offer a seamless user interaction process through, and on-hand inventory levels. Add AI based chat services for improved interaction with customers.

3.3.4.4 Key Functionality

1. Forecasting for Prediction in Sales Implement forecasting algorithms to predict future sales.
2. Real-Time Inventory Automation Monitor stock availability driven from real-time data.
3. Order Management Automation Automate order processing from suppliers to customers.
4. Supplier and Vendor Collaboration Facilitate seamless communication between suppliers, manufacturers, and customers
5. AI Chatbot Solve customer and seller queries.

3.4 Non-Functional Requirements

3.4.1 Reliability

1. The system will maintain a minimum uptime of 99.9% to ensure high availability and reliability.
2. It will include automated data backup and recovery features to protect against data loss and ensure business continuity.

3.4.2 Availability

3.4.2.1 Overall System Availability

3.4.2.1.1 Depiction

The SCMS will be designed for 24/7 availability. It will leverage MYSQL infrastructure, redundancy, and failover mechanisms to ensure that downtime is minimized and that all users can access the system at all times.

3.4.2.1.2 Accessibility Target

1. Uptime: 99.9% annual uptime with scheduled maintenance during off-peak hours.
2. Concurrency: 10,000 concurrent users.
3. Platform Accessibility: Accessible through web and mobile on all major browsers and devices.

3.4.2.1.3 Capabilities

1. Redundancy: Load balancing and redundant servers to prevent service disruptions.
2. Disaster Recovery: Daily backup with a 2-hour RTO and 15-minute RPO.
3. Monitoring: Real-time system health monitoring with automated alerts.
4. Fault Tolerance: Modular design helps contain and isolate faults.
5. Maintenance: Updates with minimal impact and prior notification.

3.4.3 Security

This section outlines the controls implemented to prevent unauthorized access, misuse, modification, destruction, or disclosure of the SCMS. Such controls ensure confidentiality, integrity, and availability of system data and resources.

3.4.3.1 Logging

Validation password and user name

3.4.3.2 Security Measures

Implement multi-factor authentication to protect sensitive data.

3.4.3.3 Data Integrity

1. Check-sum validation and frequent integrity checks are performed on key data.
2. Others Trigger alerts and lockouts on repetitive failed login.

3.4.4 Maintainability

The SCMS is designed to be maintainable with a modular architecture and encapsulated components that can be easily updated, standardized with proper documentation, and readable, low-complexity code that follows best practices. Version control is managed with Git, ensuring backward compatibility for two previous versions. Automated testing covers 80% of the codebase, supported by a staging environment for thorough pre-deployment validation. Detailed logging aids in real-time monitoring while providing comprehensive documentation for developers and administrators to help them maintain and enhance the system.

3.4.5 Portability

The SCMS is designed to be portable so that it is easy to deploy across different environments:

3.4.5.1 Host Independence

The system minimizes host-dependent code, with less than 5% of the code tailored to specific platforms.

3.4.5.2 Portable Language

The application developed is on some proven portable language, like python or PHP, Java Script, HTML, CSS and so can operate on many devices.

3.4.5.3 Compiler and Framework:

Standard compilers Like VS CODE, Visual Studio Code, Code blocker etc. and frameworks, such as PHP, Java Script, HTML, CSS are used to enhance portability.

3.4.5.4 Operating System Compatibility:

The system is designed to run on the major operating systems, which include Windows,
adjustments are automatically applied during deployment.

3.4.5.5 Containerization:

The XGBoost machine learning model and backend components (developed in PHP) were deployed in separate containers, making the system modular and easy to manage.

3.4.6 Performance

The system shall be in compliance with all relevant regulatory standards, including quality management with ISO 9001 and data protection with GDPR:

3.4.6.1 Static Numerical Requirements

3.4.6.1.1 Simultaneous Users:

The system shall support at least maximum users without performance degradation.

3.4.6.1.2 Data Storage:

The system shall accommodate at least 8GB of data storage, scalable on business growth.

3.4.6.1.3 Network Bandwidth:

The system shall provide a minimum network bandwidth of 1 Gbps to achieve optimal performance in high traffic environments.

3.4.6.2 Dynamic Numerical Requirements

3.4.6.2.1 Transaction Throughput

1. Average Load: The system shall process 50 TPS at startup.
2. Peak Load: The system shall process 1,000 TPS during peak load hours.

3.4.6.2.2 Response Time

1. For average loads, the system shall respond to users' requests within 2 seconds.
2. For peak load, the system shall not exceed 5 seconds for response time.

3.4.6.2.3 Data Processing:

The system shall perform batch operations for up to half million within 30 minutes.

3.4.6.2.4 Uptime:

The system shall achieve an uptime of 90% that has minimal downtimes due to maintenance.

3.4.7 Usability Requirements

The SCMS shall be designed to accommodate all types of users (admin, retailer, manufacturer) to offer a user-friendly experience. The following are the usability requirements:

3.4.7.1 User Interface (UI)

1. The system shall provide a clean, intuitive, and responsive UI, which enables easy navigation to core functionalities and access to all functionalities.
2. The system shall be able to present customizable dashboards for the view of users that is quick information viewing, for instance, on levels of inventory or order statuses.

3.4.7.2 Accessibility

The system should be fully operational in all the key browsers, for example, Chrome, Firefox, Safari, Edge.

3.4.7.3 Onboarding and Training

1. Ai-chatbot Help a user for using a application and solve their queries.
2. In-app tooltips and help documentation should be accessible to the user to guide the user on navigating the system

3.4.7.4 Response Time

The system's user interface shall react to user input (e.g., clicks, form submissions) within 2 seconds under normal load conditions.

3.4.7.5 Error Handling and Feedback

1. The system shall clearly provide actionable error messages to lead users to resolution.
2. Users shall receive immediate feedback for critical actions (e.g., order placed, inventory updated).

3.5 Proposed Solution

The proposed solution for the Supply Chain Management System (SCMS) aims to address key challenges in the supply chain by providing a centralized platform that streamlines processes such as inventory management, order tracking, supplier collaboration, and reporting. The solution will consist of the following components:

3.5.1 Centralized Dashboard:

A user-friendly interface that allows real-time tracking of inventory, orders, and shipment giving users a clear and complete view of the entire supply chain at a glance.

3.5.2 Modular Architecture:

The system will be designed using a modular approach to allow flexibility, scalability, and easy integration with existing enterprise systems (e.g., ERP, third-party logistics providers).

3.5.3 Data Integrity & Security:

Multifactor authentication and validation data is use for security purpose.

3.5.4 Performance Optimization:

The system will be capable of handling 50 concurrent users and 10 transactions per second during peak load periods. Response time for user requests will be kept under 2 seconds.

3.5.5 Portability & Scalability:

Built using portable languages such as HTML, CSS, JavaScript, or Python, the system is designed to be compatible with multiple operating systems, including Windows.

3.5.6 Automation and Reporting:

Automated notifications, reporting, and analytics will facilitate decision-making and ensure transparency across the supply chain.

3.5.7 User-Centric Design:

The system will provide a simple and intuitive UI, compliance with accessibility standards, and robust user support features like in-app tutorials and help documentation.

3.6 Alternative Solution

Instead of building a custom system, businesses can opt for third-party supply chain solutions like COTS or SaaS platforms. These offer quick setup, built-in features, and vendor support. However, they may lack flexibility, pose integration challenges, and lead to higher long-term costs. Companies may also need to adjust their processes to fit the software.

3.6.1 Use of Third-Party Software:

SAP Ariba, Oracle SCM Cloud, or even Microsoft Dynamics 365 can be used to streamline supply chain activity. These contain pre-built options for inventory and order tracking plus supplier relationships that can be available. These shall be cloud-native solutions, taking away the pressure of infrastructure handling, with performance and scalability entirely taken care of by the service provider.

3.6.2 Out-of-the-Box Functionality:

Some of the existing features are fully ready to be used, resulting in less time spent on development with quick deployment capability, but customized options may have limited scope.

3.6.3 Security and Compliance:

Built-in security features plus compliance with such industry standards such as GDPR or ISO 27001 would exist, thus potentially saving the customers from custom-based security configurations.

3.6.4 Cost Considerations:

Third-party software is often less expensive to develop, but the licensing fees, subscription models, and long-term support can be expensive compared to a custom-built solution.

3.6.5 Vendor Lock-In:

Third-party platforms could lead to vendor lock-in, limiting flexibility for future changes and integrations.

3.7 External Interface Requirements

3.7.1 User Interfaces

3.7.1.1 User Interface Characteristics

3.7.1.1.1 Screen Formats:

The system must be responsive design to be compatible with desktops.

3.7.1.1.2 Page/Window Layouts

Consistent layout with a navigation bar on the left and content displayed in sections for easy access to inventory, orders, and reports.

3.7.1.1.3 Menus

Clear and hierarchical menus for easy access to system modules (e.g., Orders, Inventory, Reports).

3.7.1.1.4 Reports and Dashboards:

1. Customizable, user-friendly reports with key metrics (e.g., inventory levels, order status).
2. Dashboards should display graphical charts for easy visualization of data.

3.7.1.1.5 Graph:

Graph is used for product priority and demanding through forecasting prediction model.

3.7.1.2 Optimizing User Interaction

3.7.1.2.1 Intuitive Navigation:

Simple navigation with clearly labeled buttons and icons for key actions.

3.7.1.2.2 Minimalistic Design:

Avoid clutter and focus on the most essential information with clear action buttons.

3.7.1.2.3 Consistency:

Apply consistent color schemes, fonts, and icons to increase recognition and make it easy to use.

3.7.1.2.4 Error Prevention

Implement input validation, clear instructions, and tooltips to assist the user in performing the desired task and avoiding errors.

3.7.1.2.5 Accessibility:

The system will adhere to WCAG 2.1 standards in support of users with disabilities (screen reader support, high contrast mode, keyboard navigation).

3.7.1.2.6 Help and Support:

In-app tooltips, user manuals, and FAQs will be available to guide users in navigating and troubleshooting the application.

3.7.2 Hardware Interfaces

3.7.2.1 PC or Laptop

1. Model: Intel core 2duo or later.
2. Operating System: Windows 10 or later.

3.7.3 Software Interfaces

3.7.3.1 Web Application:

1. Platform: Windows 10.
2. Database Management System: MySQL 8.0 PostgreSQL 13.
3. Web Server: Xampp Apache 2.4

3.7.4 Communications Protocols

3.7.4.1 Information Type Used by Other Functions:

3.7.4.1.1 Customer Information:

1. Data Type: Name, contact information, shipping address.
2. Usage: This information is used for customer registration, order processing, communication, and managing user-specific details in the Supply Chain Management System (SCMS).

3.7.4.1.2 Product Information

1. Data Type: Product name, SKU, category, quantity, price, description.
2. Usage: Inventory management, product catalog, and order fulfillment.
3. This information provides the number of products, inventory levels, and updates.

3.7.4.1.3 Order Information

1. Data Type: Order ID, customer information, products ordered, status of order, payment status.
2. Usage: For creating and managing customer orders within the SCMS. The order information is necessary for fulfilling the orders, shipping, and sending the invoices.

3.7.4.1.4 Retailer Information

1. Data Type: Retailer name, contact information, products supplied, pricing, delivery times.
2. Usage: This information is applied to monitor the supplier relationships, procurement, and supplier performance in the system.

3.7.4.1.5 Manufacture Information

1. Data Type: Manufacture name, contact information, products supplied, pricing, delivery times.
2. Usage: This information is applied to monitor the supplier relationships, procurement, and supplier performance in the system.

3.7.4.1.6 Inventory Data

1. Data Type: Inventory level, stock turn, warehouse locations, reorder point.
2. Usage: This information is applied to monitor the stock level, manage supply chain logistics, and trigger the reordering process to avoid stockouts.

3.7.4.1.7 User Authentication

1. Data Type: User credentials, which include username, password, authentication tokens.
2. Usage: It ensures secure access for authorized users such as system administrators, supply chain managers and others to the SCMS while authenticating the login credentials.

3.8 Database Requirements

3.8.1 User Information

1. User ID, name, email, and contact details.
2. Role, such as supplier, retailer, admin.
3. Login details, and stored securely.

3.8.2 Product Information

1. Product ID, name, category, and description.
2. Quantity available.
3. Unit of measurement, e.g., kg, liters, units.
4. Storage conditions, e.g., temperature requirements, if any.

3.8.3 Inventory Data

1. Stock at different locations for each product.
2. Incoming and outgoing stock records.
3. Threshold alerts on low inventory.

3.8.4 Manufacturer Information

1. Manufacturer ID, name, and contact details.
2. Products supplied.
3. Performance metrics, such as delivery timeliness, product quality.

3.8.5 Retailer Information

1. Retailer ID, name and contact details
2. Order preferences and history
3. Delivery schedules

3.8.6 Orders

1. Order ID, date, and status pending, completed, canceled, etc.
2. Products ordered along with their quantity
3. Delivery time, location, and assigned agents

3.8.7 Chatbot Data

1. Chat history logs
2. Frequently asked questions (FAQs)
3. User queries and answers
4. Training data for the chatbot's improvement

3.8.8 Analytics and Reports

1. Inventory reports: stock movement, product trends, etc.
2. Order fulfillment metrics
3. Supplier and retailer activity logs
4. Predictive analytics data on demand.

4. SOFTWARE DESIGN DESCRIPTION

4.1 Introduction

The SDD for the "Supply Chain Management System with AI Chatbot" project is a very comprehensive guide outlining the purpose, scope, and essential features of the AI-powered supply chain system. It is a very important reference for developers, architects, and stakeholders in terms of insights into the architecture, design considerations, and a roadmap for the software development life cycle.

4.2 Design Overview

The design for the Supply Chain Management System with AI Chatbot is structured to make sure it becomes scalable and efficient; it is highly interactive to all users. It has used modern technologies, along with the power of automation through AI. Below are some of the must-have features for the design so that it satisfies the requirements above:

4.2.1 Modular Architecture

The system uses modular architecture to create flexibility and ease of maintenance. Key components involve:

4.2.1.1 AI Chatbot Module:

It is used for the usage of application and provide solution to the queries.

4.2.1.2 Inventory Management Module:

Stock levels update in real time.

4.2.1.3 Order Processing Module:

Handles customer orders and tracks their fulfillment status.

4.2.1.4 Retailer Notification Module:

Automates the restocking alert and manages the Retailer’s interactions.

4.2.1.5 Prediction Model Module:

The system leverages machine learning algorithms to accurately forecast demand, optimize inventory levels, and predict delivery timelines for more efficient supply chain operations.

4.2.2 Prediction Model Integration

The system uses a simple predictive model to aid in decision-making:

4.2.2.1 Demand Forecasting:

It predicts future demand patterns using historical order data and trends, thus maintaining optimal inventory levels.

4.2.2.2 Restocking Optimization

Anticipate stock replenishment needs to reduce lead times and cut holding costs.

4.2.2.3 Delivery Time Forecasting:

Determine the delivery times based on order volume, location, and performance of logistics.

4.2.3 AI Chatbot (Basic Version)

A rule-based AI chatbot is intended to:

1. Answer FAQs and give live responses on the status of an order and the delivery status.
2. Facilitate order placement, issue resolution, and basic queries from customers and suppliers.

4.2.4 User-Friendly Interfaces

Simple and intuitive interfaces are designed for each stakeholder:

4.2.4.1 Customers:

Interface based on a chatbot to manage orders, tracking, and issue resolution.

4.2.4.2 Retailer's:

Dashboard for restocking with notifications.

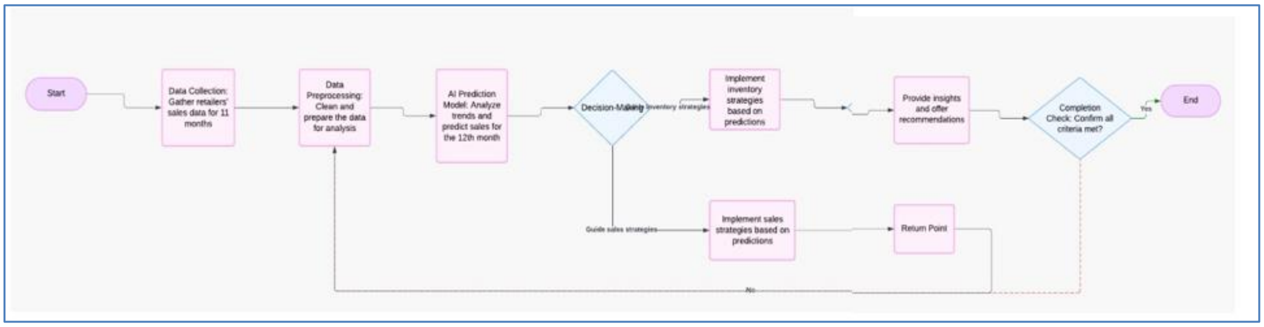
4.2.4.3 Managers:

Interactive dashboards provide real-time analytics and reporting, enabling users to monitor ongoing operations and make well-informed decisions.

4.2.5 Security and Privacy

1. Data encryption and secure storage mechanisms.
2. Authentication and role-based access control of sensitive information.

4.3 Work Flow Diagram



Start
Data Collection: Gather retailers' sales data for 11 months
Data Preprocessing: Clean and prepare the data for analysis
AI Prediction Model: Analyze trends and predict sales for the 12th month
Decision-Making
Guide inventory strategies
Implement inventory strategies based on predictions
Provide insights and offer recommendations
Completion Check: Confirm all criteria met?
Yes
End
Guide sales strategies
Implement sales strategies based on predictions
Return Point
No

Figure 3 Work Flow Diagram

4.3.1 Figure 3 Work Flow Diagram Explanation:

The flowchart illustrates a sales data analysis process. It starts with collecting sales data for several months, followed by preprocessing and predictive modeling to forecast trends for the next month. Based on these predictions, inventory management decisions are made, and insights are provided. The process ends with a confirmation check to ensure all criteria are met.

4.4 Business Workflow Diagram

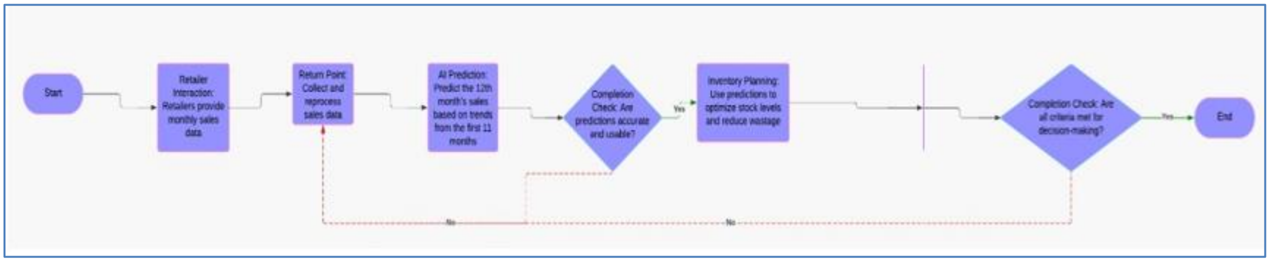

Start
Retailer Interaction: Retailers provide monthly sales data
Return Point: Collect and reprocess sales data
AI Prediction: Predict the 12th month's sales based on trends from the first 11 months
Completion Check: Are predictions accurate and usable?
Yes
Inventory Planning: Use predictions to optimize stock levels and reduce wastage
Completion Check: Are all criteria met for decision-making?
Yes
End
No
No

Figure 4 Business Work Flow Diagram

4.4.1 Figure 4 Business Work Flow Diagram Explanation:

The flowchart describes a retail data management process. It starts with collecting monthly sales data and making predictions based on previous months. A confirmation check assesses product availability, followed by inventory planning to ensure adequate stock and messaging. The process ends with a final validation check.

4.5 Work Breakdown Structure

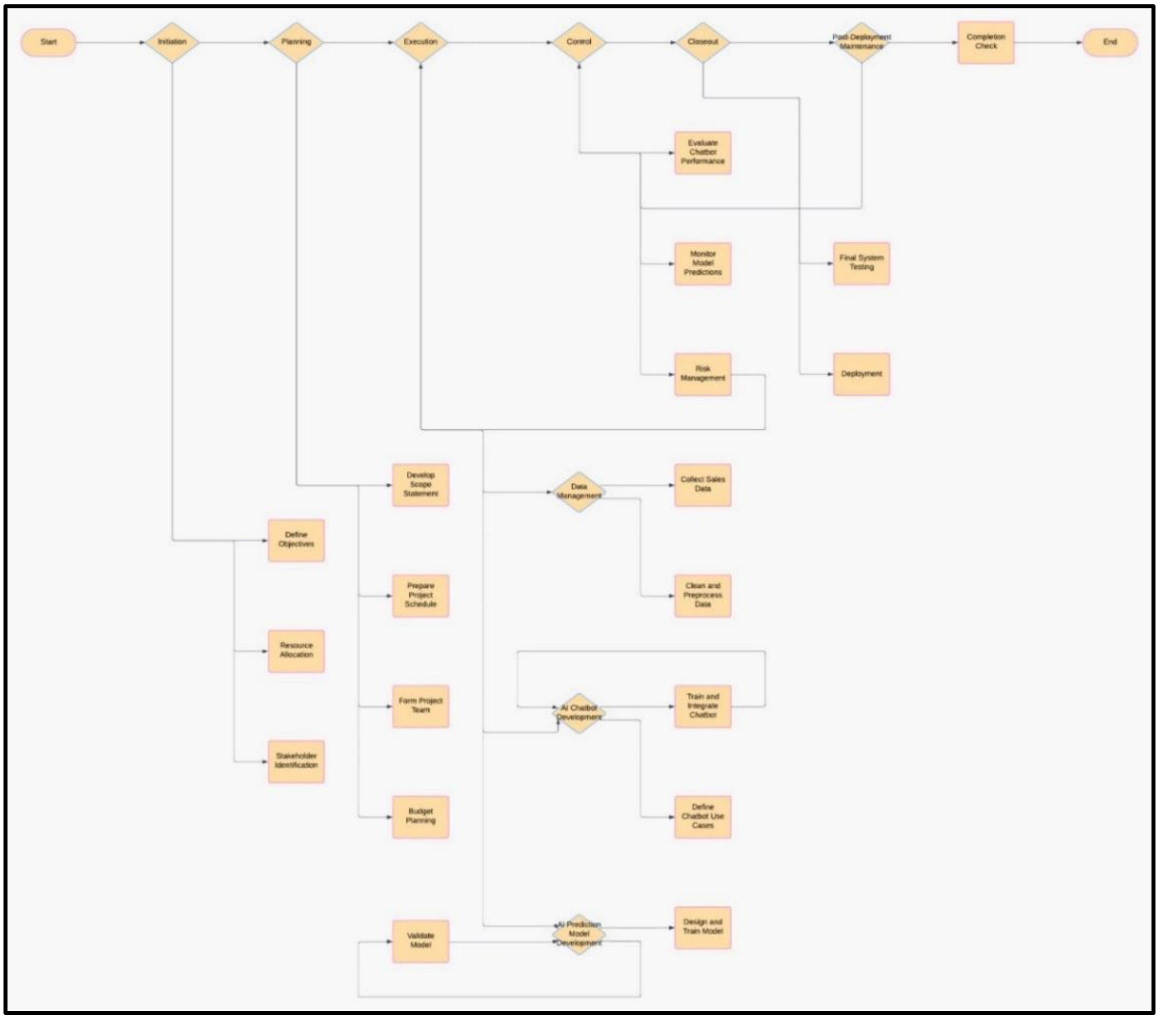


Figure 5 Work Breakdown Structure

4.5.1 <u>Figure 5 WBS Explanation:</u>

The flowchart outlines a project management process. It begins with initial planning and progresses through various stages, including development, testing, and evaluation.

Key decision points determine whether to continue, refine, or deploy the project. The process involves tasks such as vendor selection, submission reviews, and audits. It concludes with a final evaluation before completion.

4.6 Use case Diagram

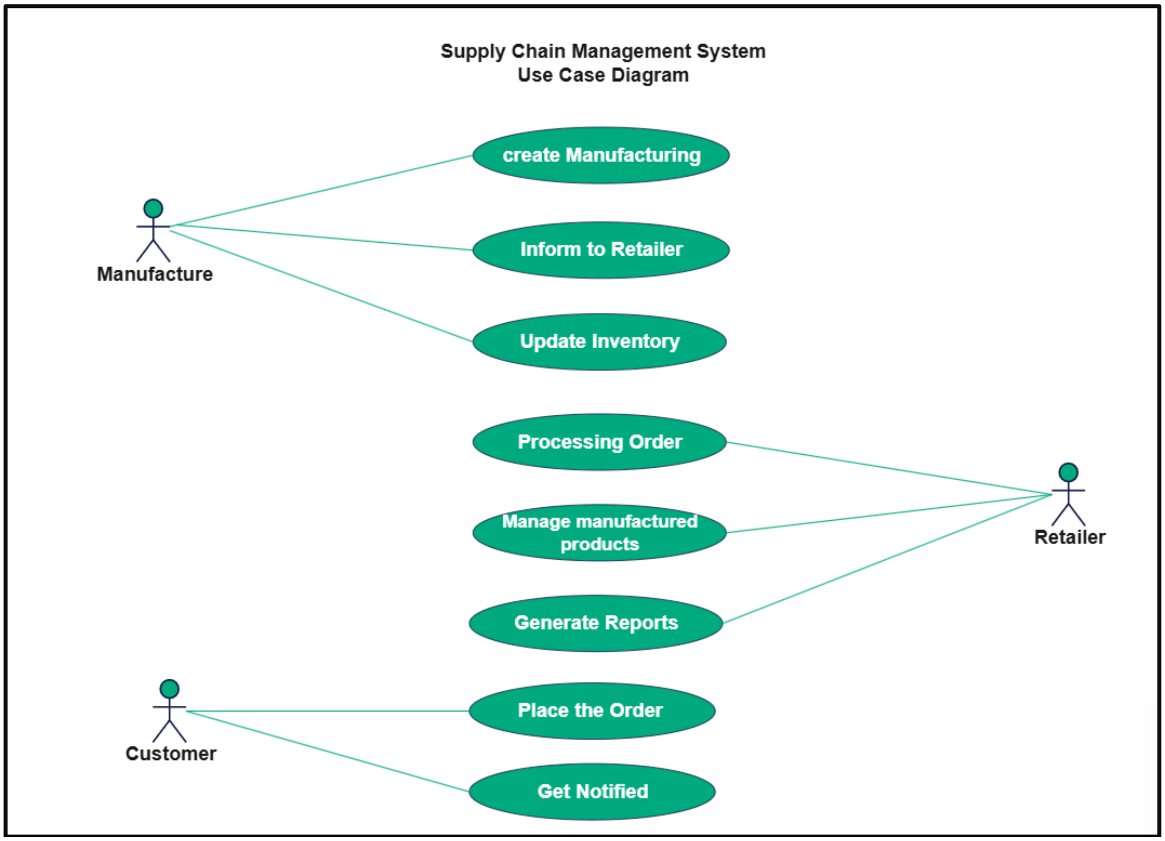


Figure 6 Use Case Diagram

4.6.1 Figure 6 Use Case Diagram Explanation:

The diagram illustrates use cases for a Supply Chain Management System, highlighting key actors and their interactions. The Customer can place orders and receive notifications, while the Retailer is responsible for processing orders and managing products. The manufacturing process includes creating manufacturing plans, informing the retailer, and updating inventory. Additionally, the retailer manages products and generates reports. Overall, the diagram captures the essential interactions within the supply chain system.

4.7 Sequence Diagram

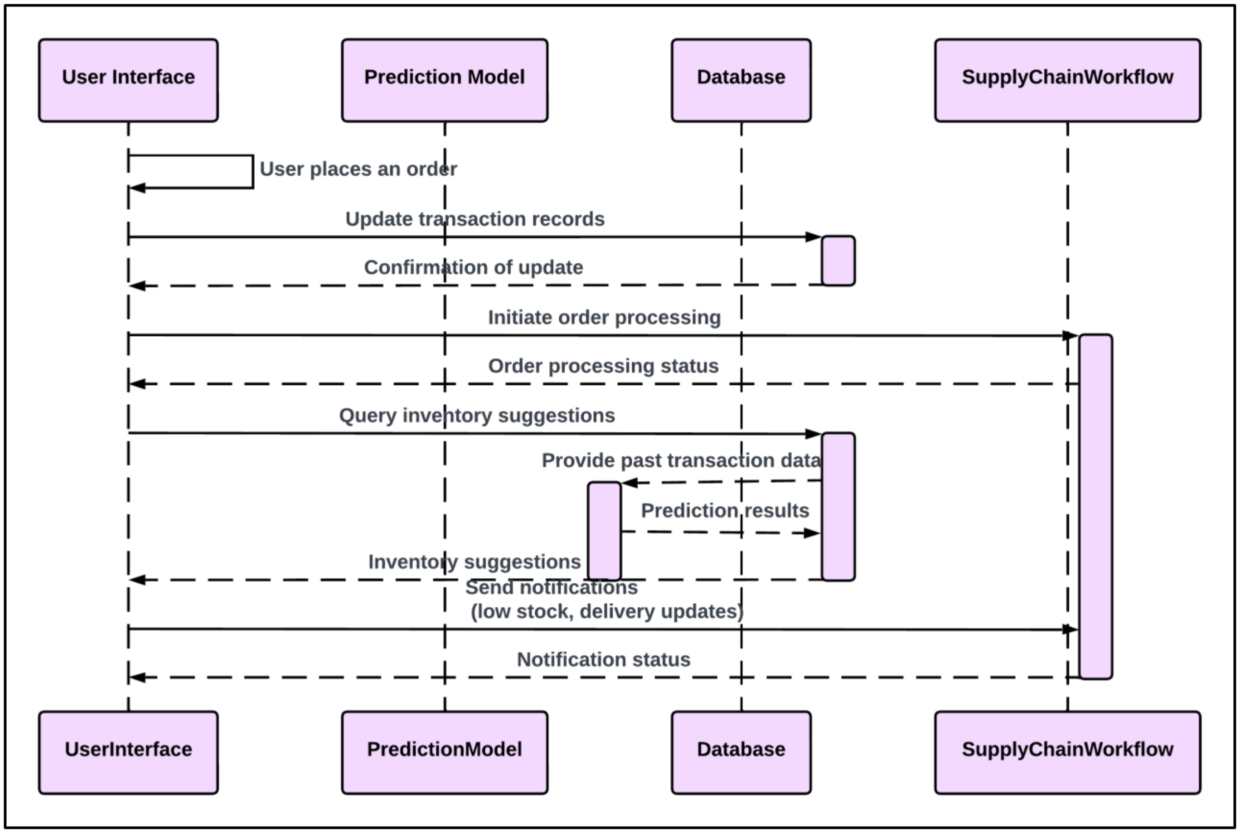


Figure 7 Sequence Diagram

4.7.1 Figure 7 Sequence Diagram Explanation:

The diagram depicts the interaction flow within a Supply Chain Management System. It features several components, including the User Interface, Prediction Model, Database, and Supply Chain Workflow. When a user places an order, the system updates transaction records and confirms the update. The order processing status is initiated, and inventory queries are made to provide past transaction data and predictions. Additionally, inventory suggestions are generated, sending notifications regarding low stock and delivery updates. Overall, the diagram illustrates how these components communicate to manage orders and inventory effectively.

4.8 ER Diagram

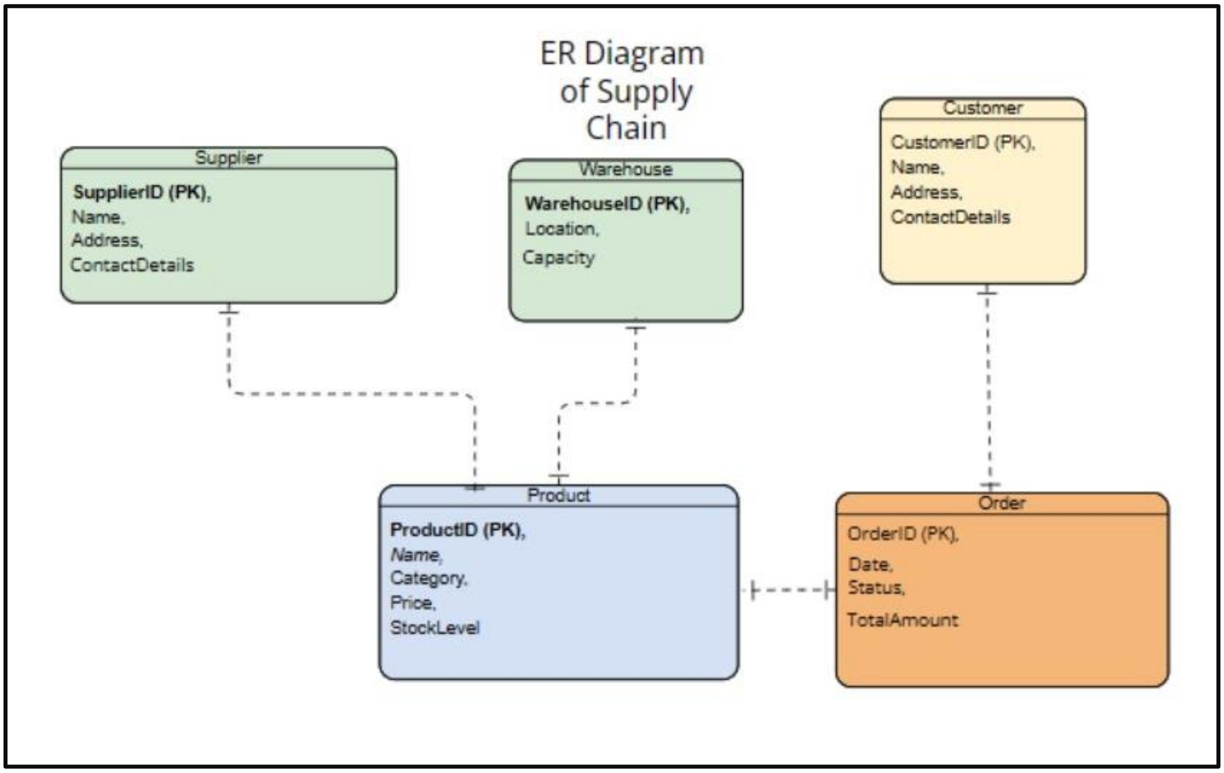


Figure 8 ER Diagram

4.8.1 Figure 8 ER Diagram Explanation:

The diagram shows an Entity-Relationship (ER) model for a Supply Chain system, featuring key entities:

4.8.1.1 Supplier:

Identified by Supplier ID, with attributes such as name, address, and contact details.

4.8.1.2 Warehouse:

Identified by Warehouse ID, including location and capacity.

4.8.1.3 Customer:

Identified by Customer ID, with attributes for name, address, and contact details.

4.8.1.4 Product:

Identified by Product ID, including name, category, price, and stock level.

4.8.1.5 Order:

Identified by Order ID, with attributes for date, status, and total amount.

The relationships illustrate how suppliers provide products to warehouses, which customers then order. This structure aids in managing data within the supply chain.

4.9 Requirements Traceability Matrix

Provide a matrix showing where each feature identified in the SRS is supported by the design components.

		Prediction Model (1)	Inventory Management (2)	Delivery Management (4)
Requirements	R1: Real-time Order Placement		✓	
	R2: Basic Inventory Tracking	✓		
	R3: Basic Chatbot Responses			
	R4: Demand Forecasting	✓	✓	
	R5: Inventory Optimizing	✓	✓	

Table 1 Requirements Traceability Matrix

4.10 System Architectural Design

For your Supply Chain with Prediction and Basic AI Model, the System Architectural Design would involve structure creation that integrates SCM with predictive analytics and basic AI models to optimize inventory, forecasting, logistics, and decision-making. Here is a comprehensive approach to designing the architecture for such a system:

4.10.1 System Architecture for Supply Chain Management System

4.10.1.1 System Requirements and Goals

4.10.1.1.1 Functional Requirements:

4.10.1.1.1.1 Demand Forecasting:

Predict future demand based on products.

4.10.1.1.1.2 Inventory Management:

Optimize stock levels and reorder points.

4.10.1.1.1.3 Route Optimization:

Predict the best shipping routes and delivery times.

4.10.1.1.1.4 Retailer Performance Analysis:

Predict Retailer performance and lead times.

4.10.1.1.2 Non-Functional Requirements:

4.10.1.1.2.1 Scalability:

System should scale with increasing data, users, and queries.

4.10.1.1.2.2 Real-time Processing:

Provide real-time updates on stock levels, orders, and shipments.

4.10.1.1.2.3 Security:

Ensure data privacy and protection of sensitive business information.

4.10.1.1.2.4 Interoperability:

Work with other systems like ERP, CRM, and other SCM software.

4.10.1.2 Architectural Pattern

4.10.1.2.1 Microservices Architecture

The system's components will be modular, meaning that each service can scale independently: inventory management, order processing, AI models, and the chatbot.

4.10.1.2.2 Event-driven Architecture

Events such as inventory or customer orders processes like reordering or shipment planning, which can be communicated through the order.

4.11 Chosen System Architecture

The SCMS with AI prediction and chatbot integration aims to streamline and optimize operations within the supply chain by leveraging AI models for demand forecasting, inventory management. Below is the description of the system architecture, the major component groupings, their interfaces, and significant technical risks with corresponding contingency plans.

4.11.1 System Architectural Design Overview

The architecture is based on a Microservices-based and Event-driven design pattern, which comprises multiple interacting components that are designed to work together to provide an efficient and scalable solution for managing the whole supply chain lifecycle.

4.11.1.1 Major Component Groupings

4.11.1.1.1 Frontend (User Interface):

1. Web Application Built using HTML CSS Java Script. It provides a dashboard for the supply chain managers and staff to interact with the system.

2. Chatbot Interface Built into the web application, this offers users conversational access to solve queries.

4.11.1.1.2 Backend Services (Microservices):

1. Order Management Service is responsible for customer orders the status and order fulfillment.

2. Inventory Management Service is responsible for managing data related to the inventory, monitoring stock levels, and giving restocking recommendations through predictive analytics.

3. AI Prediction Service runs AI models to make predictions of demand, to optimize logistical routes, and to predict required inventory.

4. Retailer Management Service is service keeps a track of performance by the suppliers and predicts when there will be delays and accordingly optimizes supplier interactions.

4.11.1.1.3 AI and Machine Learning Models:

1. Demand Forecasting Model predicts future demand based on historical sales and other external factors by using time-series forecasting, for example, XGBoost.

2. Inventory Optimization Model suggests the optimum stock levels and reorder points based on the forecasted demand and historical data.

3. Natural Language Processing Engine Uses tools such as Rasa, Google Dialog Flow, or Microsoft LUIS to parse and process the user input.

4. Dialogue Management System controls multi turn conversation and retains contextual information from a series of interactions with the user.

5. Backend Integration Connects to microservices to retrieve inventory, orders, and logistics and respond in natural language to a user's queries.

6. Data Storage Layer Database with MySQL for the transactional data (orders, inventory) and more unstructured data or flexible storage in for logistics and chatbot logs.

4.11.2 High Technical Risks and Contingency Plans

4.11.2.1 Risk Associated with Data Integration and Synchronization

Risk As the system requires data from multiple sources, for example, from retailers, product providers, and ERP, etc, there can be inconsistencies in data or latency in synchronization.

4.11.2.2 Contingency Plan:

1. Data validation and error handling mechanism to make sure that data is correct and timely.
2. Event-driven architecture to handle asynchronous updates and keep data synchronization across all systems.
3. Put in place data reconciliation processes and fallback procedures when there is a data discrepancy.

4.11.3 Risk of AI Model Inaccuracy and Performance

4.11.3.1 Risk:

The AI models for demand forecasting, inventory optimization, and logistics will not work as intended in some cases (for example, unanticipated chan ges in the market, incorrect data).

4.11.3.2 Contingency Plan:

1. Retrain the models from time to time with new data to enhance the accuracy.
2. Implement model monitoring and performance tracking to detect any anomalies in model predictions and adapt accordingly.
3. Provide alternatives, such as using lesser heuristics or rule-based systems if AI models cannot come up with predictions.

4.11.4 Chatbot NLP Accuracy Risk

4.11.4.1 Risk:

The chatterbot might misinterpret user inputs or fail to answer complex queries, therefore not yielding a good quality for the user's experience.

4.11.4.2 Contingency Plan:

1. Continuously improve the NLP model by training it with real user queries and feedback.
2. Implement fallback strategies, including asking the user for clarification in case the chatbot is unsure about the query.
3. Offer human escalation options where the user can talk to a human agent if the chatbot is unable to address the issue.

4.11.5 System Scalability and Load Management

4.11.5.1 Risk:

The system will degrade in terms of performance as more users are being added, especially when dealing with real-time data coming from suppliers or logistics tracking.

4.11.5.2 Contingency Plan:

1. Auto-scaling of infrastructure (such as with Kubernetes or AWS Elastic Beanstalk) to scale horizontally based on demand.
2. Use of caching mechanisms such as Redis for reducing the backend services and databases load.
3. Ensure that all microservices are stateless so that they could be replicated during high traffic.

4.11.6 Security Risks

4.11.6.1 Risk:

Exposed data breach or unauthorized access to sensitive supply chain data, like financial data and supplier information

4.11.6.2 Contingency Plan:

1. Use of strong encryption at rest and TLS for data in motion.
2. Use of multi-factor authentication (MFA) for users having high privileges (such as admins, managers).
3. Conduct regular security audits and penetration testing to identify and address vulnerabilities.
4. Implement role-based access control to limit access to sensitive data and operations.

4.12 Discussion of Alternative Designs

Some of the architectural and technical design options were considered for the SCMS with AI Predictions and AI Chatbot Integration during the design process. Below we describe the alternative designs considered, the rationale behind their non-inclusion in the final system architecture.

4.12.1 Monolithic Architecture

4.13 Description:

In monolithic architecture, all the components of a system such as order management, inventory management, AI models, and chatbot services are coupled tightly in a single application, to be deployed as one big unit. Typically, a monolithic application will utilize a single database and a single runtime environment.

4.14 Advantages:

4.14.1 Easier to build:

More straightforward to get started, especially for small teams or scope-limited projects.

4.14.2 Less overhead:

No inter-service communication or orchestration between components is required.

4.14.3 Challenges:

1. Scalability issues: Scaling a monolithic application is difficult because the entire system has to be scaled together. When one part of the application gets high traffic, the whole application may have to be scaled, which might lead to inefficiencies.
2. Lack of flexibility: Changes to one part of the system can impact the entire application, making maintenance and updates more challenging.

3. Difficulty in incorporating AI/ML models: Managing separate services for AI models (e.g., demand forecasting, inventory prediction) alongside other components would be harder in a monolithic design.
4. Deployment challenges: Continuous integration and deployment would be more complex, especially as the system grows.

4.14.4 Centralized AI Model

4.14.5 Description:

Host all the AI prediction models (demand forecasting, inventory optimization, logistics optimization) in one centralized service or server which would compute and return results to other parts of the SCM system.

4.14.6 Advantages

1. Simplicity There's only one point of maintenance for the models using a single model service.
2. Cost-effective is a single AI service can reduce operational costs compared to multiple, isolated AI models.

4.14.7 Challenges

1. Scalability issues as the system grows, centralizing all AI computation in one service can become a bottleneck, as it may struggle to handle high volumes of requests from multiple components simultaneously.

2. Lack of Flexibility The system becomes harder to alter or change a single model of demand forecasts without affecting others. Also, the centralized service would require that much more complexity be added to accommodate other models for other applications.

3. Single point of failure When the central AI service crashes, this can have the cascading effect of crippling all parts of the system involved, including the chatbot, the inventory management module, and the order tracking module.

4.15 System Interface Description

This section describes the system interfaces for the Supply Chain Management System (SCMS) with AI prediction and chatbot integration. These interfaces include interactions with the operating system (O/S), file systems, networking protocols, external libraries, and tools used to implement and run the system.

4.15.1 Operating System Interface

The system is developed as a web application and deployed on cloud platforms with support for local development and testing environments. Below are the operating system requirements and interactions:

4.15.1.1 Supported Operating Systems:

4.15.1.1.1 Development and Local Testing:

Development can be done in Windows 10 for a developer's comfort to develop the web application.

4.15.1.1.2 OS-Level Interaction:

4.15.1.1.2.1 Process Management:

The backend services, including AI-based modules to ensure consistency across environments. The OS interacts for process management.

4.15.1.1.2.2 Security and Permissions:

1. The file system permissions are configured so that sensitive data like configuration files, and database credentials are kept safe.
2. Role-based access control is enforced for services interacting with the operating system to limit unauthorized access and actions.

4.15.2 File Interaction

File interactions in the application are meant for temporary storage, model data handling, and backups. Example items include:

4.15.2.1 Data Formats:

1. CSV for data imports and exports, such as order or inventory updates.
2. Parquet for storing and processing huge datasets (e.g., training data of AI models).
3. Log Files application and server logs in structured log files
   for debugging and tracking.

4.15.2.2 File Management:

1. File Upload/Downloads through the UI: Supplier reports upload, inventory forecast downloads, etc.
2. Data storage into MySQL database by integration with MySQL Database.
3. Scheduled data cleanup jobs manage the usage of storage and archive old logs/datasets.

4.15.3 Systems administration Connection point

4.15.3.1 Depiction:

The framework imparts over organizations to help client cooperations, constant updates, and outside incorporations.

4.15.3.2 Functionalities:

1. Execution of WebSocket convention for continuous updates.
2. Usage of HTTP/HTTPS conventions for electronic correspondence
3. Coordination with SMTP for sending email warnings.

4.15.4 Library Interfaces

The application relies on numerous libraries that handle back-end, AI and chat-bot-related functionality

4.15.4.1 Backend Libraries:

1. Express.js (Node.js): For building RESTful APIs.
2. FastAPI (Python): For lightweight and fast backend services, especially for AI models.
3. Spring Boot (Java, optional): For complex business logic, if required.

4.15.4.2 AI/ML Libraries:

1. TensorFlow / PyTorch: For building and deploying AI models.
2. scikit-learn: For traditional machine learning algorithms like demand forecasting.
3. NumPy, Pandas, Matplotlib: used for data manipulation and visualization.

4.15.4.3 Chatbot libraries

1. Rasa: Used in building conversational AI chatbot.
2. spaCy or NLTK: Natural Language processing, like identifying intent, as well as extracting entities from text.

4.15.4.4 Database Libraries

1. SQLAlchemy (Python): A library to handle relational databases.
2. Mongoose (Node.js): MongoDB handling is done through it.

4.16 Detailed Description of Components

4.16.1 Component-1

4.16.1.1 User Management Component:

4.16.1.1.1 Responsibilities:

1. Manages user-related functions such as registration, login, profile updates, and password recovery to ensure secure and seamless access to the system.
2. Inventory levels are updated in real time.
3. Optimizes stock using AI predictions according to demand forecast.

4.16.1.1.2 Constraints:

1. Guarantees secure storage of user information and provides secure authentication mechanisms for users.
2. Guarantees safe storage and access of inventory data in order to prevent unauthorized modifications.

4.16.1.1.3 Composition:

1. Composed of subcomponents like registration, login, and profile management.
2. User interface for administrators for viewing and managing inventory.

4.16.1.1.4 Interactions:

1. It interacts with the internal user database for storing and retrieving data.
2. It uses authentication services, such as OAuth, to authenticate users safely.

4.16.1.1.5 Resources:

1. It uses an internal database, which is secure for storing and retrieving users' data.
2. It uses external authentication services to safely authenticate users.

4.17 User Interface Design

The User Interface outlines the system's layout, menus, and interactive features, ensuring a user-friendly design that supports efficient navigation and functionality.

4.17.1 Description of the User Interface

The Supply Chain Management System has a responsive and user-friendly web application interface designed for administrators, coordinators, and participants. Key elements include a clean, minimalistic layout, easy navigation, and real-time data displays.

4.17.1.1 Core Interfaces:

1. Login Screen: Secure access with user roles and "Forgot Password" functionality.
2. Dashboard: Overview of inventory levels, orders, and notifications.
3. Inventory Management: View, update, and reorder stock with predictive alerts.
4. Order Management: Track and manage orders with status updates.
5. AI Chatbot: Conversational interface for queries and insights

4.17.2 Screen Images

4.17.2.1 LOGIN SCREEN

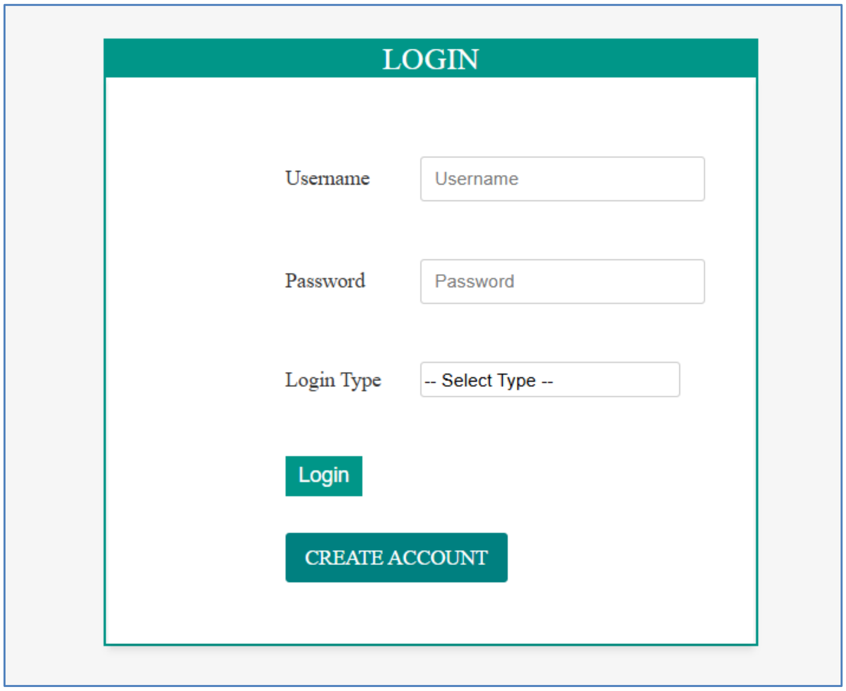


Figure 9 Login Screen

4.17.2.1.1 Figure 9 Login Screen Explain:

The image displays a login interface for a system. It includes fields for the user to enter their Username and Password, along with a dropdown menu for selecting the Login Type. Below the login fields, there are buttons for Login and Create Account. This design provides a straightforward way for users to access their accounts or register for a new one.

4.17.2.2 Create Account

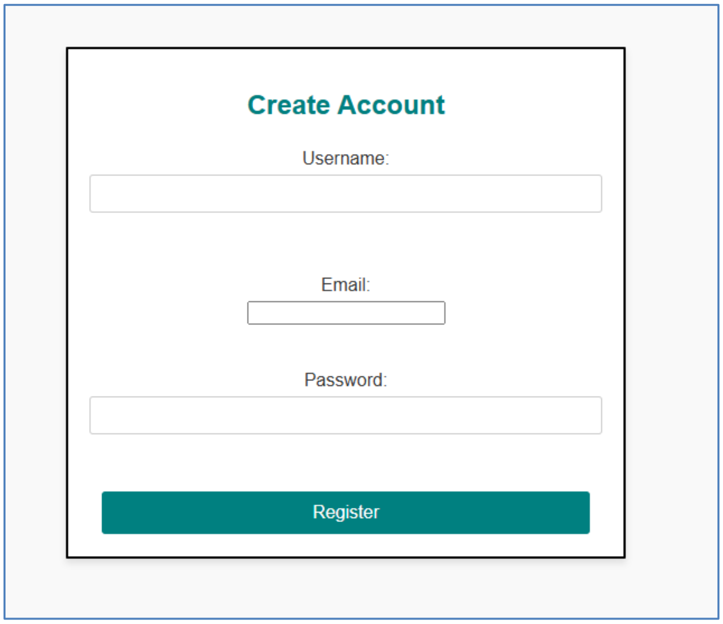
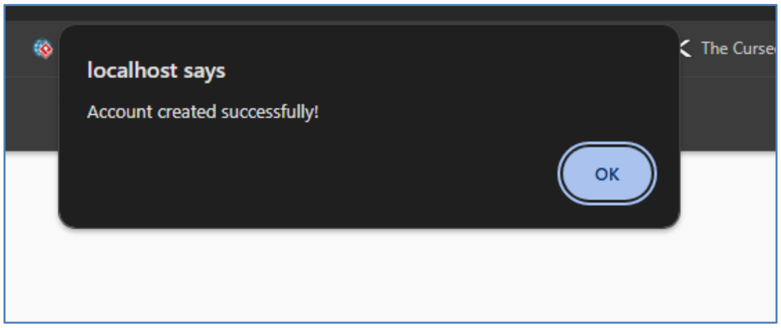


Figure 10 Create Account

4.17.2.2.1 Figure 10 Create Account Explain:

The image shows a registration interface for creating an account. It includes fields for entering a **Username**, **Email**, and **Password**. A **Register** button is provided for submitting the information. This design allows users to set up a new account easily.

4.17.2.3 Account Registered Successfully



localhost says

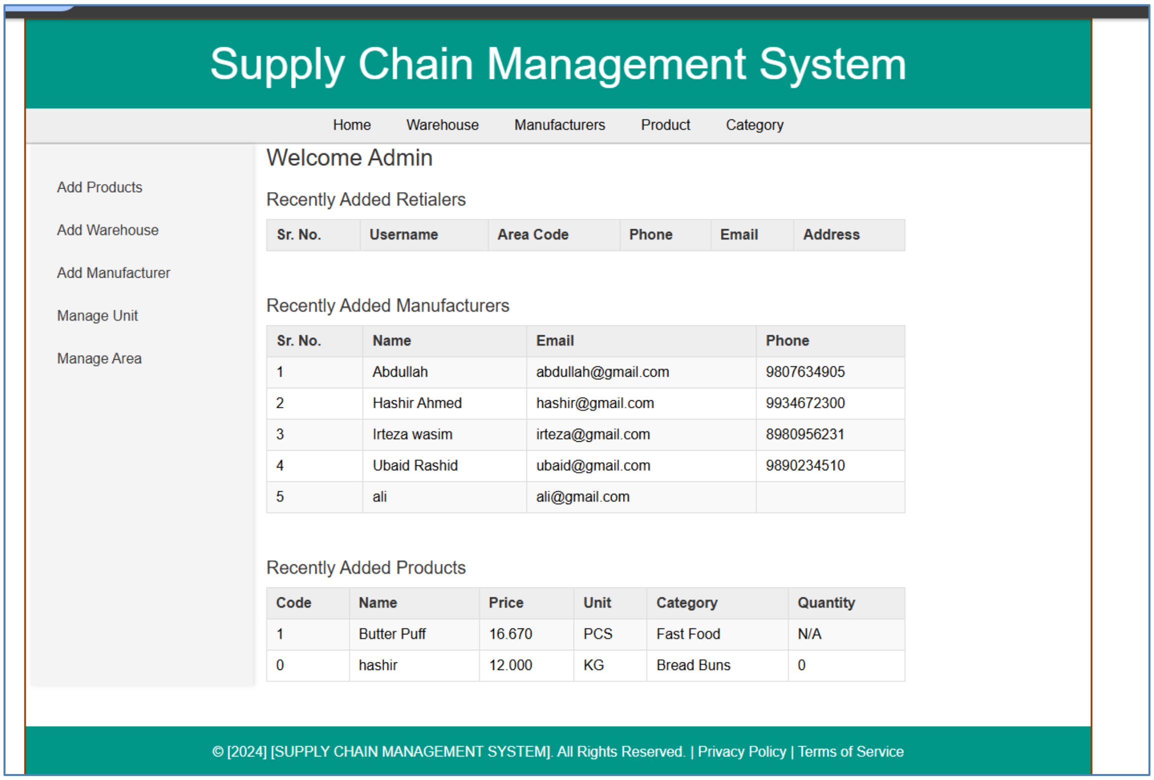
Account created successfully!

Figure 11 Account Registered Successfully

4.17.2.3.1 Figure 11 Account Registered Successfully Explain:

The image displays a notification message indicating that an account was created successfully. The message is from "localhost" and includes an **OK** button for the user to acknowledge the notification. This confirmation alerts users that their registration was successful.

4.17.2.4 Dashboard Screen



Supply Chain Management System

Welcome Admin

Recently Added Retailers

Sr. No.	Username	Area Code	Phone	Email	Address

Recently Added Manufacturers

Sr. No.	Name	Email	Phone
1	Abdullah	abdullah@gmail.com	9807634905
2	Hashir Ahmed	hashir@gmail.com	9934672300
3	Irteza wasim	irteza@gmail.com	8980956231
4	Ubaid Rashid	ubaid@gmail.com	9890234510
5	ali	ali@gmail.com	

Recently Added Products

Code	Name	Price	Unit	Category	Quantity
1	Butter Puff	16.670	PCS	Fast Food	N/A
0	hashir	12.000	KG	Bread Buns	0

© [2024] [SUPPLY CHAIN MANAGEMENT SYSTEM]. All Rights Reserved. | Privacy Policy | Terms of Service

Figure 12 Dashboard

4.17.2.4.1 Figure 12 Dashboard Explain:

The image displays an admin dashboard for a Supply Chain Management System. It features sections for recently added retailers and manufacturers, showing their usernames, emails, and phone numbers, along with a list of recently added products that includes code, name, price, and stock status. This interface helps manage various system components efficiently.

4.17.2.5 Add Product:

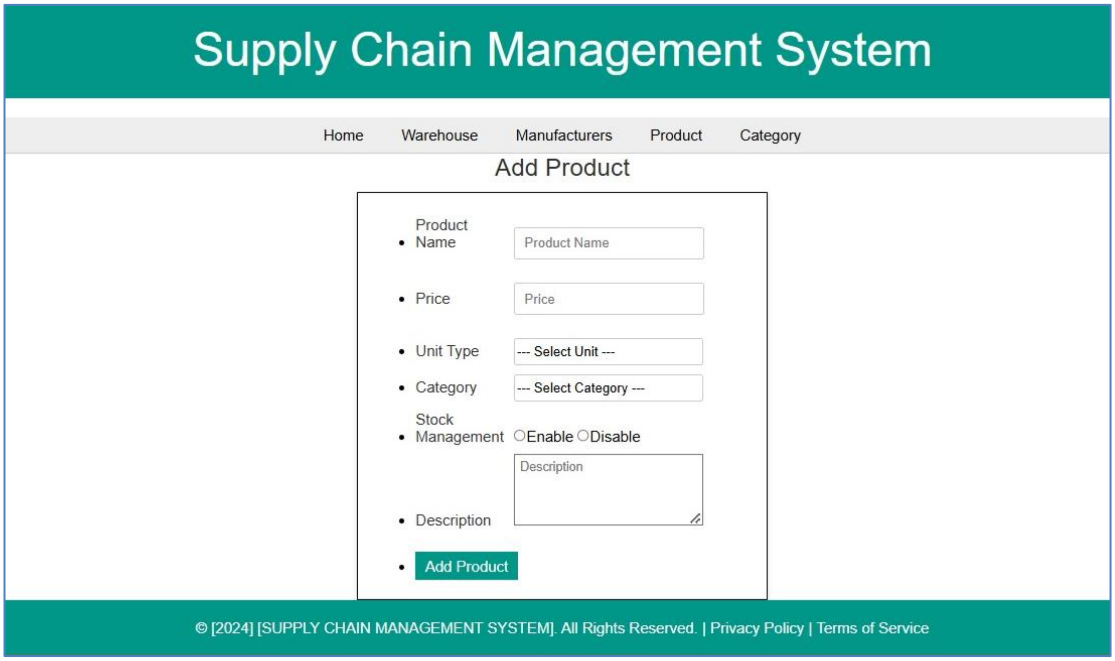


Figure 13 Add Product

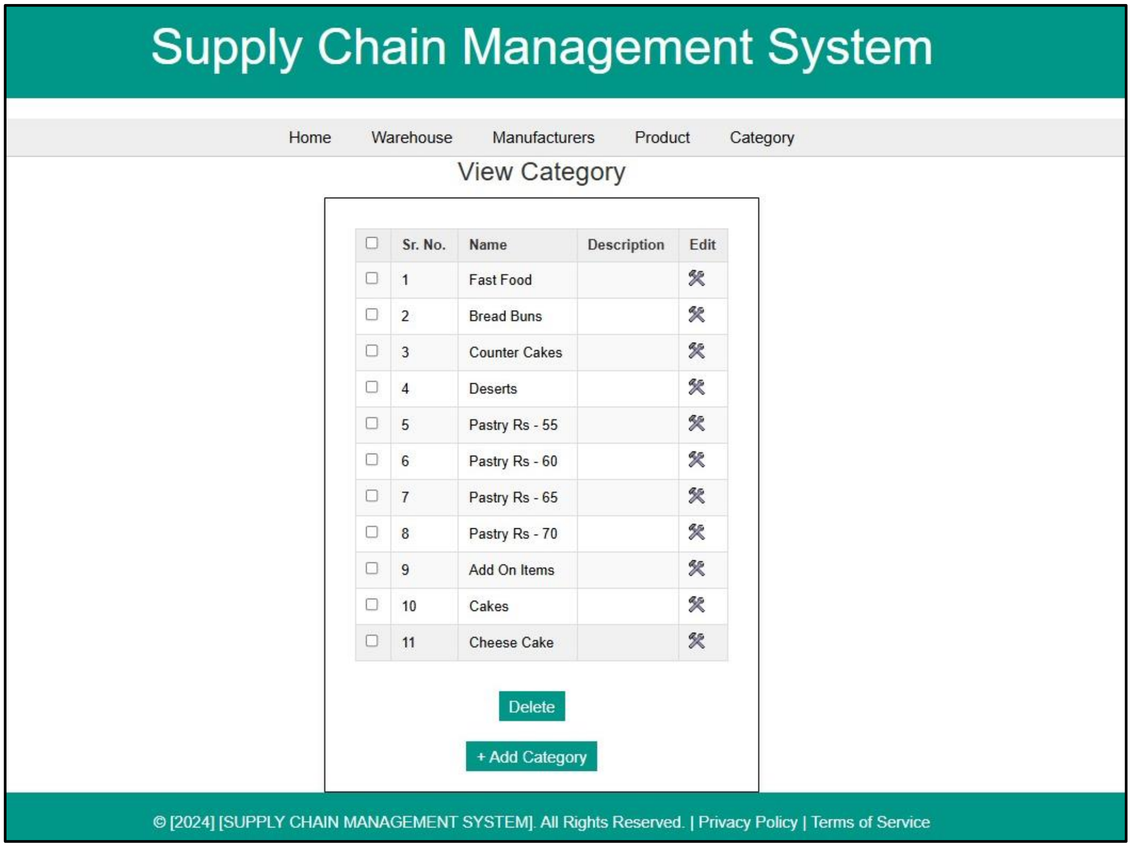
4.17.2.5.1 Figure 13 Dashboard Explain:

The image shows a form for adding a product within a Supply Chain Management System. It includes fields for:

1. Product Name
2. Price
3. Unit Type (with a dropdown selection)
4. Category (with a dropdown selection)
5. Stock Management (with an option to enable or disable)
6. Description

A button labeled **Add Product** allows users to submit the information. This form facilitates product entry into the system

4.17.2.6 View Category:



□	Sr. No.	Name	Description	Edit
□	1	Fast Food		⚒
□	2	Bread Buns		⚒
□	3	Counter Cakes		⚒
□	4	Deserts		⚒
□	5	Pastry Rs - 55		⚒
□	6	Pastry Rs - 60		⚒
□	7	Pastry Rs - 65		⚒
□	8	Pastry Rs - 70		⚒
□	9	Add On Items		⚒
□	10	Cakes		⚒
□	11	Cheese Cake		⚒

© [2024] [SUPPLY CHAIN MANAGEMENT SYSTEM]. All Rights Reserved. | Privacy Policy | Terms of Service

Figure 14 View Category

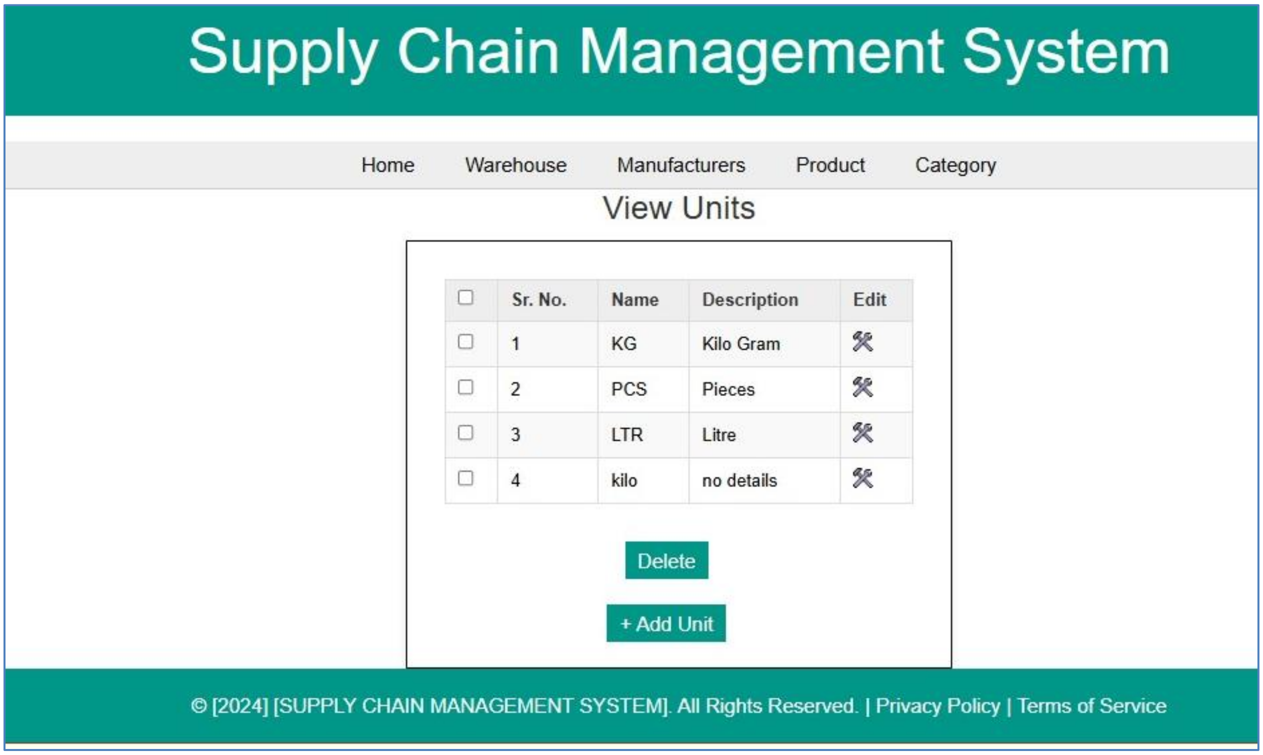
4.17.2.6.1 Figure 14 View Category Explain:

The image displays a **View Category** interface for a Supply Chain Management System. It features a table listing various product categories, including:

1. Serial Number (Sr. No.)
2. Category Name
3. Description
4. Edit options for each category.

A button for **Delete** is available for removing selected categories, along with an option to **Add Category**. This interface allows for efficient management of product categories

4.17.3 View Units


Supply Chain Management System
Home
Warehouse
Manufacturers
Product
Category
View Units
Delete
+ Add Unit
© [2024] [SUPPLY CHAIN MANAGEMENT SYSTEM]. All Rights Reserved. | Privacy Policy | Terms of Service

□	Sr. No.	Name	Description	Edit
□	1	KG	Kilo Gram	🛠
□	2	PCS	Pieces	🛠
□	3	LTR	Litre	🛠
□	4	kilo	no details	🛠

Figure 15 View Units

4.17.3.1 Figure 15 View Units Explain:

The image shows a **View Units** interface for a Supply Chain Management System. It features a table that lists various units of measurement, including:

1. Serial Number (Sr. No.)
2. Unit Name
3. Description
4. Edit options for each unit.

There is also a **Delete** button for removing selected units and an option to **Add Unit**. This interface facilitates the management of measurement units within the system.

5. IMPLEMENTATION

5.1 CODE:

5.1.1 Login:

```
<?php
  include('config.php');
  $reqErr = $loginErr = "";
  if($_SERVER['REQUEST_METHOD'] == "POST") {
    if(!empty($_POST['txtUsername']) && !empty($_POST['txtPassword']) && isset($_POST['login_type'])){
      session_start();
      $username = $_POST['txtUsername'];
      $password = $_POST['txtPassword'];
      $_SESSION['sessLogin_type'] = $_POST['login_type'];
      if($_SESSION['sessLogin_type'] == "retailer") {
        $query_selectRetailer = "SELECT retailer_id,username,password FROM retailer WHERE username='$username'
AND password='$password'";
        $result = mysqli_query($con,$query_selectRetailer);
        $row = mysqli_fetch_array($result);
        if($row) {
          $_SESSION['retailer_id'] =  $row['retailer_id'];
          $_SESSION['sessUsername'] = $_POST['txtUsername'];
          $_SESSION['sessPassword'] = $_POST['txtPassword'];
          $_SESSION['retailer_login'] = true;
          header('Location:retailer/index.php');
        }
        else {
          $loginErr = "* Username or Password is incorrect.";
        }
      }
      else if($_SESSION['sessLogin_type'] == "manufacturer") {
        $query_selectManufacturer = "SELECT man_id,username,password FROM manufacturer WHERE
username='$username' AND password='$password'";
        $result = mysqli_query($con,$query_selectManufacturer);
        $row = mysqli_fetch_array($result);
        if($row) {
          $_SESSION['manufacturer_id'] =  $row['man_id'];
          $_SESSION['sessUsername'] = $_POST['txtUsername'];
          $_SESSION['sessPassword'] = $_POST['txtPassword'];
          $_SESSION['manufacturer_login'] = true;
          header('Location:manufacturer/index.php');
        }
        else {
          $loginErr = "* Username or Password is incorrect.";
        }
      }
      else if($_SESSION['sessLogin_type'] == "admin") {
        $query_selectAdmin = "SELECT username,password FROM admin WHERE username='$username' AND
password='$password'";
        $result = mysqli_query($con,$query_selectAdmin);
        $row = mysqli_fetch_array($result);
          if($row) {
            $_SESSION['admin_login'] = true;
            $_SESSION['sessUsername'] = $_POST['txtUsername'];
            $_SESSION['sessPassword'] = $_POST['txtPassword'];
            header('Location:index2.php');
          }
          else {
            $loginErr = "* Username or Password is incorrect.";
```

```
                    }
                }
            }
        else {
            $reqErr = "* All fields are required.";
        }
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title> Login </title>
    <link rel="stylesheet" href="styles.css" >
</head>

<body class="login-box">

    <div style="background-color: white;" >

    <h1 >LOGIN</h1>
    <form action="" method="POST" class="login-form" style="border: none; width: 436px;">
    <ul class="form-list">
    <li>
        <div class="label-block"> <label for="login:username">Username</label> </div>
        <div class="input-box"> <input type="text" id="login:username" name="txtUsername" placeholder="Username" />
</div>
    </li>
    <li>
        <div class="label-block"> <label for="login:password">Password</label> </div>
        <div class="input-box"> <input type="password" id="login:password" name="txtPassword" placeholder="Password" />
</div>
    </li>
    <li>
        <div class="label-block"> <label for="login:type">Login Type</label> </div>
        <div class="input-box">
        <select name="login_type" id="login:type">
        <option value="" disabled selected>-- Select Type --</option>

        <option value="admin">Admin</option>
        </select>
        </div>
    </li>
    <li>

    <input type="submit" value="Login" class="submit_button" />
        </a>


        <span class="error_message"> <?php echo $loginErr; echo $reqErr; ?> </span>
        <br><br><br>
        <a href="register.php" class="createaccount">CREATE ACCOUNT </a>

    </li>
    </ul>
    </form>
    </div>
</body>
</html>
```

5.1.2 index:

```
<?php
    include("config.php");
    session_start();
    if(isset($_SESSION['admin_login'])) {
        if($_SESSION['admin_login'] == true) {
            //select last 5 retialers
            // $query_selectRetailer = "SELECT * FROM retailer,area WHERE retailer.area_id=area.area_id ORDER BY
retailer_id DESC LIMIT 5";
            // $result_selectRetailer = mysqli_query($con,$query_selectRetailer);
            //select last 5 manufacturers
            $query_selectManufacturer = "SELECT * FROM manufacturer ORDER BY man_id DESC LIMIT 5";
            $result_selectManufacturer = mysqli_query($con,$query_selectManufacturer);
            //select last 5 products
            $query_selectProducts = "SELECT * FROM products,categories,unit WHERE
products.pro_cat=categories.cat_id AND products.unit=unit.id ORDER BY pro_id DESC LIMIT 5";
            $result_selectProducts = mysqli_query($con,$query_selectProducts);
        }
        else {
            header('Location:../index.php');
        }
    }
    else {
        header('Location:../index.php');
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title> Admin: Home </title>
    <link rel="stylesheet" href="cssadmin.css" >


</head>
<body>
    <?php
        include("header.inc.php");
        include("nav_admin.inc.php");

    ?>

    <div class="flex-container" style="display:flex">


    <div class="bar">
    <?php

            include("aside_admin.inc.php");
    ?>
    </div >
    <div class="everything">
    <section>

        <h1>Welcome Admin</h1>
        <article>
            <h2>Recently Added Manufacturers</h2>
            <table class="table_displayData">
            <tr>
                <th>Sr. No.</th>
                <th>Name</th>
                <th>Email</th>
        <th>Phone</th>

      </tr>

      <?php $i=1; while($row_selectManufacturer = mysqli_fetch_array($result_selectManufacturer)) { ?>
      <tr>
        <td> <?php echo $i; ?> </td>
        <td> <?php echo $row_selectManufacturer['man_name']; ?> </td>
        <td> <?php echo $row_selectManufacturer['man_email']; ?> </td>
        <td> <?php echo $row_selectManufacturer['man_phone']; ?> </td>
      </tr>
      <?php $i++; } ?>
    </table>
    </article>

    <article>
      <h2>Recently Added Products</h2>
      <table class="table_displayData">
      <tr>
        <th> Code </th>
        <th> Name </th>
        <th> Price </th>
        <th> Unit </th>
        <th> Category </th>
        <th> Quantity </th>
      </tr>
      <?php $i=1; while($row_selectProducts = mysqli_fetch_array($result_selectProducts)) { ?>
      <tr>
        <td> <?php echo $row_selectProducts['pro_id']; ?> </td>
        <td> <?php echo $row_selectProducts['pro_name']; ?> </td>
        <td> <?php echo $row_selectProducts['pro_price']; ?> </td>
        <td> <?php echo $row_selectProducts['unit_name']; ?> </td>
        <td> <?php echo $row_selectProducts['cat_name']; ?> </td>
        <td> <?php if($row_selectProducts['quantity'] == NULL){ echo "N/A";} else {echo
$row_selectProducts['quantity'];} ?> </td>
      </tr>
      <?php $i++; } ?>
    </table>
    </article>
  </section>
  </div>
  </div>
  <?php
    include("footer.inc.php");
  ?>
</body>
</html>
```

5.1.3 Show Graph:

```
<?php
  include("config.php");
  include("validate_data.php");
  session_start();

  if (isset($_SESSION['admin_login']) && $_SESSION['admin_login'] == true) {
    $startYear = $endYear = "";
    $startYearErr = $endYearErr = $fileErr = $requireErr = $confirmMessage = "";
    $startYearHolder = $endYearHolder = "";

    if ($_SERVER['REQUEST_METHOD'] == "POST") {
      if (!empty($_POST['txtStartYear'])) {
        $startYearHolder = $_POST['txtStartYear'];
        $validate_start = validate_year($_POST['txtStartYear']);
```

```
                if ($validate_start === 1) {
                    $startYear = $_POST['txtStartYear'];
                } else {
                    $startYearErr = $validate_start;
                }
            }
            if (!empty($_POST['txtEndYear'])) {
                $endYearHolder = $_POST['txtEndYear'];
                $validate_end = validate_year($_POST['txtEndYear']);
                if ($validate_end === 1) {
                    $endYear = $_POST['txtEndYear'];
                } else {
                    $endYearErr = $validate_end;
                }
            }

            if (isset($_FILES['dataset']) && $_FILES['dataset']['error'] == 0) {
                $upload_dir = "uploads/";
                $filename = basename($_FILES['dataset']['name']);
                $target_path = $upload_dir . $filename;

                if (move_uploaded_file($_FILES['dataset']['tmp_name'], $target_path)) {
                    if ($startYear && $endYear) {
                        $_SESSION['dataset_path'] = $target_path;
                        $_SESSION['start_year'] = $startYear;
                        $_SESSION['end_year'] = $endYear;

                        // Call Python script to generate the graph
                        $pythonScriptPath = "path/to/your/product_prediction_app.py";  // Adjust this path
                        $command = "python3 $pythonScriptPath $target_path $startYear $endYear";
                        $output = shell_exec($command);  // Executes the Python script and returns the output

                        // Check if the Python script was successful
                        if ($output) {
                            $confirmMessage = "Graphs generated successfully!";
                            // Optionally, display the generated images here
                        } else {
                            $confirmMessage = "Error generating graphs.";
                        }
                        header("Location: show_graph.php");  // Redirect to the page that shows the graphs
                        exit();
                    } else {
                        $requireErr = "* Valid Start and End Year required.";
                    }
                } else {
                    $fileErr = "File upload failed.";
                }
            } else {
                $fileErr = "Please upload a dataset file.";
            }
        }
    } else {
        header("Location: index.php");
        exit();
    }
    function validate_year($year) {
        if (!preg_match("/^[0-9]{4}$/", $year)) {
            return "* Invalid year format";
        }
        if ($year < 1900 || $year > date("Y")) {
            return "* Year out of range";
        }
        return 1;
    }
?>
<!DOCTYPE html>
```

```
<html>
<head>
  <title>Upload Dataset & View Graph</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="cssadmin.css">
</head>
<body>
  <?php
    include("header.inc.php");
    include("nav_admin.inc.php");
  ?>
  <section>
    <h1>Upload Dataset and View Graph</h1>
    <form action="" method="POST" enctype="multipart/form-data" class="form">
      <ul class="form-list">
        <li>
          <div class="label-block"><label for="dataset">Upload Dataset (CSV)</label></div>
          <div class="input-box"><input type="file" name="dataset" id="dataset" required></div>
          <span class="error_message"><?php echo $fileErr; ?></span>
        </li>
        <li>
          <div class="label-block"><label for="start_year">Start Year</label></div>
          <div class="input-box"><input type="text" id="start_year" name="txtStartYear" placeholder="e.g. 2020"
value="<?php echo $startYearHolder; ?>" required /></div>
          <span class="error_message"><?php echo $startYearErr; ?></span>
        </li>
        <li>
          <div class="label-block"><label for="end_year">End Year</label></div>
          <div class="input-box"><input type="text" id="end_year" name="txtEndYear" placeholder="e.g. 2023"
value="<?php echo $endYearHolder; ?>" required /></div>
          <span class="error_message"><?php echo $endYearErr; ?></span>
        </li>
        <li>
          <input type="submit" value="Show Graph" class="submit_button" />
          <span class="error_message"><?php echo $requireErr; ?></span>
          <span class="confirm_message"><?php echo $confirmMessage; ?></span>
        </li>
      </ul>
    </form>
  </section>
  <?php include("footer.inc.php"); ?>
</body>
</html>
```

5.1.4 Testing.php:

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);

$graph = '';  // Initialize the graph variable to hold the base64 image string.
$error = '';  // To hold error messages if any.

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['csv_file'])) {
  $upload_dir = __DIR__ . '/uploads/';
  if (!is_dir($upload_dir)) mkdir($upload_dir);

  $filename = basename($_FILES['csv_file']['name']);
  $filepath = $upload_dir . $filename;

  // Check if the file is a valid CSV
  if ($_FILES['csv_file']['type'] != 'text/csv') {
    $error = "Please upload a valid CSV file.";
  } else {
    if (move_uploaded_file($_FILES['csv_file']['tmp_name'], $filepath)) {
            // Run the Python script and capture the output.
            $command = "python predict.py " . escapeshellarg($filepath) . " 2>&1";
            $output = shell_exec($command);

            // Debugging output — check if base64 data is being generated.
            if (!$output) {
                $error = "Error executing the Python script.";
            } else {
                // Check if the output is a valid base64 string (start with iVBOR)
                if (substr(trim($output), 0, 5) === 'iVBOR') {
                    $graph = trim($output);  // This stores the base64-encoded image string for the graph.
                } else {
                    $error = "Error: Python script did not generate a valid base64 string.";
                }
            }
        } else {
            $error = "Error uploading the file.";
        }
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Product Sales Graph</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        body { font-family: Arial, sans-serif; padding: 20px; }
        img { max-width: 100%; border: 1px solid #ccc; margin-top: 20px; }
        .error { color: red; }
    </style>
</head>
<body>
    <h2>Upload CSV to View Product-wise Sales Graph</h2>
    <form method="post" enctype="multipart/form-data">
        <input type="file" name="csv_file" accept=".csv" required>
        <button type="submit">Generate Graph</button>
    </form>

    <?php if ($error): ?>
        <p class="error"><?php echo $error; ?></p>
    <?php endif; ?>

    <!-- Show Chart -->
    <section>
        <div style="text-align: center;"><h1>Product Insights</h1></div>

        <!-- File Upload -->
        <div class="mb-3" style="margin: 20px;">
            <label for="fileInput" class="form-label">Choose CSV File (optional)</label>
            <input type="file" class="form-control" id="fileInput" accept=".csv">
        </div>

        <!-- Year, Month, Day Dropdowns -->
        <div class="row" style="margin: 20px;">
            <!-- Year From Dropdown -->
            <div class="col-md-3">
                <label>Year From</label>
                <select id="yearFrom" class="form-select">
                    <?php for ($y = 2016; $y <= 2025; $y++) {
```

```
                echo "<option value='$y'>$y</option>";
            } ?>
        </select>
    </div>

    <!-- Year To Dropdown -->
    <div class="col-md-3">
        <label>Year To</label>
        <select id="yearTo" class="form-select">
            <?php for ($y = 2016; $y <= 2025; $y++) {
                $selected = ($y == 2025) ? "selected" : "";
                echo "<option value='$y' $selected>$y</option>";
            } ?>
        </select>
    </div>

    <!-- Month Dropdown -->
    <div class="col-md-3">
        <label>Select Month</label>
        <select id="monthSelect" class="form-select">
            <option value="01">January</option>
            <option value="02">February</option>
            <option value="03">March</option>
            <option value="04">April</option>
            <option value="05" selected>May</option>
            <option value="06">June</option>
            <option value="07">July</option>
            <option value="08">August</option>
            <option value="09">September</option>
            <option value="10">October</option>
            <option value="11">November</option>
            <option value="12">December</option>
        </select>
    </div>

    <!-- Day Dropdown -->
    <div class="col-md-3">
        <label>Select Day</label>
        <select id="daySelect" class="form-select">
            <?php for($d=1; $d<=31; $d++) {
                $val = str_pad($d, 2, "0", STR_PAD_LEFT);
                echo "<option value='$val'>$val</option>";
            } ?>
        </select>
    </div>
</div>

<!-- Particular Day Input -->
<div class="mb-3" style="margin: 20px;">
    <label for="particularDay" class="form-label">Particular Day</label>
    <input type="text" id="particularDay" class="form-control" placeholder="e.g., 2025-05-05">
</div>

<!-- Show Button -->
<div style="margin: 20px;">
    <button class="btn btn-success" onclick="showData()">Show Data</button>
</div>

<!-- Chart -->
<div style="margin: 20px;">
    <canvas id="productChart" height="100"></canvas>
</div>
```

```
  </section>

  <script>
    const yearSelect = document.getElementById("yearSelect");
    const monthSelect = document.getElementById("monthSelect");
    const daySelect = document.getElementById("daySelect");
    const chartCtx = document.getElementById("productChart").getContext('2d');

    let chart = new Chart(chartCtx, {
      type: 'bar',
      data: {
        labels: ['Sales'],
        datasets: [{
          label: 'Sales Data',
          data: [0],
          backgroundColor: 'rgba(54, 162, 235, 0.6)'
        }]
      },
      options: {
        scales: {
          y: {
            beginAtZero: true
          }
        }
      }
    });

    function showData() {
      const year = yearSelect.value;
      const month = monthSelect.value;
      const day = daySelect.value;
      const label = `${year}-${month}-${day}`;

      // Placeholder data - replace with real DB value if needed
      const randomSales = Math.floor(Math.random() * 100);

      chart.data.labels = [label];
      chart.data.datasets[0].label = `Sales on ${label}`;
      chart.data.datasets[0].data = [randomSales];
      chart.update();
    }

    function searchData() {
      let keyword = document.getElementById('searchInput').value.toLowerCase();
      alert("Search triggered for: " + keyword);
    }

    window.onload = showData;

  </script>

  <?php include("footer.inc.php"); ?>
</body>
</html>
```

5.1.5 ui.php

```
<?php
  include("config.php");
  session_start();
  if(isset($_SESSION['admin_login'])) {
    $query_selectProducts = "SELECT * FROM products,categories,unit WHERE products.pro_cat=categories.cat_id
AND products.unit=unit.id ORDER BY pro_id";
    $result_selectProducts = mysqli_query($con,$query_selectProducts);
  } else {
    header('Location:../index.php');
  }
?>
<!DOCTYPE html>
<html>
<head>
  <title> View Products </title>
  <link rel="stylesheet" href="cssadmin.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <?php
    include("header.inc.php");
    if(isset($_SESSION['admin_login'])){
      include("nav_admin.inc.php");
    }
  ?>
  <section>
    <div style="text-align: center;"><h1>Product Insights</h1></div>

    <!-- File Upload -->
    <div class="mb-3" style="margin: 20px;">
      <label for="fileInput" class="form-label">Choose CSV File (optional)</label>
      <input type="file" class="form-control" id="fileInput" accept=".csv">
    </div>
    <!-- Year, Month, Day Dropdowns -->
    <div class="row" style="margin: 20px;">
  <!-- Year From Dropdown -->
  <div class="col-md-3">
    <label>Year From</label>
    <select id="yearFrom" class="form-select">
      <?php for ($y = 2016; $y <= 2025; $y++) {
        echo "<option value='$y'>$y</option>";
      } ?>
    </select>
  </div>

  <!-- Year To Dropdown -->
  <div class="col-md-3">
    <label>Year To</label>
    <select id="yearTo" class="form-select">
      <?php for ($y = 2016; $y <= 2025; $y++) {
        $selected = ($y == 2025) ? "selected" : "";
        echo "<option value='$y' $selected>$y</option>";
      } ?>
    </select>
  </div>

  <!-- Month Dropdown -->
  <div class="col-md-3">
    <label>Select Month</label>
    <select id="monthSelect" class="form-select">
      <option value="01">January</option>
      <option value="02">February</option>
      <option value="03">March</option>
      <option value="04">April</option>
      <option value="05" selected>May</option>
```

```
        <option value="06">June</option>
        <option value="07">July</option>
        <option value="08">August</option>
        <option value="09">September</option>
        <option value="10">October</option>
        <option value="11">November</option>
        <option value="12">December</option>
      </select>
    </div>

    <!-- Day Dropdown -->
    <div class="col-md-3">
      <label>Select Day</label>
      <select id="daySelect" class="form-select">
        <?php for($d=1; $d<=31; $d++) {
          $val = str_pad($d, 2, "0", STR_PAD_LEFT);
          echo "<option value='$val'>$val</option>";
        } ?>
      </select>
    </div>
  </div>

  <!-- Particular Day Input -->
  <div class="mb-3" style="margin: 20px;">
    <label for="particularDay" class="form-label">Particular Day</label>
    <input type="text" id="particularDay" class="form-control" placeholder="e.g., 2025-05-05">
  </div>

  <!-- Button for Particular Day -->
  <div style="margin: 20px;">
    <button class="btn btn-warning" onclick="showParticularDay()">Show Particular Day Data</button>
  </div>
      <!-- Show Button -->
      <div style="margin: 20px;">
        <button class="btn btn-success" onclick="showData()">Show Data</button>
      </div>

      <!-- Graph -->
      <div style="margin: 20px;">
        <canvas id="productChart" height="100"></canvas>
      </div>
    </section>

    <script>
      const yearSelect = document.getElementById("yearSelect");
      const monthSelect = document.getElementById("monthSelect");
      const daySelect = document.getElementById("daySelect");
      const chartCtx = document.getElementById("productChart").getContext('2d');

      let chart = new Chart(chartCtx, {
        type: 'bar',
        data: {
          labels: ['Sales'],
          datasets: [{
            label: 'Sales Data',
            data: [0],
            backgroundColor: 'rgba(54, 162, 235, 0.6)'
          }]
        },
        options: {
          scales: {
            y: {
```

```
            beginAtZero: true
          }
        }
      }
    });

    function showData() {
      const year = yearSelect.value;
      const month = monthSelect.value;
      const day = daySelect.value;
      const label = `${year}-${month}-${day}`;

      // Placeholder data - replace with real DB value if needed
      const randomSales = Math.floor(Math.random() * 100);

      chart.data.labels = [label];
      chart.data.datasets[0].label = `Sales on ${label}`;
      chart.data.datasets[0].data = [randomSales];
      chart.update();
    }
    function searchData() {
      let keyword = document.getElementById('searchInput').value.toLowerCase();
      alert("Search triggered for: " + keyword);
    }
    window.onload = showData;
  </script>
 <div style="margin: 20px;">
  <button class="btn btn-primary" onclick="return validateDateRange()">Show Year-Month-Day Data</button>
</div>
<script>
function validateDateRange() {
  const from = parseInt(document.getElementById("yearFrom").value);
  const to = parseInt(document.getElementById("yearTo").value);
  const month = document.getElementById("monthSelect").value;
  const day = document.getElementById("daySelect").value;

  if (from > to) {
    alert("Year From must be less than or equal to Year To.");
    return false;
  }

  alert("Selected Range:\nFrom Year: " + from + "\nTo Year: " + to + "\nMonth: " + month + "\nDay: " + day);
  return true;
}
</script>
  <?php include("footer.inc.php"); ?>
</body>
</html>
```

5.1.6 database.db

```
CREATE DATABASE user_database;
USE user_database;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE `admin` (
 `id` int(11) NOT NULL,
 `username` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `admin`
--
INSERT INTO `admin` (`id`, `username`, `password`) VALUES
(1, 'admin', 'admin123');
--
-- Table structure for table `products`
--
CREATE TABLE `products` (
  `pro_id` int(11) NOT NULL,
  `pro_name` varchar(25) NOT NULL,
  `pro_desc` text,
  `pro_price` decimal(10,3) NOT NULL,
  `unit` int(11) NOT NULL,
  `pro_cat` int(11) NOT NULL,
  `quantity` int(6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `products`
--
INSERT INTO `products` (`pro_id`, `pro_name`, `pro_desc`, `pro_price`, `unit`, `pro_cat`, `quantity`) VALUES
(1, 'Butter Puff', '', '16.670', 2, 1, NULL),
(2, 'Corn Puff', '', '16.670', 2, 1, NULL),
(3, 'Garlic Cheese Roll', '', '39.570', 2, 1, NULL),
(4, 'Butter Stuffed Bun', '', '42.000', 2, 1, NULL),
(5, 'Paneer Tikka S. Bun', '', '52.500', 2, 1, NULL),
(6, 'Burger Bun 4 PCS', '', '42.000', 2, 2, NULL),
(7, 'Hot Dog Bun 4 PCS', '', '46.000', 2, 2, NULL);
--
-- Dumping data for table `unit`
--
INSERT INTO `unit` (`id`, `unit_name`, `unit_details`) VALUES
(1, 'KG', 'Kilo Gram'),
(2, 'PCS', 'Pieces'),
(3, 'LTR', 'Litre');

--table for categories
CREATE TABLE `categories` (
  `cat_id` int(11) NOT NULL,
  `cat_name` varchar(25) NOT NULL,
  `cat_details` text
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `categories`
INSERT INTO `categories` (`cat_id`, `cat_name`, `cat_details`) VALUES
(1, 'Fast Food', ''),
(2, 'Bread Buns', ''),
(3, 'Counter Cakes', ''),
(4, 'Deserts', ''),
(5, 'Pastry Rs - 55', ''),
(6, 'Pastry Rs - 60', ''),
(7, 'Pastry Rs - 65', ''),
(8, 'Pastry Rs - 70', ''),
(9, 'Add On Items', ''),
(10, 'Cakes', ''),
(11, 'Cheese Cake', '');

-- Table structure for table `manufacturer`
--

CREATE TABLE `manufacturer` (
  `man_id` int(11) NOT NULL,
  `man_name` varchar(25) NOT NULL,
  `man_email` varchar(50) DEFAULT NULL,
  `man_phone` varchar(10) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Dumping data for table `manufacturer`
--

INSERT INTO `manufacturer` (`man_id`, `man_name`, `man_email`, `man_phone`) VALUES
(1, 'Ubaid Rashid', 'ubaid@gmail.com', '9890234510'),
(2, 'Irteza wasim', 'irteza@gmail.com', '8980956231'),
(3, 'Hashir Ahmed', 'hashir@gmail.com', '9934672300'),
(4, 'Abdullah', 'abdullah@gmail.com', '9807634905');
--
-- Table structure for table `warehouse`
--

CREATE TABLE `warehouse` (
  `manager_id` int(11) NOT NULL,
  `manager_name` varchar(25) NOT NULL,
  `stocknumber` int(25) NOT NULL,
  `product_name` varchar(25) NOT NULL,
  `product_id` int(11) NOT NULL,

  `address` varchar(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
INSERT INTO `warehouse` (`manager_id`, `manager_name`, `stocknumber`, `product_name`, `product_id`,  `address`) VALUES
(1, 'salman', '10', 'slice juice carton', 1, 'johar block 5 Karachi'),
(2, 'dawood', '20', 'shan masala carton', 2, 'korangi zaman town'),
(3, 'hamza', '24', 'cocacola carton', 3, 'defence phase 4'),
(4, 'farrukh', '12', 'milkpack carton', 4, 'bahria town karachi');
--
-- Table structure for table `area`
--
CREATE TABLE `area` (
  `area_id` int(11) NOT NULL,
  `area_name` varchar(50) NOT NULL,
  `area_postalcode` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `area`
--

INSERT INTO `area` (`area_id`, `area_name`, `area_postalcode`) VALUES
(1, 'johar', 'jhr'),
(2, 'korangi', 'krg'),
(3, 'shah faisal', 'sf'),
(4, 'clifton', 'cc'),
(5, 'dha', 'dha');
-- --------------------------------------------------------
--
-- Table structure for table `unit`
--

CREATE TABLE `unit` (
  `id` int(11) NOT NULL,
  `unit_name` varchar(20) NOT NULL,
  `unit_details` text
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `unit`
--
INSERT INTO `unit` (`id`, `unit_name`, `unit_details`) VALUES
(1, 'KG', 'Kilo Gram'),
(2, 'PCS', 'Pieces'),
(3, 'LTR', 'Litre');
--------------------------------------------------------
-- Table structure for table `invoice`
--
CREATE TABLE `invoice` (
  `invoice_id` int(11) NOT NULL,
  `order_id` int(11) NOT NULL,
  `retailer_id` int(11) NOT NULL,
  `dist_id` int(11) NOT NULL,
  `date` date NOT NULL,
  `total_amount` decimal(10,3) NOT NULL,
  `comments` text
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `invoice`
--
INSERT INTO `invoice` (`invoice_id`, `order_id`, `retailer_id`, `dist_id`, `date`, `total_amount`, `comments`) VALUES
(1, 2, 4, 3, '2015-04-28', '5119.500', ''),
(2, 1, 2, 5, '2015-04-28', '4780.150', ''),
(3, 3, 1, 1, '2015-04-28', '8891.680', ''),
(4, 4, 5, 4, '2015-04-28', '7888.960', ''),
(5, 5, 5, 1, '2020-12-07', '8919.880', 'asd');

-- --------------------------------------------------------
--
-- Table structure for table `invoice_items`
--

CREATE TABLE `invoice_items` (
  `invoice_items_id` int(11) NOT NULL,
  `invoice_id` int(11) NOT NULL,
  `product_id` int(11) NOT NULL,
  `quantity` int(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `invoice_items`
--

INSERT INTO `invoice_items` (`invoice_items_id`, `invoice_id`, `product_id`, `quantity`) VALUES
(1, 1, 4, 20),
(2, 1, 5, 5),
(3, 1, 7, 10),
(4, 1, 9, 10),
(5, 1, 12, 6),
(6, 1, 14, 5),
(7, 2, 1, 20),
(8, 2, 2, 15),
(9, 2, 5, 10),
(10, 2, 8, 5),
(11, 2, 10, 8),
(12, 2, 11, 10),
(13, 2, 13, 2),
(14, 2, 15, 3),
(15, 3, 1, 2),
(16, 3, 2, 4),
(17, 3, 3, 3),
(18, 3, 4, 8),
(19, 3, 5, 10),
(20, 3, 6, 12),
(21, 3, 8, 4),
(22, 3, 11, 10),
(23, 3, 13, 3),
(24, 3, 14, 5),
(25, 3, 15, 6),
(26, 4, 2, 12),
(27, 4, 4, 30),
(28, 4, 8, 4),
(29, 4, 11, 20),
(30, 4, 14, 8);
--
-- Table structure for table `orders`
--

CREATE TABLE `orders` (
  `order_id` int(11) NOT NULL,
  `date` date NOT NULL,
  `retailer_id` int(11) NOT NULL,
  `approved` tinyint(1) NOT NULL,
  `status` tinyint(1) NOT NULL,
  `total_amount` decimal(10,3) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `orders`
--

INSERT INTO `orders` (`order_id`, `date`, `retailer_id`, `approved`, `status`, `total_amount`) VALUES
(1, '2015-04-28', 2, 1, 1, '4780.150'),
(2, '2015-04-28', 4, 1, 1, '5119.500'),
(3, '2015-04-28', 1, 1, 1, '8891.680'),
(4, '2015-04-28', 5, 1, 1, '7888.960'),
(5, '2015-04-28', 5, 1, 0, '8919.880'),
(6, '2020-12-07', 1, 0, 0, '50.010');
--
-- Table structure for table `order_items`
--

CREATE TABLE `order_items` (
  `order_items_id` int(11) NOT NULL,
  `order_id` int(11) NOT NULL,
  `pro_id` int(11) NOT NULL,
  `quantity` int(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `order_items`
--

INSERT INTO `order_items` (`order_items_id`, `order_id`, `pro_id`, `quantity`) VALUES
(1, 1, 1, 20),
(2, 1, 2, 15),
(3, 1, 5, 10),
(4, 1, 8, 5),
(5, 1, 10, 8),
(6, 1, 11, 10),
(7, 1, 13, 2),
(8, 1, 15, 3),
(9, 2, 4, 20),
(10, 2, 5, 5),
(11, 2, 7, 10),
(12, 2, 9, 10),
(13, 2, 12, 6),
(14, 2, 14, 5),
(15, 3, 1, 2),
(16, 3, 2, 4),
(17, 3, 3, 3),
(18, 3, 4, 8),
(19, 3, 5, 10),
(20, 3, 6, 12),
(21, 3, 8, 4),
(22, 3, 11, 10),
(23, 3, 13, 3),
(24, 3, 14, 5),
(25, 3, 15, 6),
(26, 4, 2, 12),
(27, 4, 4, 30),
(28, 4, 8, 4),
(29, 4, 11, 20),
(30, 4, 14, 8),
(31, 5, 1, 1),
(32, 5, 2, 3),
(33, 5, 3, 5),
(34, 5, 4, 15),
(35, 5, 6, 30),
(36, 5, 8, 45),
(37, 5, 12, 20),
(38, 5, 14, 5),
(39, 6, 1, 1),
(40, 6, 2, 2);
```

5.1.7 Config.php

```
<?php
$conn_error='Could not Connect.';
$mysql_host='localhost';
$mysql_user='root';
$mysql_pass='';
$mysql_db='user_database';

$con=mysqli_connect($mysql_host, $mysql_user, $mysql_pass, $mysql_db);
// Check connection
if (mysqli_connect_errno()){
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```

5.1.8 prediction model app.py

```
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import os
import uuid

app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads'
GRAPH_FOLDER = 'static/graphs'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(GRAPH_FOLDER, exist_ok=True)

# Load and clean dataset
def load_dataset(filepath):
    df = pd.read_csv(filepath)
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
    df['TotalAmount'] = pd.to_numeric(df['TotalAmount'], errors='coerce')
    df.dropna(subset=['Date', 'TotalAmount', 'Product Category'], inplace=True)
    return df

# Filter by year range
def filter_data_by_year(df, start_year, end_year):
    return df[(df['Date'].dt.year >= start_year) & (df['Date'].dt.year <= end_year)]

# Plotting functions
def save_plot(fig, filename):
    path = os.path.join(GRAPH_FOLDER, filename)
    fig.savefig(path, bbox_inches='tight')
    plt.close(fig)
    return path

def plot_sales_per_year(df):
    df['Year'] = df['Date'].dt.year
    summary = df.groupby('Year')['TotalAmount'].sum().reset_index()
    fig, ax = plt.subplots(figsize=(12, 4))
    sns.barplot(data=summary, x='Year', y='TotalAmount', ax=ax)
    ax.set_title('Total Sales Per Year')
    return save_plot(fig, f"sales_year_{uuid.uuid4()}.png")

def plot_products_vs_month(df):
    df['Month'] = df['Date'].dt.to_period('M').astype(str)
    summary = df.groupby(['Month', 'Product Category']).size().reset_index(name='Count')
    fig, ax = plt.subplots(figsize=(12, 4))
    sns.barplot(data=summary, x='Month', y='Count', hue='Product Category', ax=ax)
    ax.set_title('Products vs Month')
    return save_plot(fig, f"products_month_{uuid.uuid4()}.png")

def plot_products_vs_date(df):
    df['DateStr'] = df['Date'].dt.strftime('%Y-%m-%d')
    summary = df.groupby(['DateStr', 'Product Category']).size().reset_index(name='Count')
    fig, ax = plt.subplots(figsize=(12, 4))
    sns.barplot(data=summary, x='DateStr', y='Count', hue='Product Category', ax=ax)
    ax.set_title('Products vs Date')
    return save_plot(fig, f"products_date_{uuid.uuid4()}.png")

# Prediction
def predict_most_sold_product(df):
    return df['Product Category'].value_counts().idxmax()

# Routes
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['dataset']
        start_year = int(request.form['start_year'])
        end_year = int(request.form['end_year'])

        if file:
            filename = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(filename)
            df = load_dataset(filename)
            filtered_df = filter_data_by_year(df, start_year, end_year)

            img1 = plot_sales_per_year(filtered_df)
            img2 = plot_products_vs_month(filtered_df)
            img3 = plot_products_vs_date(filtered_df)
            most_sold = predict_most_sold_product(filtered_df)

            return render_template('results.html', img1=img1, img2=img2, img3=img3, most_sold=most_sold)

    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

5.1.9 Generategraph.py

```
# generate_graph.py
import sys
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import base64
from io import BytesIO
from datetime import datetime

# Functions from original file
def load_dataset(filepath):
    df = pd.read_csv(filepath)
    df['Date'] = pd.to_datetime(df['Date'])
    df['TotalAmount'] = pd.to_numeric(df['TotalAmount'], errors='coerce')
    df.dropna(subset=['Date', 'TotalAmount', 'Product Category'], inplace=True)
    return df

def filter_data_by_year(df, start_year, end_year):
    return df[(df['Date'].dt.year >= start_year) & (df['Date'].dt.year <= end_year)]

def plot_sales_per_year(df):
    df['Year'] = df['Date'].dt.year
    summary = df.groupby('Year')['TotalAmount'].sum().reset_index()
    fig, ax = plt.subplots(figsize=(10, 4))
    sns.barplot(data=summary, x='Year', y='TotalAmount', ax=ax)
    ax.set_title('Total Sales Per Year')
    ax.set_xlabel('Year')
    ax.set_ylabel('Total Amount')
    ax.tick_params(axis='x', rotation=45)
    plt.tight_layout()
    return fig
# Get arguments
if len(sys.argv) != 4:
    print("Usage: python generate_graph.py <dataset_path> <start_year> <end_year>")
    sys.exit(1)

dataset_path = sys.argv[1]
start_year = int(sys.argv[2])
end_year = int(sys.argv[3])

try:
    df = load_dataset(dataset_path)
    filtered_df = filter_data_by_year(df, start_year, end_year)

    fig = plot_sales_per_year(filtered_df)

    buffer = BytesIO()
    fig.savefig(buffer, format="png")
    buffer.seek(0)
    img_base64 = base64.b64encode(buffer.read()).decode('utf-8')
    print(img_base64)

except Exception as e:
    print("Error:", str(e))
    sys.exit(1)
```

5.1.10 Add Product.php:

```
<?php
    include("config.php");
    include("validate_data.php");
    session_start();

    if (!isset($_SESSION['admin_login']) || $_SESSION['admin_login'] !== true) {
        header('Location: ../index.php');
        exit();
    }

    // Fetch categories and units from database
    $categories = mysqli_query($con, "SELECT cat_id, cat_name FROM categories");
    $units = mysqli_query($con, "SELECT id, unit_name FROM unit");

    // Initialize variables
    $productName = $productPrice = $productUnit = $productCategory = $stockStatus = $productDesc = "";
    $productNameErr = $productPriceErr = $formError = $successMessage = "";
    $productNameVal = $productPriceVal = $productDescVal = "";

    if ($_SERVER['REQUEST_METHOD'] == "POST") {

        // Validate product name
        if (!empty($_POST['txtProductName'])) {
            $productNameVal = trim($_POST['txtProductName']);
            $productName = $productNameVal;
        }

        // Validate product price
        if (!empty($_POST['txtProductPrice'])) {
            $productPriceVal = trim($_POST['txtProductPrice']);
            $validatePrice = validate_price($productPriceVal);
            if ($validatePrice === 1) {
                $productPrice = $productPriceVal;
            } else {
                $productPriceErr = $validatePrice;
            }
        }

        // Capture dropdown selections
        $productUnit = $_POST['cmbProductUnit'] ?? '';
        $productCategory = $_POST['cmbProductCategory'] ?? '';

        // Stock management radio selection
        $stockStatus = $_POST['rdbStock'] ?? '';

        // Optional product description
        if (!empty($_POST['txtProductDescription'])) {
            $productDescVal = trim($_POST['txtProductDescription']);
            $productDesc = $productDescVal;
        }

        // If all required fields are valid
        if ($productName && $productPrice && $productUnit && $productCategory && $stockStatus) {
            $quantity = ($stockStatus == 1) ? 0 : "NULL";

            $query = "INSERT INTO products (pro_name, pro_desc, pro_price, unit, pro_cat, quantity)
                      VALUES ('$productName', '$productDesc', '$productPrice', '$productUnit', '$productCategory',
$quantity)";

            if (mysqli_query($con, $query)) {
                echo "<script>alert('Product added successfully.');</script>";
                header('Refresh:0');
                exit();
            } else {
                $formError = "Failed to add product. Please try again.";
            }
        } else {
            $formError = "* All fields except Description are required.";
        }
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title>Add Product</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="cssadmin.css">
</head>
```

```
<body>

<?php include("header.inc.php"); ?>
<?php include("nav_admin.inc.php"); ?>

<section>
    <h1>Add New Product</h1>
    <form method="POST" class="form">
        <ul class="form-list">
            <li>
                <div class="label-block"><label for="productName">Product Name</label></div>
                <div class="input-box">
                    <input type="text" id="productName" name="txtProductName" placeholder="Enter product name"
value="<?= $productNameVal; ?>" required>
                </div>
                <span class="error_message"><?= $productNameErr; ?></span>
            </li>

            <li>
                <div class="label-block"><label for="productPrice">Price</label></div>
                <div class="input-box">
                    <input type="text" id="productPrice" name="txtProductPrice" placeholder="Enter price" value="<?=
$productPriceVal; ?>" required>
                </div>
                <span class="error_message"><?= $productPriceErr; ?></span>
            </li>

            <li>
                <div class="label-block"><label for="productUnit">Unit Type</label></div>
                <div class="input-box">
                    <select name="cmbProductUnit" id="productUnit" required>
                        <option value="" disabled selected>Select Unit</option>
                        <?php while ($unit = mysqli_fetch_assoc($units)) { ?>
                            <option value="<?= $unit['id']; ?>"><?= $unit['unit_name']; ?></option>
                        <?php } ?>
                    </select>
                </div>
            </li>

            <li>
                <div class="label-block"><label for="productCategory">Category</label></div>
                <div class="input-box">
                    <select name="cmbProductCategory" id="productCategory" required>
                        <option value="" disabled selected>Select Category</option>
                        <?php while ($cat = mysqli_fetch_assoc($categories)) { ?>
                            <option value="<?= $cat['cat_id']; ?>"><?= $cat['cat_name']; ?></option>
                        <?php } ?>
                    </select>
                </div>
            </li>

            <li>
                <div class="label-block"><label>Stock Management</label></div>
                <label><input type="radio" name="rdbStock" value="1" required> Enable</label>
                <label><input type="radio" name="rdbStock" value="2"> Disable</label>
            </li>

            <li>
                <div class="label-block"><label for="productDescription">Description</label></div>
                <div class="input-box">
                    <textarea id="productDescription" name="txtProductDescription" placeholder="Optional
description"><?= $productDescVal; ?></textarea>
```

```
                </div>
            </li>

            <li>
                <input type="submit" value="Add Product" class="submit_button">
                <span class="error_message"><?= $formError; ?></span>
                <span class="confirm_message"><?= $successMessage; ?></span>
            </li>
        </ul>
    </form>
</section>

<?php include("footer.inc.php"); ?>

</body>
</html>
```

5.1.11 Chatbot App.py:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from train_logic import train_on_data, get_answer, log_interaction

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"]
)

train_on_data()

@app.get("/ask/")
def ask(q: str):
    answer = get_answer(q)
    log_interaction(q, answer)
    return {"response": answer}
```

5.1.12 train_logic.py:

```
import json
import re
import os

qa_data = {}
preprocessed_qa_map = {}
chat_history = []
user_language = None
qa_file = "scm_qa.json"

def load_qa_data():
    global qa_data, preprocessed_qa_map
    if os.path.exists(qa_file):
        with open(qa_file, "r", encoding="utf-8") as f:
            qa_data = json.load(f)
            preprocessed_qa_map = {
                preprocess(q): a for q, a in qa_data.items()
            }
    else:
        qa_data = {}
        preprocessed_qa_map = {}

def preprocess(text):
    return re.sub(r'[^\w\s]', '', text.lower().strip())

def tokenize(text):
    return set(preprocess(text).split())

def detect_language_choice(text):
    text = preprocess(text)
    if "english" in text:
        return "english"
    elif "roman urdu" in text or "urdu" in text:
        return "roman urdu"
    return None

def get_best_fuzzy_match(user_tokens):
    best_score = 0
    best_answer = None

    for q, a in qa_data.items():
        q_tokens = tokenize(q)
        intersection = user_tokens & q_tokens
        union = user_tokens | q_tokens
        if not union:
            continue
        similarity = len(intersection) / len(union)

        if similarity > best_score:
            best_score = similarity
            best_answer = a

    if best_score >= 0.5:  # You can tweak this threshold
        return best_answer
    return None

def get_answer(question):
    global user_language
    if not qa_data:
        load_qa_data()

    clean_q = preprocess(question)
    user_tokens = tokenize(question)

    if user_language is None:
        choice = detect_language_choice(clean_q)
        if choice:
            user_language = choice
            return (
                "Okay, I will continue in English. Ask your question."
                if user_language == "english"
                else "Theek hai, ab mein Roman Urdu mein jawab doon ga. Sawal kijiyega."
            )
        else:
            return "Welcome! Which language do you want to talk in: English or Roman Urdu?"

    if "english" in clean_q and user_language != "english":
        user_language = "english"
        return "Language switched to English. Please continue."
    if ("roman urdu" in clean_q or "urdu" in clean_q) and user_language != "roman urdu":
        user_language = "roman urdu"
        return "Zuban ab Roman Urdu mein ho gayi hai. Barah-e-karam apna sawal poochain."

    greetings = ["hi", "hello", "salam", "salaam"]
    if clean_q in greetings:
        return "How can I assist you?" if user_language == "english" else "Mein aap ki madad kesay kar sakta hoon?"

    # Exact or partial match
    for stored_q, answer in preprocessed_qa_map.items():
        if stored_q in clean_q or clean_q in stored_q:
            return answer

    # Fuzzy match
    fuzzy_answer = get_best_fuzzy_match(user_tokens)
    if fuzzy_answer:
        return fuzzy_answer

    return (
        "Sorry, I didn’t understand that. Can you rephrase your question?"
        if user_language == "english"
        else "Maaf kijiye, mein samajh nahi paaya. Kya aap apna sawal dobara farmaenge?"
    )

def log_interaction(q, a):
    with open("chat_log.json", "a", encoding="utf-8") as f:
        json.dump({"question": q, "answer": a}, f)
        f.write("\n")

def train_on_data():
    load_qa_data()
    print(f"Trained on {len(qa_data)} QA pairs from {qa_file}")
```

5.1.13 chatbot_ui.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SCM Assistant</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            height: 100vh;
            margin: 0;
            display: flex;
            align-items: center;
            justify-content: center;
        }

        .chatbot-container {
            width: 400px;
            height: 600px;
            background: rgba(255, 255, 255, 0.95);
            backdrop-filter: blur(10px);
            border-radius: 20px;
            box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);
            display: flex;
            flex-direction: column;
            overflow: hidden;
        }

        .chat-header {
            background: linear-gradient(135deg, #4f46e5, #7c3aed);
            color: white;
            padding: 20px;
            text-align: center;
            font-size: 18px;
            font-weight: bold;
        }

        .chat-messages {
            flex: 1;
            padding: 20px;
            overflow-y: auto;
            display: flex;
            flex-direction: column;
            gap: 15px;
        }

        .message {
            display: flex;
            align-items: flex-start;
            gap: 10px;
        }

        .message.user {
            flex-direction: row-reverse;
        }

        .message-content {
            max-width: 70%;
            padding: 10px 15px;
            border-radius: 15px;
            font-size: 14px;
            line-height: 1.4;
        }

        .user-message {
            background: linear-gradient(135deg, #4f46e5, #7c3aed);
            color: white;
            border-bottom-right-radius: 4px;
        }

        .bot-message {
            background: #f1f5f9;
            color: #334155;
            border-bottom-left-radius: 4px;
        }

        .chat-input-container {
            padding: 15px;
            background: white;
            border-top: 1px solid #e2e8f0;
        }

        .input-wrapper {
            display: flex;
            gap: 10px;
```

```
        }

        .chat-input {
            flex: 1;
            padding: 12px 16px;
            border: 2px solid #e2e8f0;
            border-radius: 25px;
            font-size: 14px;
            outline: none;
        }

        .chat-input:focus {
            border-color: #4f46e5;
        }

        .send-btn {
            width: 44px;
            height: 44px;
            border: none;
            border-radius: 50%;
            background: linear-gradient(135deg, #4f46e5, #7c3aed);
            color: white;
            cursor: pointer;
            font-size: 16px;
        }
    </style>
</head>
<body>

<div class="chatbot-container">
    <div class="chat-header">
        SCM Assistant
    </div>
    <div class="chat-messages" id="chatMessages"></div>
    <div class="chat-input-container">
        <div class="input-wrapper">
            <input type="text" id="chatInput" class="chat-input" placeholder="Type your message...">
            <button class="send-btn" onclick="sendMessage()">➤</button>
        </div>
    </div>
</div>

<script>
    const chatMessages = document.getElementById("chatMessages");
    const chatInput = document.getElementById("chatInput");

    function addMessage(text, isUser = false) {
        const msgDiv = document.createElement("div");
        msgDiv.classList.add("message");
        msgDiv.classList.add(isUser ? "user" : "bot");

        const content = document.createElement("div");
        content.classList.add("message-content");
        content.classList.add(isUser ? "user-message" : "bot-message");
        content.textContent = text;

        msgDiv.appendChild(content);
        chatMessages.appendChild(msgDiv);
        chatMessages.scrollTop = chatMessages.scrollHeight;
    }

    function sendMessage() {
```

```
    const question = chatInput.value.trim();
    if (!question) return;
    addMessage(question, true);
    chatInput.value = "";

    fetch("http://127.0.0.1:3000/ask/?q=" + encodeURIComponent(question))
      .then(res => res.json())
      .then(data => {
        addMessage(data.response || "No response from server.");
      })
      .catch(err => {
        console.error(err);
        addMessage("Error connecting to chatbot.", false);
      });
  }

  chatInput.addEventListener("keypress", function (e) {
    if (e.key === "Enter") {
      sendMessage();
    }
  });
</script>
</body>
</html>
```

6. SOFTWARE TEST DOCUMENT

6.1 System Overview

The system is a web-based platform integrated with an AI-powered chatbot designed for efficient supply chain operations, including warehouse management, order processing, forecasting, and analytics. **Version to be tested**: v1.0 (Prototype), v1.5 (Beta), v2.0 (Final)

6.2 Test Approach

The project uses **black-box testing** for functional features and **white-box testing** for backend logic and AI predictions. Tools include:

1. Manual UI testing
2. PHPUnit for backend
3. frontend responsiveness
4. Rasa for chatbot behavior testing

Each module (Products, Orders, Chatbot, Forecasting) is tested independently, then integrated testing is conducted. Time estimates and test activities follow agile sprints.

6.3Test Plan

6.3.1 Scope: Warehouse, Order Processing, Forecasting, AI Chatbot

6.3.2 Resources:

1. 2 testers (Hashir Ahmed, Abdullah)
2. Tools: PhpMyAdmin, Selenium Testing, XAMPP

6.3.3 Schedule:

1. Week 1: Module-wise testing
2. Week 2: Integration and regression
3. Week 3: User Acceptance Testing (UAT)

6.4 Features to be Tested

Feature	Test Case ID	Software Version
AI Chatbot FAQs	TC-1.0-0002	v1.0
Forecasting Accuracy	TC-2.0-0004	v2.0
User Login/Registration	TC-1.0-0005	v1.0

Table 2 Features to be Tested

6.5 Features not to be Tested

1. Third-party email notifications (planned for later release)
2. Multi-language chatbot responses (deferred due to time constraints)

6.6 Testing Tools and Environment

6.6.1 Tools:

XAMPP, PhpMyAdmin, Selenium Testing

6.6.2 Environment:

1. OS: Windows 10
2. Browsers: Chrome, Firefox
3. Hardware: Intel i5+, 8GB RAM

6.6.3 Staffing:

1. Ubaid – chatbot and backend testing
2. Hashir Ahmed – UI, Database and frontend testing

6.7 Test Cases

Test Case ID	Feature Tested	Version	Test Description	Expected Result	Actual Result	Status
TC-1.0-001	Products Auto-Update	v1.0	Reduce stock when product is purchased	Products decrease by ordered quantity	Product updated	Pass
TC-1.0-002	AI Chatbot FAQs	v1.0	Respond to "Where is my order?"	Accurate order status is returned	Response accurate	Pass
TC-2.0-003	Forecasting Accuracy	v2.0	Predict product demand using XGBoost	Predictions match 85%+ historical accuracy	88% accuracy achieved	Pass
TC-1.0-004	User Login/Registration	v1.0	User login and session creation	Redirect to dashboard after login	Login successful	Pass
TC-1.0-005	Invalid Login Attempt	v1.0	Enter wrong password	Display error message, deny access	Access denied, alert	Pass
TC-1.5-006	Add Product by Admin	v1.5	Add product with details	Product saved to DB, visible in UI	Product added	Pass
TC-2.0-007	Chatbot Misunderstood Input	v2.0	Enter unclear query like “what is supplychain?”	Chatbot asks for clarification	Clarification prompted	Pass
TC-1.5-008	Order Invoice Generation	v1.5	Create invoice on order confirmation	PDF invoice generated	Invoice created	Pass
TC-2.0-009	Role-Based Access Control	v2.0	Prevent retailer from accessing admin settings	Access denied with error	Access denied	Pass

Table 3 Test Cases

6.8 Case-n (use a unique ID of the form TC for this heading)

6.8.1 Purpose

6.8.1.1 Software Version: v1.0

6.8.1.1.1 Test Item:

Product Management and Warehouse Update Module.

6.8.1.1.2 Objective:

Verify that the supply chain system correctly updates the quantity of a product in a specific warehouse when modified by an admin (e.g., restocking).

6.8.1.1.3 SRS Reference:

1. Section 3.3.1 – Functional Requirements → Product and Warehouse Management
2. Section 3.9.3 – Inventory Data → Product Stock Levels by Warehouse

6.8.2 Inputs

Input Type	Value	Description
Product ID	P001	Example: "Milk" in system
Warehouse ID	WH-01	Main Karachi warehouse
Previous Quantity	120 units	Product quantity before update
Updated Quantity	170 units	Admin-entered new quantity after stock replenishment
User Role	Admin	Only admin can modify product quantities in warehouses

Table 4 Inputs

6.8.3 Expected Outputs & Pass/Fail criteria

Output	Expected Value	Pass/Fail Criteria
New Quantity in Warehouse	170 units	Must match exactly in both UI and database
UI Update Confirmation	Product updated successfully	Shown immediately after submission
Database Sync	Quantity value = 170	Verified in MySQL via phpMyAdmin
Response Time	< 2 seconds	From submit click to UI update

Table 5 Expected Outputs & Pass/Fail Criteria

6.8.4 Test Procedure

1. **Login** as an **admin** through the web interface.
2. Go to **Product Management > Warehouses**.
3. Select Product ID **P001** stored in Warehouse **WH-01**.
4. Verify the current stock quantity (e.g., **120 units**).
5. Click **Edit** and change quantity to **170 units**.
6. Click **Save**.

7. Confirm:
    1. Updated value appears on-screen instantly.
    2. “Product updated successfully” message is displayed.
    3. Check the backend (phpMyAdmin) for Product P001 in WH-01 → Quantity = 170.
    4. Measure time taken (should be < 2s).
8. Log all results.
9. **Status** = Pass if all validations succeed; Fail otherwise.

6.9 Testing Code:

6.9.1: SCM_Localhost Project Testing code:

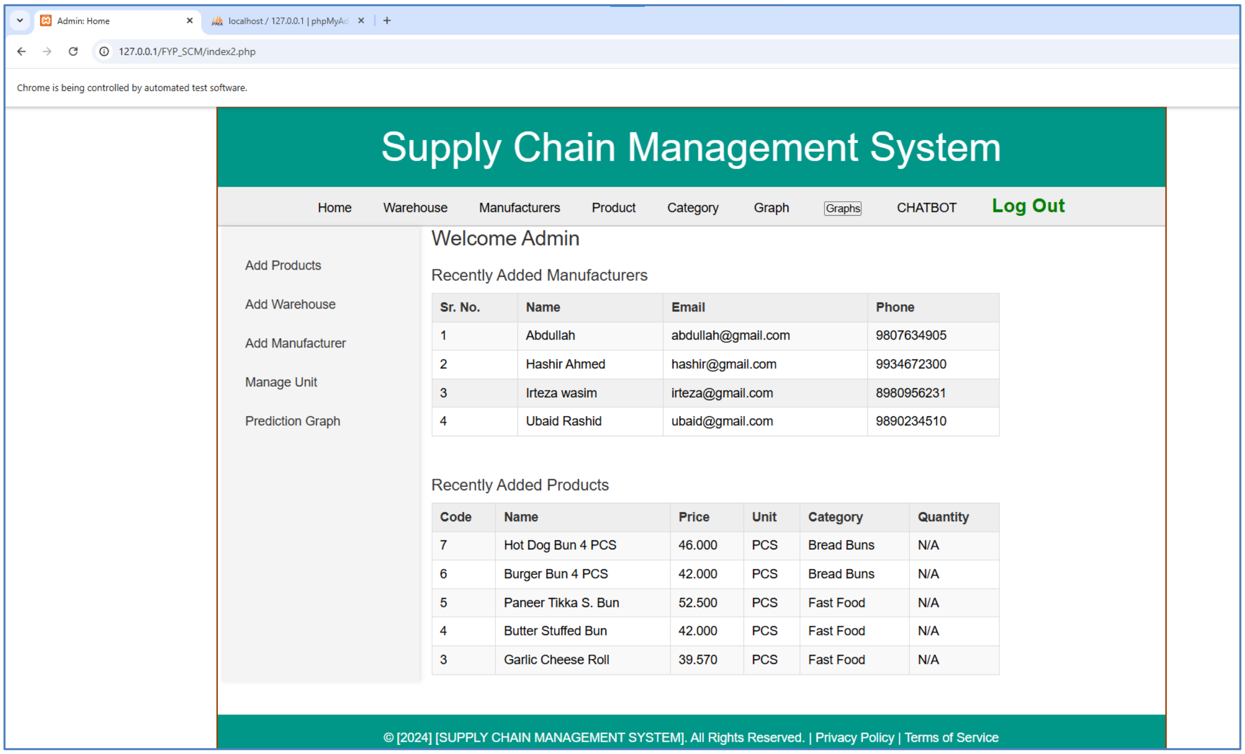
```
package pro;
  import org.openqa.selenium.By;
  import org.openqa.selenium.JavascriptExecutor;
  import org.openqa.selenium.WebDriver;
  import org.openqa.selenium.WebElement;
  import org.openqa.selenium.chrome.ChromeDriver;
  import org.testng.annotations.AfterClass;
  import org.testng.annotations.BeforeClass;
  import org.testng.annotations.Test;
  import java.util.ArrayList;

  public class XamppLocalhostTest {
    private WebDriver driver;
    @BeforeClass
    public void setUp() {
      driver = new ChromeDriver();
    driver.manage().window().maximize();
    }
    @Test(priority = 1)
    public void testFYP_SCMProject() throws InterruptedException {
      String projectUrl = "http://127.0.0.1/FYP_SCM/login.php";
      driver.get(projectUrl);
      System.out.println("=== FYP_SCM Project ===");
      System.out.println("Page title: " + driver.getTitle());
      Thread.sleep(2000);
      try {
        WebElement header = driver.findElement(By.id("main-header"));
        System.out.println("Header text: " + header.getText());
      } catch (Exception e) {
        System.out.println("Element with ID 'main-header' not found.");
      }
    }
    @Test(priority = 2)
    public void testPhpMyAdminInNewTab() throws InterruptedException {
      String phpmyadminUrl = "http://localhost/phpmyadmin/";
      ((JavascriptExecutor) driver).executeScript("window.open('" + phpmyadminUrl + "',
'_blank');");
      ArrayList<String> tabs = new ArrayList<>(driver.getWindowHandles());
      driver.switchTo().window(tabs.get(1));
      System.out.println("=== phpMyAdmin ===");
      System.out.println("Page title: " + driver.getTitle());
      Thread.sleep(2000);
      try {
        WebElement usernameField = driver.findElement(By.id("input_username"));
        WebElement passwordField = driver.findElement(By.id("input_password"));
        WebElement loginButton = driver.findElement(By.id("input_go"));

        usernameField.sendKeys("root");
        passwordField.sendKeys("");  // Default no password
        loginButton.click();

        System.out.println("phpMyAdmin login submitted.");
      } catch (Exception e) {
        System.out.println("phpMyAdmin login elements not found or already logged in.");
      }
      System.out.println("Current URL after phpMyAdmin login: " + driver.getCurrentUrl());
      driver.switchTo().window(tabs.get(0));
      System.out.println("Back to FYP_SCM tab. Title: " + driver.getTitle());
    }
  @AfterClass
  public void tearDown() {
    // Close the browser and clean up resources
    if (driver != null) {
      driver.quit();
    }
  }
```

6.9.2: OUTPUT UI Testing



Chrome is being controlled by automated test software.

Supply Chain Management System

Welcome Admin

Recently Added Manufacturers

Sr. No.	Name	Email	Phone
1	Abdullah	abdullah@gmail.com	9807634905
2	Hashir Ahmed	hashir@gmail.com	9934672300
3	Irteza wasim	irteza@gmail.com	8980956231
4	Ubaid Rashid	ubaid@gmail.com	9890234510

Recently Added Products

Code	Name	Price	Unit	Category	Quantity
7	Hot Dog Bun 4 PCS	46.000	PCS	Bread Buns	N/A
6	Burger Bun 4 PCS	42.000	PCS	Bread Buns	N/A
5	Paneer Tikka S. Bun	52.500	PCS	Fast Food	N/A
4	Butter Stuffed Bun	42.000	PCS	Fast Food	N/A
3	Garlic Cheese Roll	39.570	PCS	Fast Food	N/A

© [2024] [SUPPLY CHAIN MANAGEMENT SYSTEM]. All Rights Reserved. | Privacy Policy | Terms of Service

Figure 16 Output UI Testing

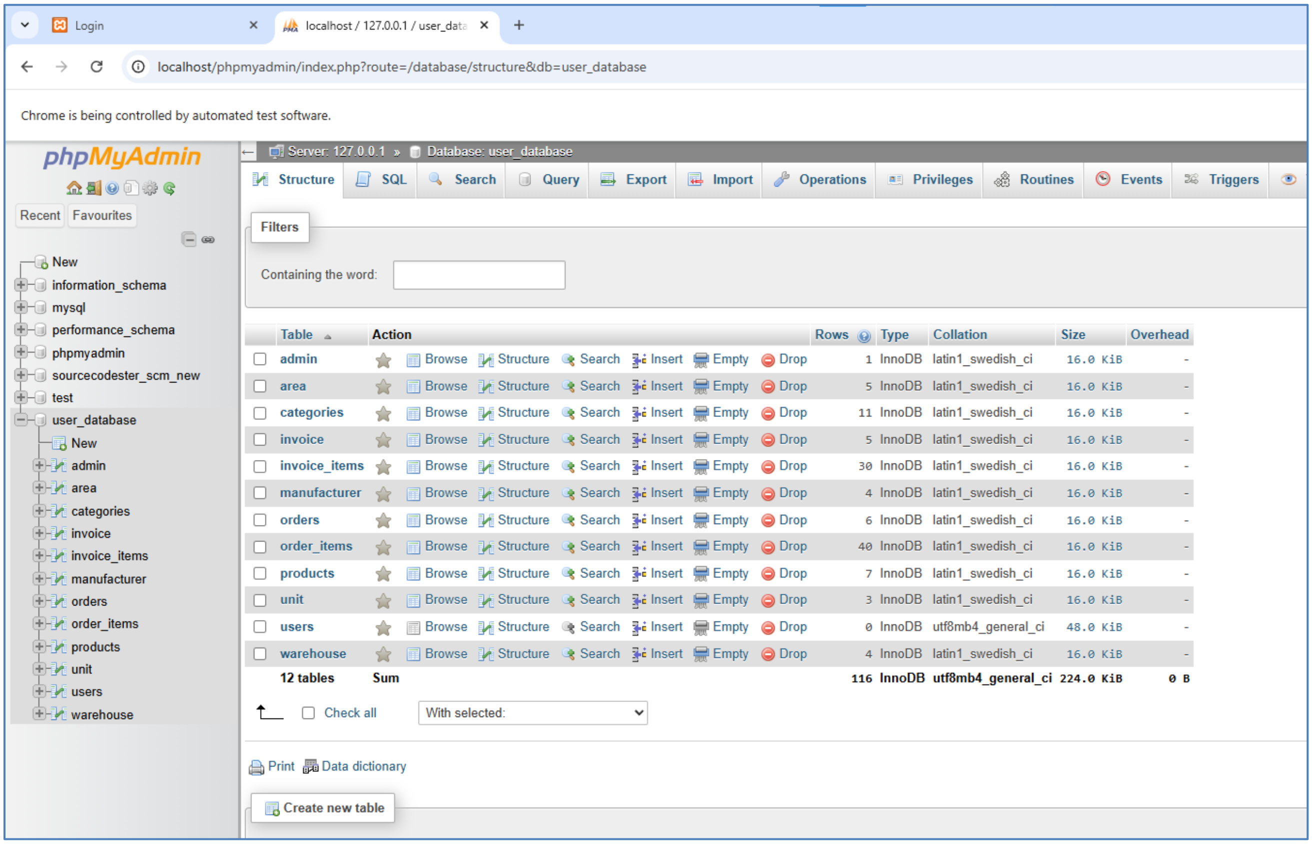
6.9.1.1 Figure 16 Output UI Testing Explanation:

The image shows an admin dashboard for a Supply Chain Management System and perform testing as well. **It displays:**

1. A welcome message for the admin.
2. A table listing Recently Added Manufacturers with their names, emails, and phone numbers.
3. A table for Recently Added Products, showing product codes, names, prices, units, categories, and quantities.
4. Options for adding products and managing other aspects of the system, along with a Log Out button.

This interface allows the admin to efficiently manage manufacturers and products.

6.9.2 Database Testing:



localhost/phpmyadmin/index.php?route=/database/structure&db=user_database

Chrome is being controlled by automated test software.

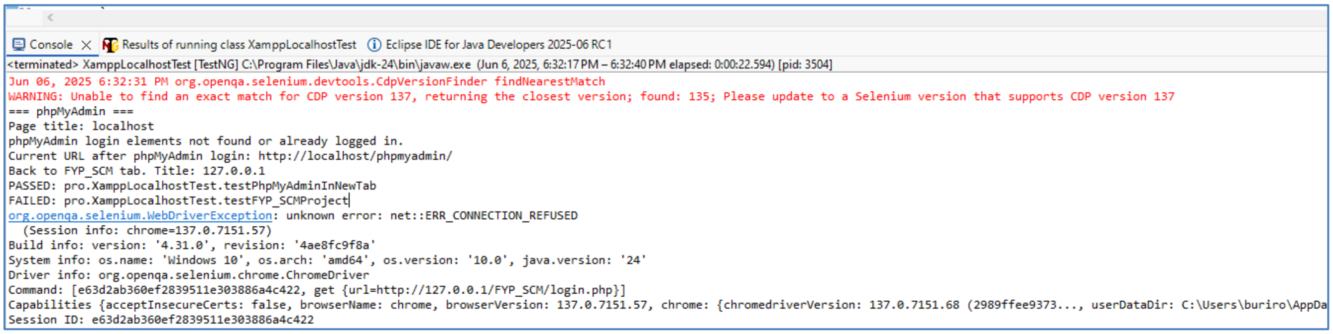
	Table	Action							Rows	Type	Collation	Size	Overhead
□	admin	☆	Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	area	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	categories	☆	Browse	Structure	Search	Insert	Empty	Drop	11	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	invoice	☆	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	invoice_items	☆	Browse	Structure	Search	Insert	Empty	Drop	30	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	manufacturer	☆	Browse	Structure	Search	Insert	Empty	Drop	4	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	orders	☆	Browse	Structure	Search	Insert	Empty	Drop	6	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	order_items	☆	Browse	Structure	Search	Insert	Empty	Drop	40	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	products	☆	Browse	Structure	Search	Insert	Empty	Drop	7	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	unit	☆	Browse	Structure	Search	Insert	Empty	Drop	3	InnoDB	latin1_swedish_ci	16.0 KiB	-
□	users	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
□	warehouse	☆	Browse	Structure	Search	Insert	Empty	Drop	4	InnoDB	latin1_swedish_ci	16.0 KiB	-
	12 tables	Sum							116	InnoDB	utf8mb4_general_ci	224.0 KiB	0 B

Figure 17 Database Testing

6.9.2.1 Figure 17 Database Testing Explanation:

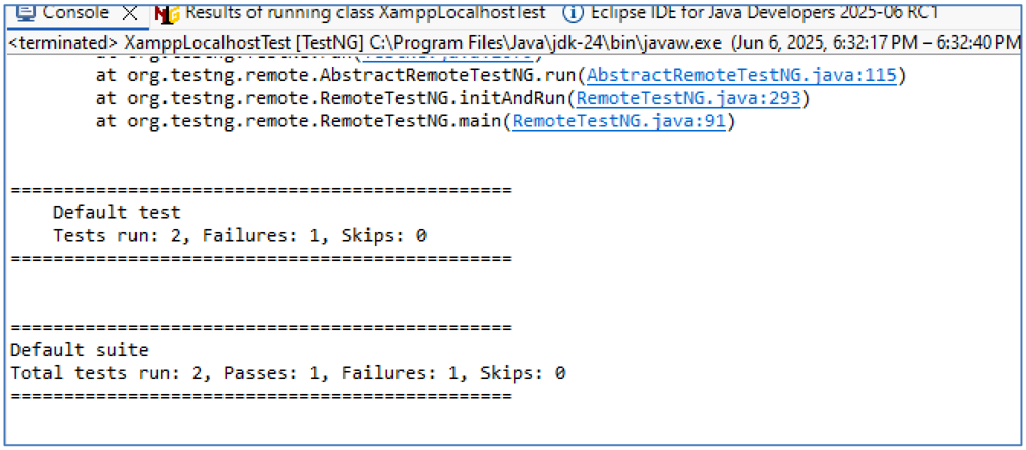
The image displays a phpMyAdmin interface showcasing a database structure. It lists various tables, such as **categories**, **products**, **orders**, and **warehouse**, along with options to view their structure, search, and drop tables. Information about the number of rows and storage size for each table is also provided. This interface is used for managing and testing the database.

6.9.3 Test Cases Check Pass or Fails:


Console
Results of running class XamppLocalhostTest
Eclipse IDE for Java Developers 2025-06 RC1

```
<terminated> XamppLocalhostTest [TestNG] C:\Program Files\Java\jdk-24\bin\javaw.exe (Jun 6, 2025, 6:32:17 PM – 6:32:40 PM elapsed: 0:00:22.594) [pid: 3504]
Jun 06, 2025 6:32:31 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 137, returning the closest version; found: 135; Please update to a Selenium version that supports CDP version 137
=== phpMyAdmin ===
Page title: localhost
phpMyAdmin login elements not found or already logged in.
Current URL after phpMyAdmin login: http://localhost/phpmyadmin/
Back to FYP_SCM tab. Title: 127.0.0.1
PASSED: pro.XamppLocalhostTest.testPhpMyAdminInNewTab
FAILED: pro.XamppLocalhostTest.testFYP_SCMProject
org.openqa.selenium.WebDriverException: unknown error: net::ERR_CONNECTION_REFUSED
  (Session info: chrome=137.0.7151.57)
Build info: version: '4.31.0', revision: '4ae8fc9f8a'
System info: os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Command: [e63d2ab360ef2839511e303886a4c422, get {url=http://127.0.0.1/FYP_SCM/login.php}]
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 137.0.7151.57, chrome: {chromedriverVersion: 137.0.7151.68 (2989ffee9373..., userDataDir: C:\Users\buriro\AppDa
Session ID: e63d2ab360ef2839511e303886a4c422
```

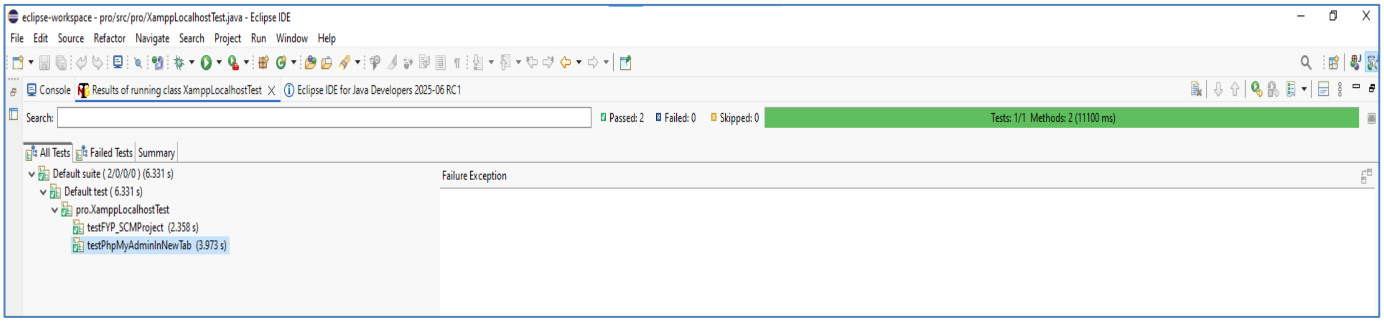
Figure 18 Test Check Pass or Fail


Console
Results of running class XamppLocalhostTest
Eclipse IDE for Java Developers 2025-06 RC1

```
<terminated> XamppLocalhostTest [TestNG] C:\Program Files\Java\jdk-24\bin\javaw.exe (Jun 6, 2025, 6:32:17 PM – 6:32:40 PM
        at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:115)
        at org.testng.remote.RemoteTestNG.initAndRun(RemoteTestNG.java:293)
        at org.testng.remote.RemoteTestNG.main(RemoteTestNG.java:91)


===============================================
    Default test
    Tests run: 2, Failures: 1, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 1, Failures: 1, Skips: 0
===============================================
```


eclipse-workspace - pro/src/pro/XamppLocalhostTest.java - Eclipse IDE
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Console
Results of running class XamppLocalhostTest
Eclipse IDE for Java Developers 2025-06 RC1
Search:
All Tests
Failed Tests
Summary
Failure Exception

```
Passed: 2   Failed: 0   Skipped: 0   Tests: 1/1  Methods: 2 (11100 ms)

Default suite ( 2/0/0/0 ) (6.331 s)
  Default test ( 6.331 s)
    pro.XamppLocalhostTest
      testFYP_SCMProject  (2.358 s)
      testPhpMyAdminInNewTab  (3.973 s)
```

All Tests Are Passed Successfully.

6.9.3.1 <u>Figure 18 Test Cases check Pass or Fail Explanation:</u>

The images show the results of a testing session for a web application, confirming that all tests passed successfully. Despite the initial connection error noted, there were no failures in any tests. The results indicate that the functionality of the system is intact, as all tests executed without issues. Overall, the testing session was successful, with all test cases passing.

7. APPENDIX A: TEST LOGS

A test log is used by the test team to record what occurred during test execution.

7.1 Log

7.1.1 Test Results

Log ID	Date/Time	Tester	Test Case	Module Tested	Inputs	Observed Result	Output Location	Status
TL-0001	2025-05-24 15:10	Hashir Ahmed	TC-1001	Product-Warehouse Quantity Update	Product ID = P001, WH = WH-01, Qty: 120 → 170	Quantity updated; reflected in UI & phpMyAdmin	Admin Panel, phpMyAdmin	Pass
TL-0002	2025-05-25 10:20	M. Irteza Waseem	TC-1002	AI Chatbot Response	User input: “Where is my stock?”	Chatbot gave accurate ETA & product location	Chatbot UI	Pass
TL-0003	2025-05-26 14:45	Ubaid Rashid	TC-1003	Forecasting Model Accuracy	Product: Rice, Data: 90 days demand history	Forecast accuracy = 88%, within acceptable threshold	Forecast Report, Console	Pass
TL-0004	2025-05-27 11:30	Abdullah	TC-1004	Role-Based Access Restriction	attempted admin access	Access denied message correctly shown	Web UI	Pass

Table 6 Test Results

7.1.2 Incident Report

Incident ID	Linked Log ID	Issue Summary	Cause	Impact	Resolution	Resolved
TIR-0001	TL-0001	UI didn’t auto-refresh after product update	Missing refresh callback in frontend	Minor UI delay	Added window.location.reload() after successful update	Yes
TIR-0002	TL-0002	Chatbot couldn’t handle irregular query phrasing	No fallback intent configured in Rasa	Moderate	Configured fallback response for unrecognized inputs	Yes
TIR-0003	TL-0003	CPU spike during model training	XGBoost using all threads during training	Low	Thread limit set; training scheduled during off-hours	Yes

Table 7 Incident Report

7.1.3 Test Summary:

Test Case ID	Feature/Module	Version	Result	Incident Report	Fixed
TC-1001	Product Quantity Update (Admin)	v1.0	Pass	TIR-0001	Yes
TC-1002	AI Chatbot Query Response	v1.0	Pass	TIR-0002	Yes
TC-1003	Demand Forecast Accuracy	v2.0	Pass	TIR-0003	Yes
TC-1004	User Access Restriction	v2.0	Pass	None	N/A

Table 8 Test Summary

8. CONCLUSIONS AND FURTHER WORK

8.1 Conclusion:

This project aimed to develop an intelligent and user-friendly **Supply Chain Management System (SCM)** integrated with an **AI-based chatbot** to assist users in navigating managing product data, forecasting demand, and enhancing overall operational efficiency.

Through the design, development, and testing phases, the team built a functional system that centralizes warehouse management while offering intelligent communication through an AI chatbot. The chatbot was able to handle basic supply chain queries such as product availability and order inquiries, reducing the load on manual support systems. **The system also featured modules for:**

1. Admin product and warehouse management.
2. Demand forecasting using machine learning (XGBoost).
3. Real-time product updates and validation.
4. Role-based user access to separate admin.

This project reflects practical knowledge in full-stack development, AI integration, and testing discipline. We ensured that each phase from requirement analysis to test logging followed a documented and systematic approach, helping us understand the real-world software development lifecycle more deeply.

8.2 Further Work and Enhancements:

While the current version of the project meets its core objectives, there are many possible areas for improvement and future expansion:

8.2.1 Multi-language Chatbot Support

Implement multilanguage capabilities to serve users from different regions, especially useful for diverse retail or supplier networks.

8.2.2 Integration with Logistics APIs

Connect the system with courier and logistics APIs for real-time shipment updates, and automated invoice generation.

8.2.3 Advanced Forecasting Features

Incorporate external factors (seasonality, promotions, holidays) into the forecasting model for improved accuracy.

8.2.4 Notification System

Introduce automated alerts via email or SMS for critical actions like low stock, order placement, or delayed delivery.

8.2.5 Data Analytics Dashboard

Add visual dashboards for analytics such as monthly sales trends, supplier performance, or return rates to assist decision-making.

9. REFERENCES

[1]. Chopra and Meindl (2019) provide a comprehensive look at supply chain management, covering strategy, planning, and operations in their widely used textbook *Supply Chain Management: Strategy, Planning, and Operation* (7th edition), published by Pearson.

[2]. In *Essentials of Supply Chain Management* (4th edition), Hugos (2018) offers a practical and accessible overview of supply chain principles, making it a valuable resource for both students and professionals. This book is published by Wiley.

[3]. Russell and Norvig's *Artificial Intelligence: A Modern Approach* (4th edition, 2021) remains a foundational text in the field of AI. Published by Pearson, it provides deep insights into the concepts and techniques that underpin modern artificial intelligence systems.

[4]. Goyal (2014), in his book *Enterprise Resource Planning: A Managerial Perspective*, presents ERP systems from a business and managerial point of view. The book, published by McGraw-Hill Education, highlights how ERP tools support organizational processes.

[5]. Chen and Guestrin (2016) introduce XGBoost, a powerful and scalable machine learning algorithm, in their influential paper *XGBoost: A Scalable Tree Boosting System*, presented at the 22nd ACM SIGKDD Conference.

[6]. Chopra and Sodhi (2004) explore how companies can manage risks to prevent disruptions in their supply chains in the article *Managing Risk to Avoid Supply-Chain Breakdown*, published in the *MIT Sloan Management Review*. Available online.

[7]. Lee (2002) discusses how supply chain strategies should be aligned with different levels of product uncertainty in his influential piece *Aligning Supply Chain Strategies with Product Uncertainties*, featured in the *California Management Review*.

[8]. In their 2013 study, Delen and Hardgrave highlight how predictive analytics can significantly enhance supply chain decision-making in their paper *Predictive Analytics in Supply Chain Management*, published in *Decision Support Systems*.

[9]. Adamopoulos and Todri (2020) examine the growing influence of artificial intelligence in supply chains, discussing both practical applications and broader implications in *AI-Driven Supply Chain: Applications and Implications*, published in the *Journal of Business Research*.

[10]. Ghosh (2022) looks at how AI-powered chatbots are being integrated into Enterprise Resource Planning (ERP) systems to streamline processes and improve user interaction in his article published in the *International Journal of Emerging Trends in Computer Science*.

[11]. Priyanka and Mishra (2021) focus on the impact of artificial intelligence and natural language processing (NLP) in delivering improved customer experiences through chatbots. Their findings are shared in the *Journal of AI Research and Applications*.

[12]. Nair and Mallya (2021) explore how chatbots and AI-driven agents can optimize various aspects of supply chain management in their study published in the *International Journal of Supply Chain Management*.

[13]. Al-Tahat (2020) presents a machine learning-based approach to demand forecasting in supply chains in the article *Forecasting Demand in the Supply Chain Using Machine Learning Algorithms*, published in the *Journal of Manufacturing Systems*.

10. SDG FEEDBACK

10.1 SDGs Feedback by Student:

SDG Name	Targets	Feedback by Student	Proof with page Number
SDG 9: Industry, Innovation and Infrastructure	9.1, 9.4, 9.c	Our AI-based supply chain project boosts innovation by using forecasting tools and chatbot automation to enhance logistics processes and infrastructure.	Page 3, 15, 113
SDG 12: Responsible Consumption and Production	12.2, 12.5	The system supports sustainability by reducing inventory waste and enabling smarter, more efficient business decisions.	Page 46–47, 113

Table 9 SDGs Feedback by Student

10.2 CEP Feedback by Students

Attributes	Feedback by Student	Proof with page Number
Preamble	Our Work contributes to sustainability by applying automation and AI to streamline supply chain operations and reduce unnecessary resource use.	Page 3, 15, 113
Range of Conflicting Requirements	During development, we had to manage trade-offs between usability, system performance, security, and technical scalability.	Page 46–47, 25
Depth of Analysis Required	Creating this solution involved deep exploration of AI models, chatbot behavior, and logistics workflows.	Page 19, 52–54
Depth of Knowledge Required	We used a range of skills from backend coding to AI algorithm tuning and user interface planning.	Page 24–30
Familiarity of Issue	We already had foundational knowledge of SCM challenges, which we expanded by applying them practically.	Page 15–16
Extent of Applicable Codes	Development followed secure and organized practices using modular code in PHP, Python, and MySQL.	Page 91–99
Extent of Stakeholder Involvement and Level of Conflicting Requirements	Regular interaction with stakeholders helped prioritize features and address trade-offs during design.	Page 25–26
Consequences	The system reduces operational bottlenecks, cuts down on errors, and enhances overall supply performance.	Page 46–47, 113
Interdependence	All system components—forecasting, chatbot, and inventory are interconnected, so stable integration was crucial.	Page 59, 103–104

Table 10 CEP Feedback by Student

10.3 CEAs Feedback by Students:

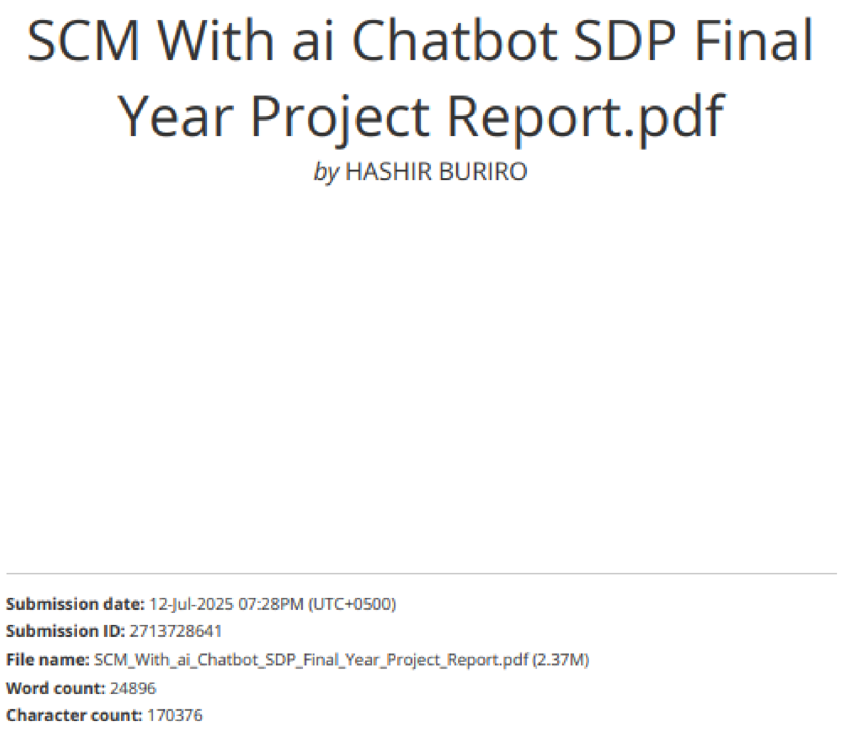
Attributes	Feedback by Student	Proof with page Number
Preamble	This final-year work brought together AI and logistics knowledge to build a meaningful, practical tool.	Page 15–16, 113
Range of Resources	We utilized programming tools, testing environments, and frameworks such as XAMPP, Selenium, and Rasa NLU.	Page 20, 104
Level of Interaction	Collaboration within our group and feedback from faculty shaped the final structure of our system.	Page 25–26
Innovation	Real-time chatbot features and predictive analysis added a fresh, tech-driven edge to our SCM approach	Page 46,113
Consequences to Society and Environment	Our tool encourages sustainable logistics practices by optimizing resource use.	Page 3,113
Familiarity	While we were familiar with supply chain basics, this project helped us apply and deepen that understanding.	Page 16, 113

Table 11 SDGs Feedback by Student

10.4 PBL Feedback by Student

Feedback by Student	Proof with page Number
Working on this project helped us experience real-world software development. We addressed an industry problem, applied AI-based solutions, and gained hands-on experience from concept through implementation. It strengthened our technical knowledge, teamwork, and problem-solving abilities.	Page 15–30, 46, 113

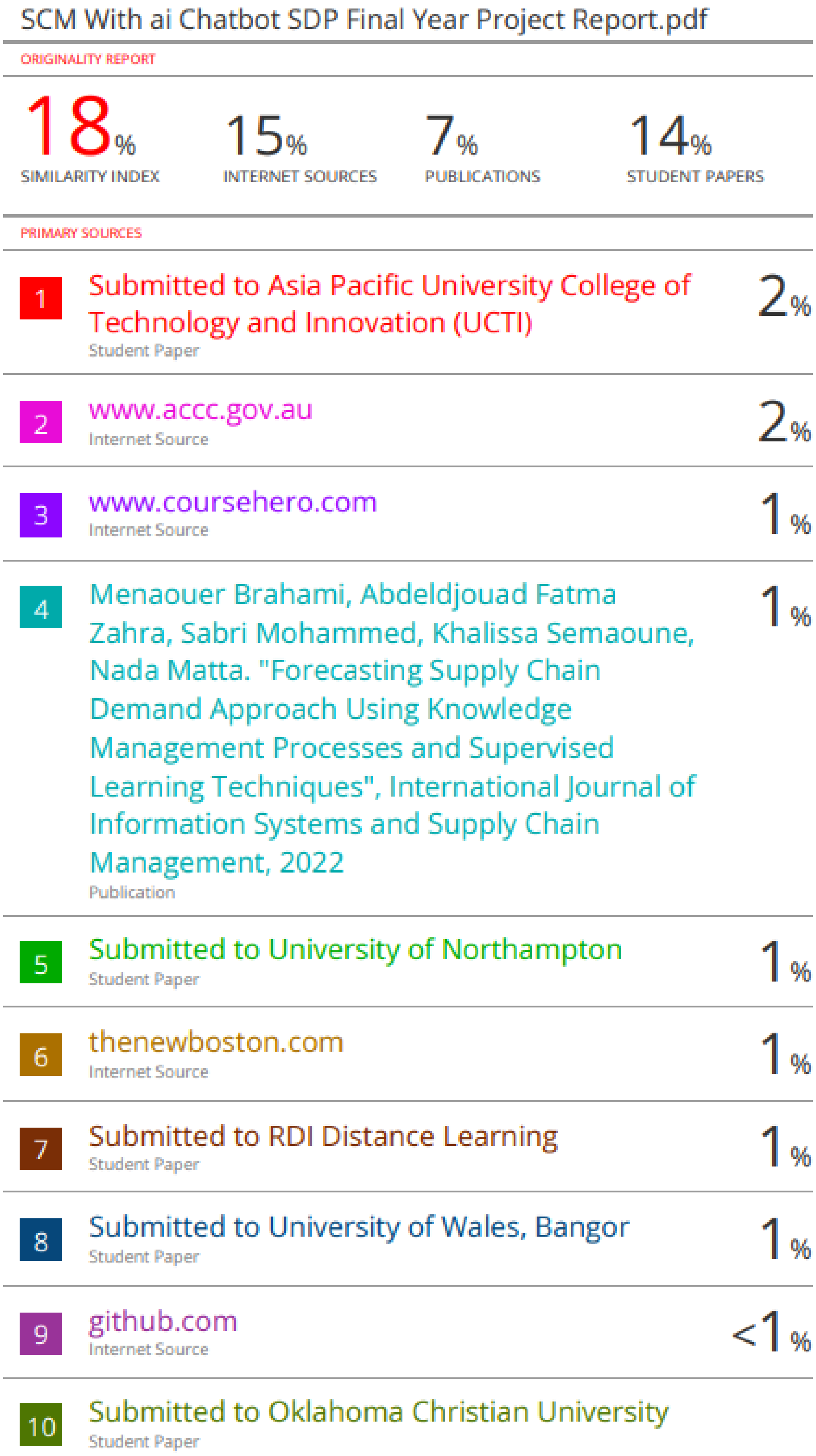
Table 12 PBL Feedback by Student



SCM With ai Chatbot SDP Final Year Project Report.pdf

by HASHIR BURIRO

Submission date: 12-Jul-2025 07:28PM (UTC+0500)
Submission ID: 2713728641
File name: SCM_With_ai_Chatbot_SDP_Final_Year_Project_Report.pdf (2.37M)
Word count: 24896
Character count: 170376



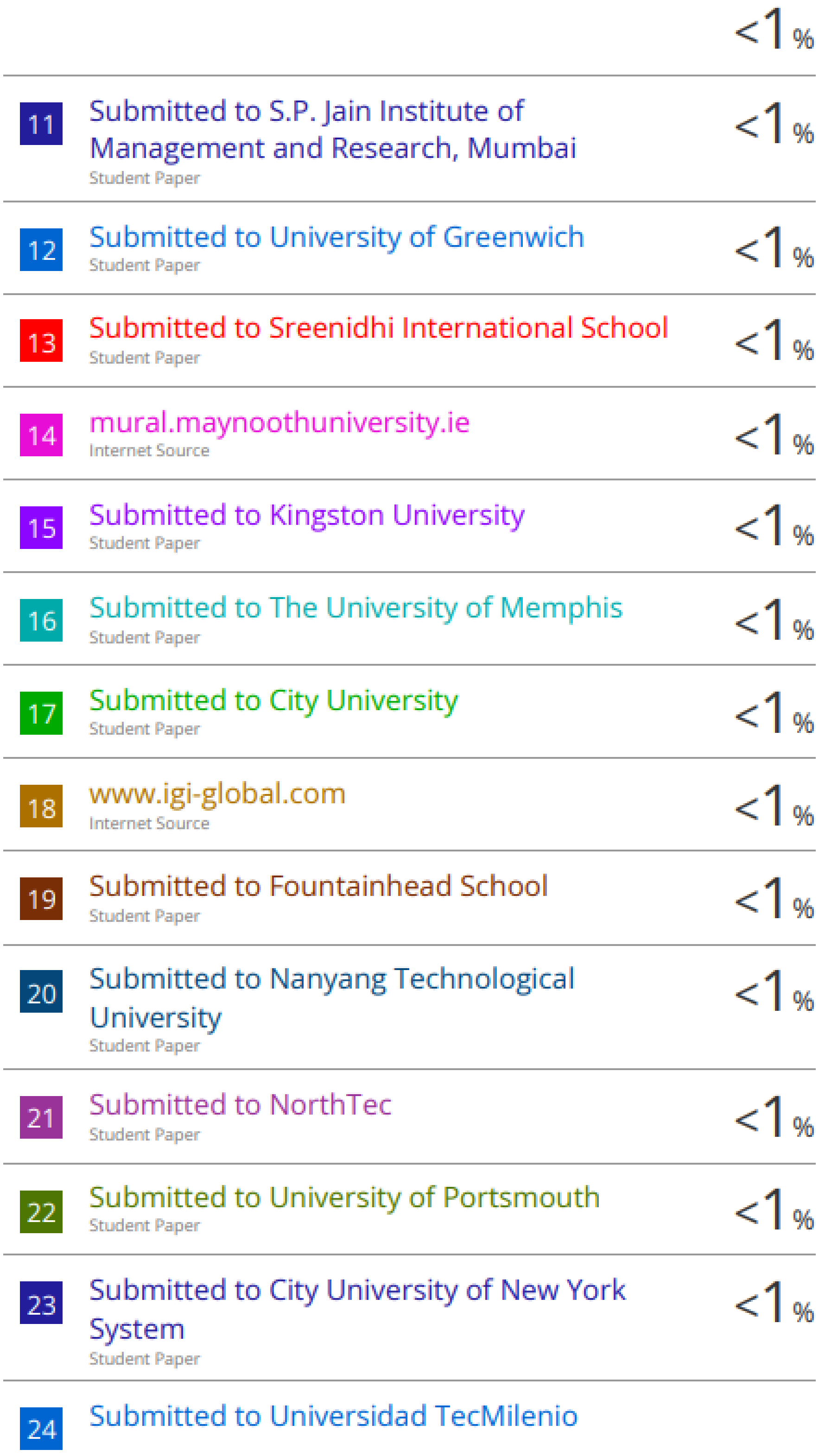
SCM With ai Chatbot SDP Final Year Project Report.pdf

ORIGINALITY REPORT

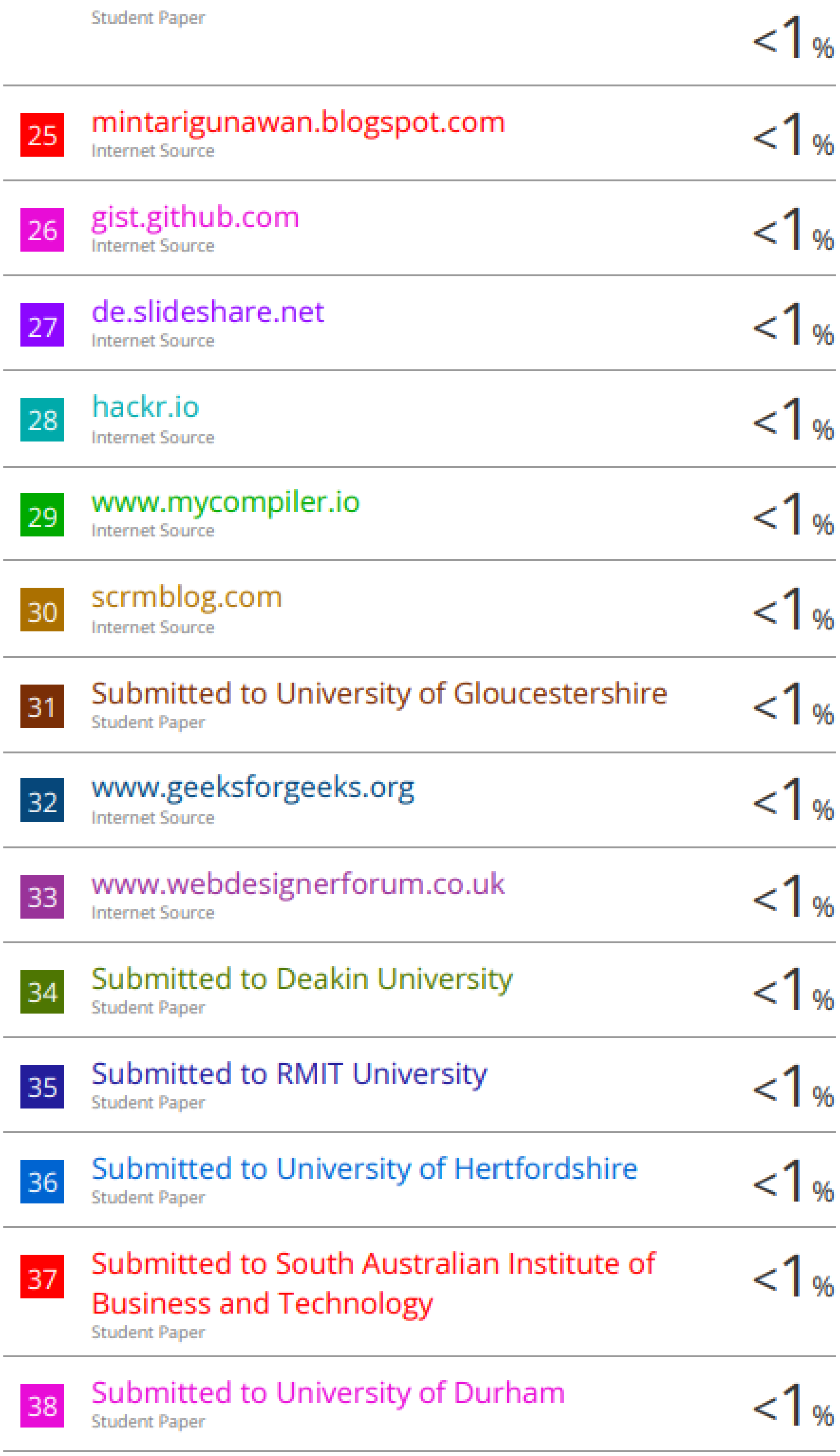
18%	15%	7%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

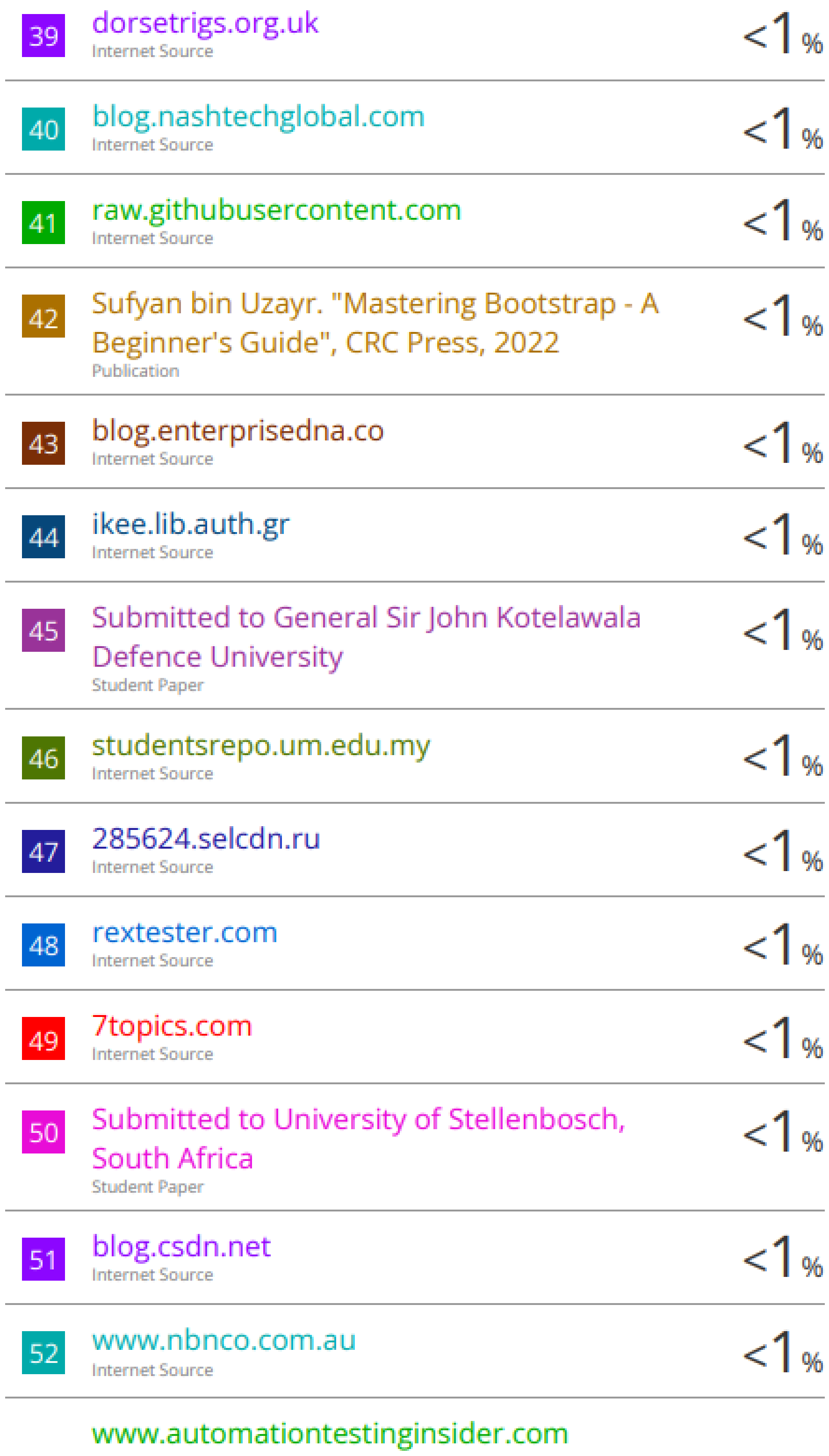
1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	2%
2	www.accc.gov.au Internet Source	2%
3	www.coursehero.com Internet Source	1%
4	Menaouer Brahami, Abdeldjouad Fatma Zahra, Sabri Mohammed, Khalissa Semaoune, Nada Matta. "Forecasting Supply Chain Demand Approach Using Knowledge Management Processes and Supervised Learning Techniques", International Journal of Information Systems and Supply Chain Management, 2022 Publication	1%
5	Submitted to University of Northampton Student Paper	1%
6	thenewboston.com Internet Source	1%
7	Submitted to RDI Distance Learning Student Paper	1%
8	Submitted to University of Wales, Bangor Student Paper	1%
9	github.com Internet Source	<1%
10	Submitted to Oklahoma Christian University Student Paper	



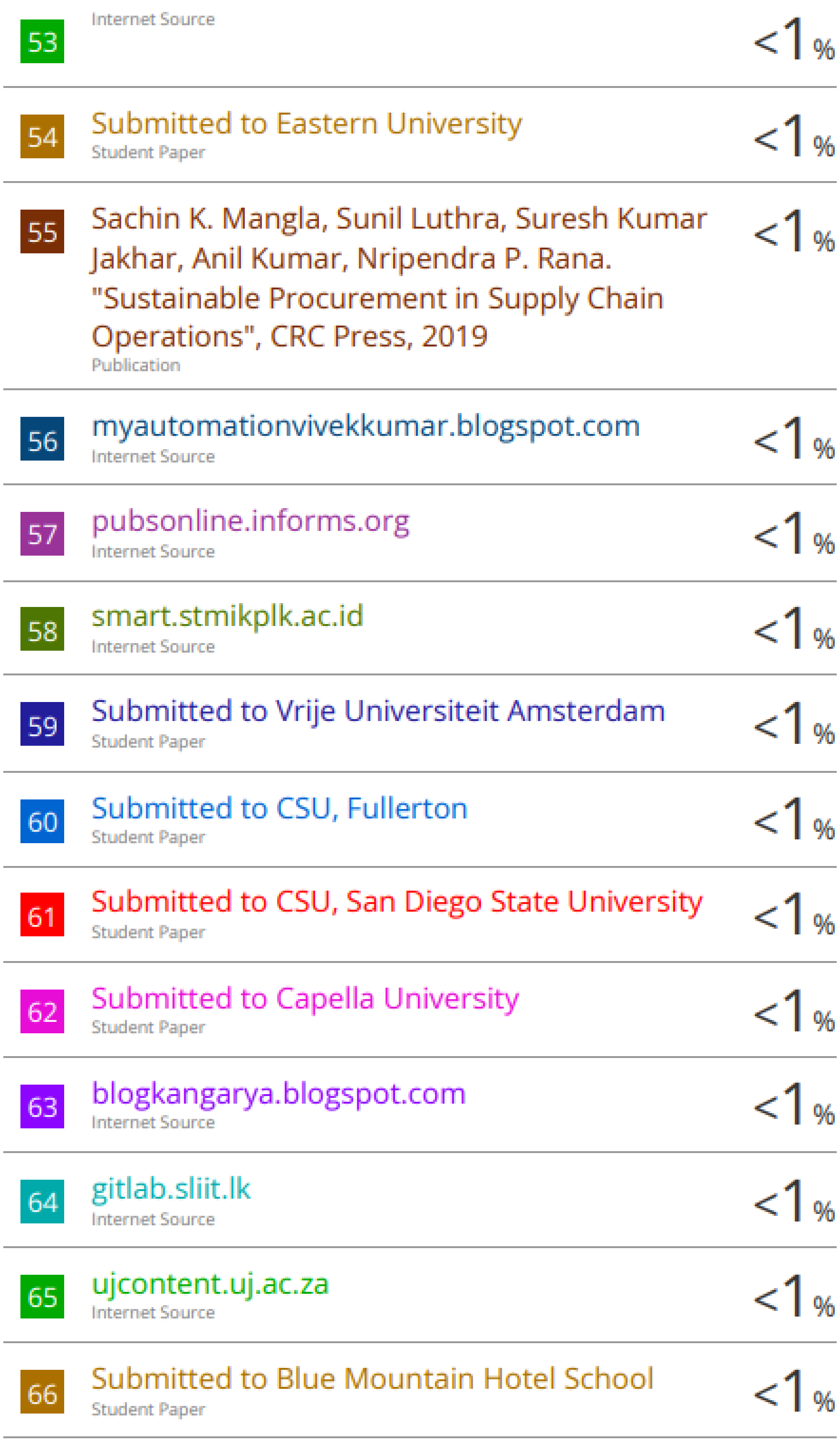
		<1 %
11	Submitted to S.P. Jain Institute of Management and Research, Mumbai Student Paper	<1 %
12	Submitted to University of Greenwich Student Paper	<1 %
13	Submitted to Sreenidhi International School Student Paper	<1 %
14	mural.maynoothuniversity.ie Internet Source	<1 %
15	Submitted to Kingston University Student Paper	<1 %
16	Submitted to The University of Memphis Student Paper	<1 %
17	Submitted to City University Student Paper	<1 %
18	www.igi-global.com Internet Source	<1 %
19	Submitted to Fountainhead School Student Paper	<1 %
20	Submitted to Nanyang Technological University Student Paper	<1 %
21	Submitted to NorthTec Student Paper	<1 %
22	Submitted to University of Portsmouth Student Paper	<1 %
23	Submitted to City University of New York System Student Paper	<1 %
24	Submitted to Universidad TecMilenio	



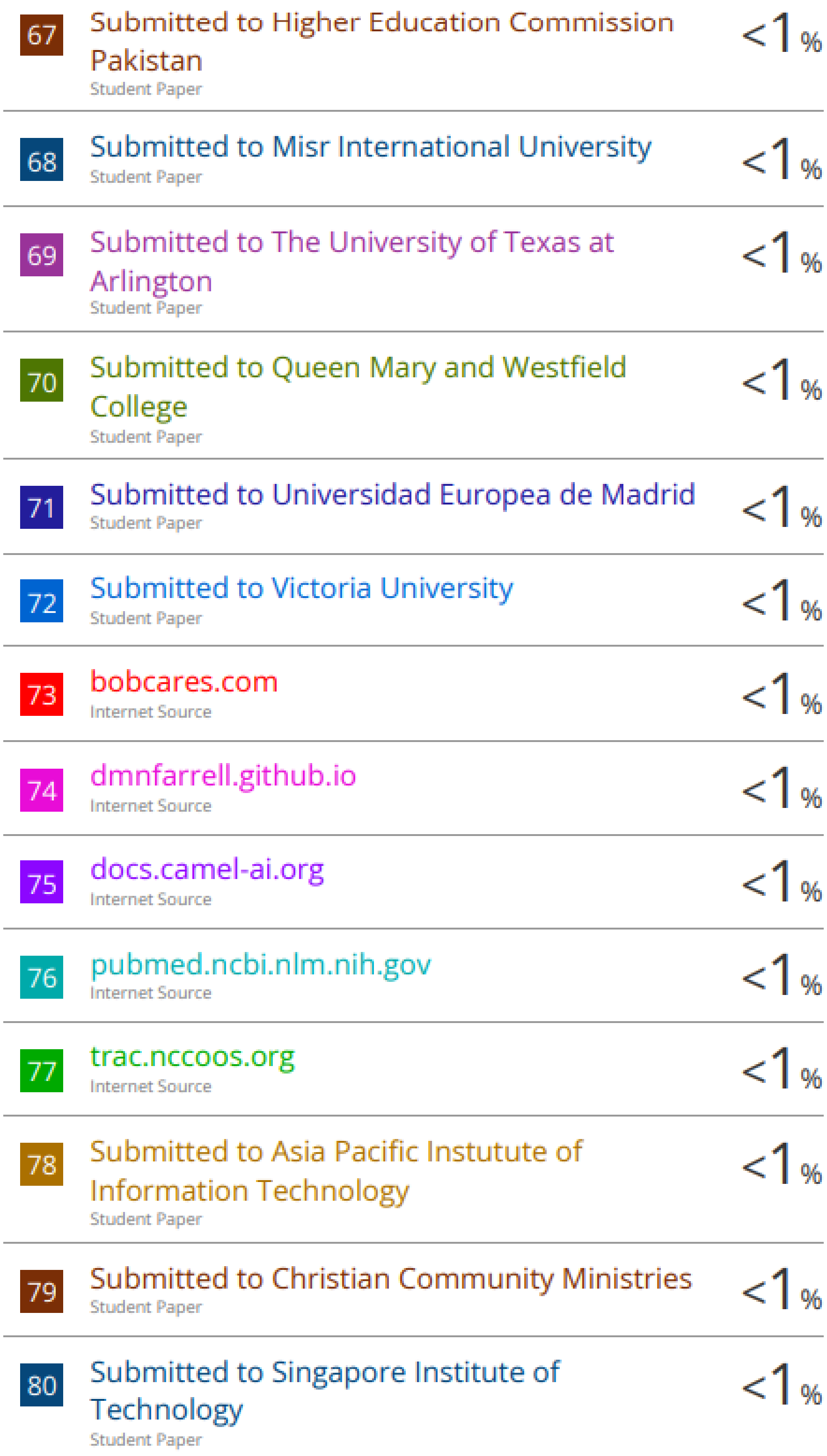
	Student Paper	<1 %
25	mintarigunawan.blogspot.com Internet Source	<1 %
26	gist.github.com Internet Source	<1 %
27	de.slideshare.net Internet Source	<1 %
28	hackr.io Internet Source	<1 %
29	www.mycompiler.io Internet Source	<1 %
30	scrmblog.com Internet Source	<1 %
31	Submitted to University of Gloucestershire Student Paper	<1 %
32	www.geeksforgeeks.org Internet Source	<1 %
33	www.webdesignerforum.co.uk Internet Source	<1 %
34	Submitted to Deakin University Student Paper	<1 %
35	Submitted to RMIT University Student Paper	<1 %
36	Submitted to University of Hertfordshire Student Paper	<1 %
37	Submitted to South Australian Institute of Business and Technology Student Paper	<1 %
38	Submitted to University of Durham Student Paper	<1 %



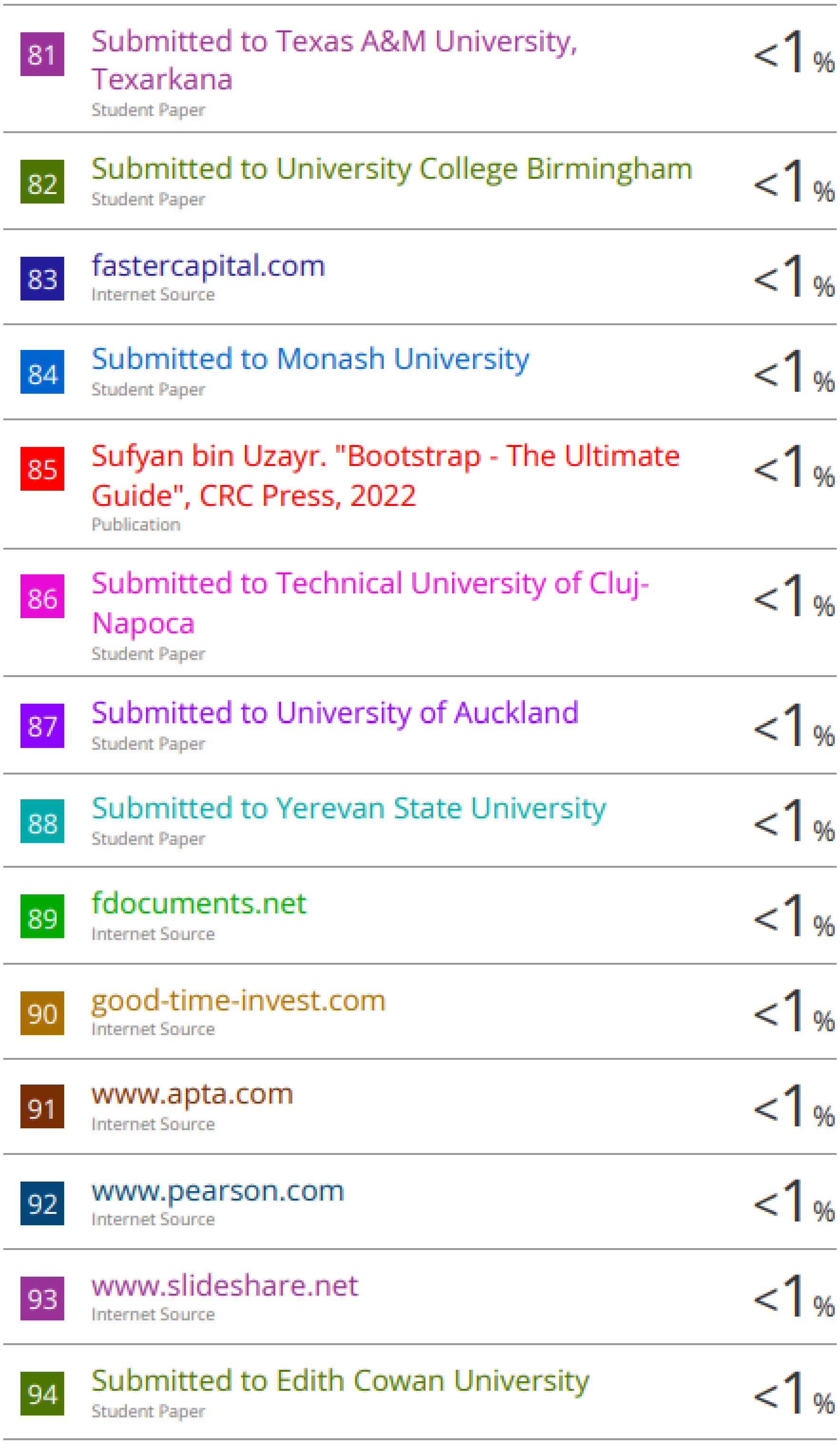
39	dorsetrigs.org.uk Internet Source	<1 %
40	blog.nashtechglobal.com Internet Source	<1 %
41	raw.githubusercontent.com Internet Source	<1 %
42	Sufyan bin Uzayr. "Mastering Bootstrap - A Beginner's Guide", CRC Press, 2022 Publication	<1 %
43	blog.enterprisedna.co Internet Source	<1 %
44	ikee.lib.auth.gr Internet Source	<1 %
45	Submitted to General Sir John Kotelawala Defence University Student Paper	<1 %
46	studentsrepo.um.edu.my Internet Source	<1 %
47	285624.selcdn.ru Internet Source	<1 %
48	rextester.com Internet Source	<1 %
49	7topics.com Internet Source	<1 %
50	Submitted to University of Stellenbosch, South Africa Student Paper	<1 %
51	blog.csdn.net Internet Source	<1 %
52	www.nbnco.com.au Internet Source	<1 %
	www.automationtestinginsider.com	



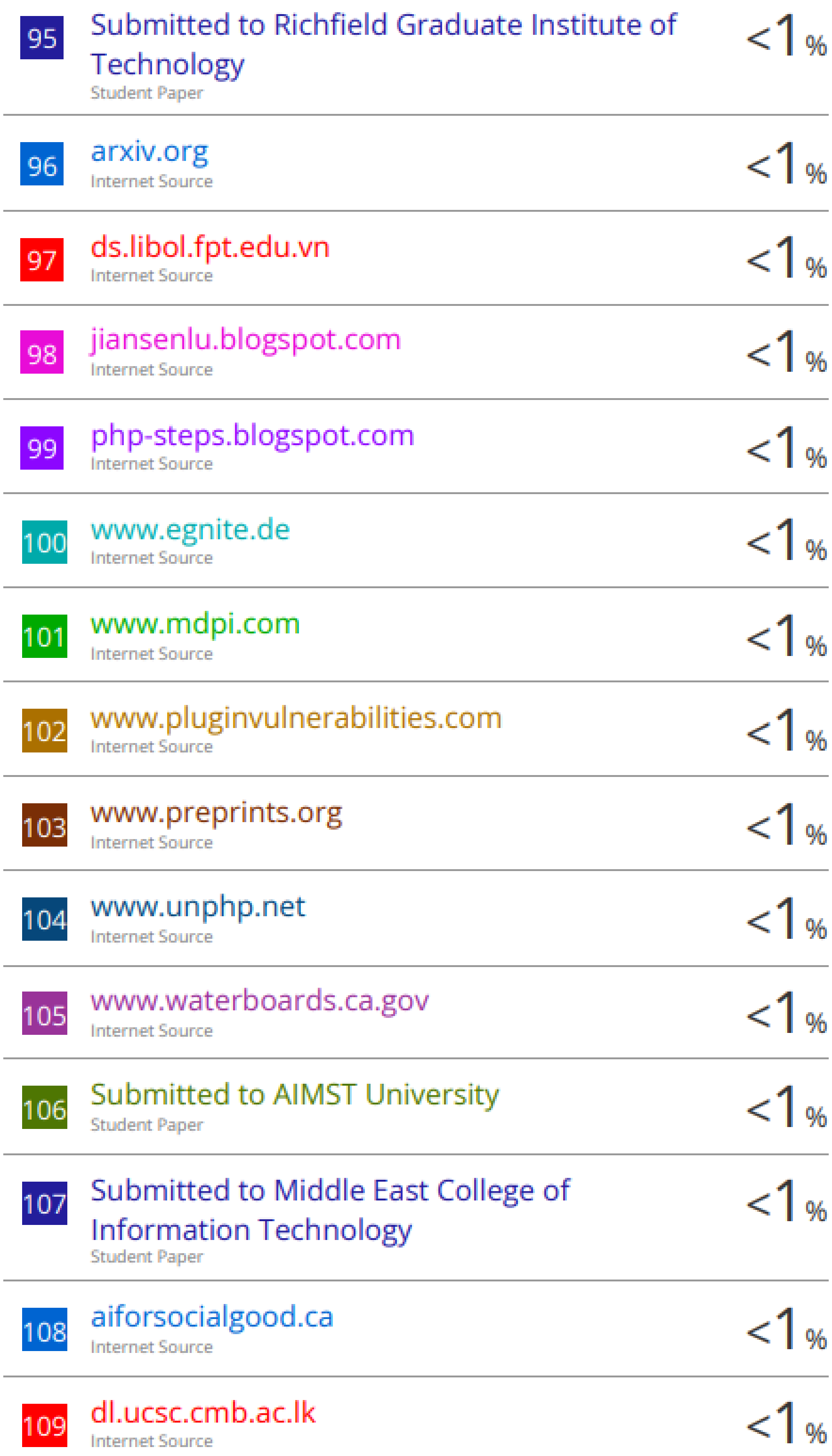
53	Internet Source	<1 %
54	Submitted to Eastern University Student Paper	<1 %
55	Sachin K. Mangla, Sunil Luthra, Suresh Kumar Jakhar, Anil Kumar, Nripendra P. Rana. "Sustainable Procurement in Supply Chain Operations", CRC Press, 2019 Publication	<1 %
56	myautomationvivekkumar.blogspot.com Internet Source	<1 %
57	pubsonline.informs.org Internet Source	<1 %
58	smart.stmikplk.ac.id Internet Source	<1 %
59	Submitted to Vrije Universiteit Amsterdam Student Paper	<1 %
60	Submitted to CSU, Fullerton Student Paper	<1 %
61	Submitted to CSU, San Diego State University Student Paper	<1 %
62	Submitted to Capella University Student Paper	<1 %
63	blogkangarya.blogspot.com Internet Source	<1 %
64	gitlab.sliit.lk Internet Source	<1 %
65	ujcontent.uj.ac.za Internet Source	<1 %
66	Submitted to Blue Mountain Hotel School Student Paper	<1 %



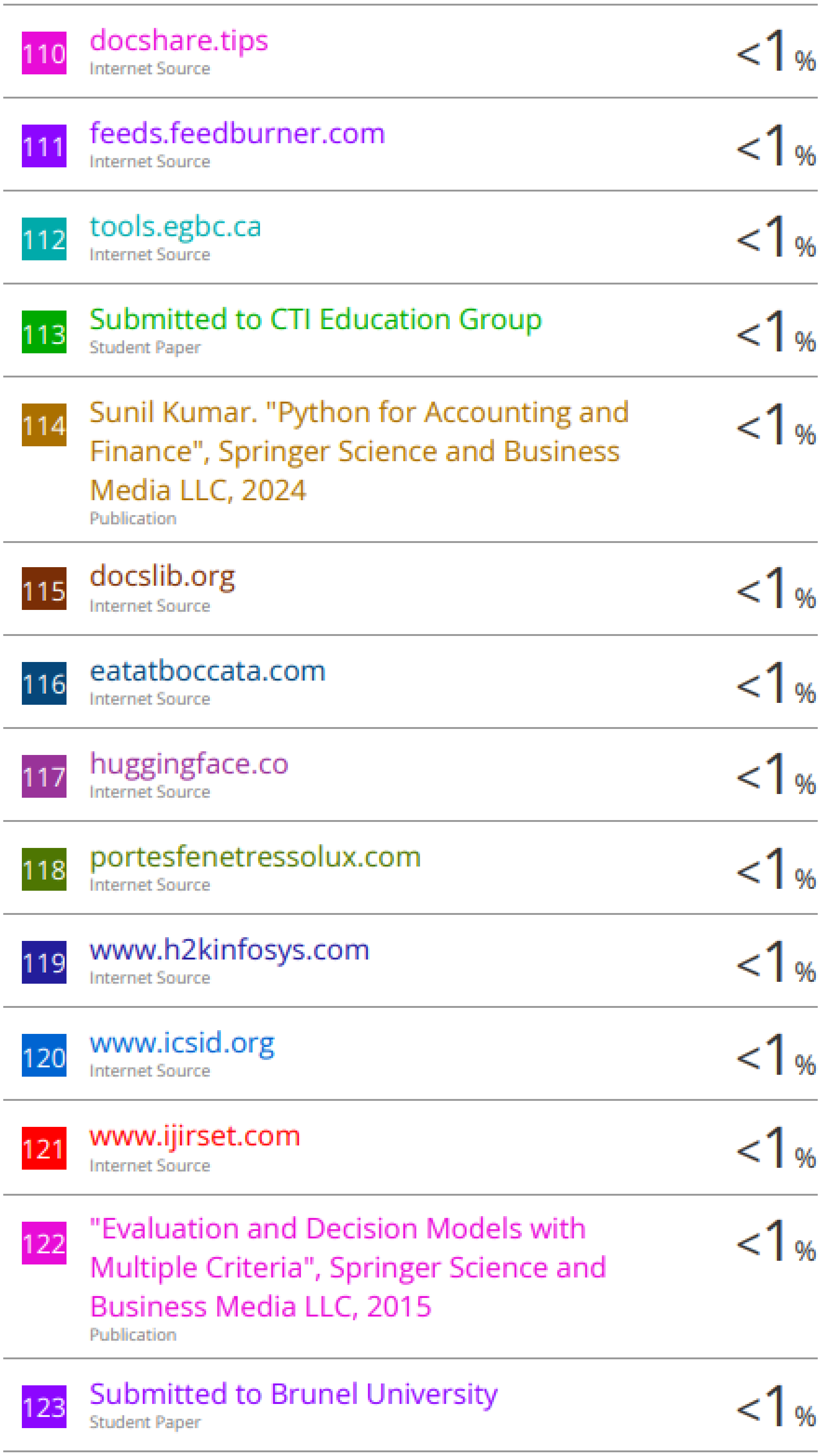
67	Submitted to Higher Education Commission Pakistan	Student Paper	<1 %
68	Submitted to Misr International University	Student Paper	<1 %
69	Submitted to The University of Texas at Arlington	Student Paper	<1 %
70	Submitted to Queen Mary and Westfield College	Student Paper	<1 %
71	Submitted to Universidad Europea de Madrid	Student Paper	<1 %
72	Submitted to Victoria University	Student Paper	<1 %
73	bobcares.com	Internet Source	<1 %
74	dmnfarrell.github.io	Internet Source	<1 %
75	docs.camel-ai.org	Internet Source	<1 %
76	pubmed.ncbi.nlm.nih.gov	Internet Source	<1 %
77	trac.nccoos.org	Internet Source	<1 %
78	Submitted to Asia Pacific Instutute of Information Technology	Student Paper	<1 %
79	Submitted to Christian Community Ministries	Student Paper	<1 %
80	Submitted to Singapore Institute of Technology	Student Paper	<1 %



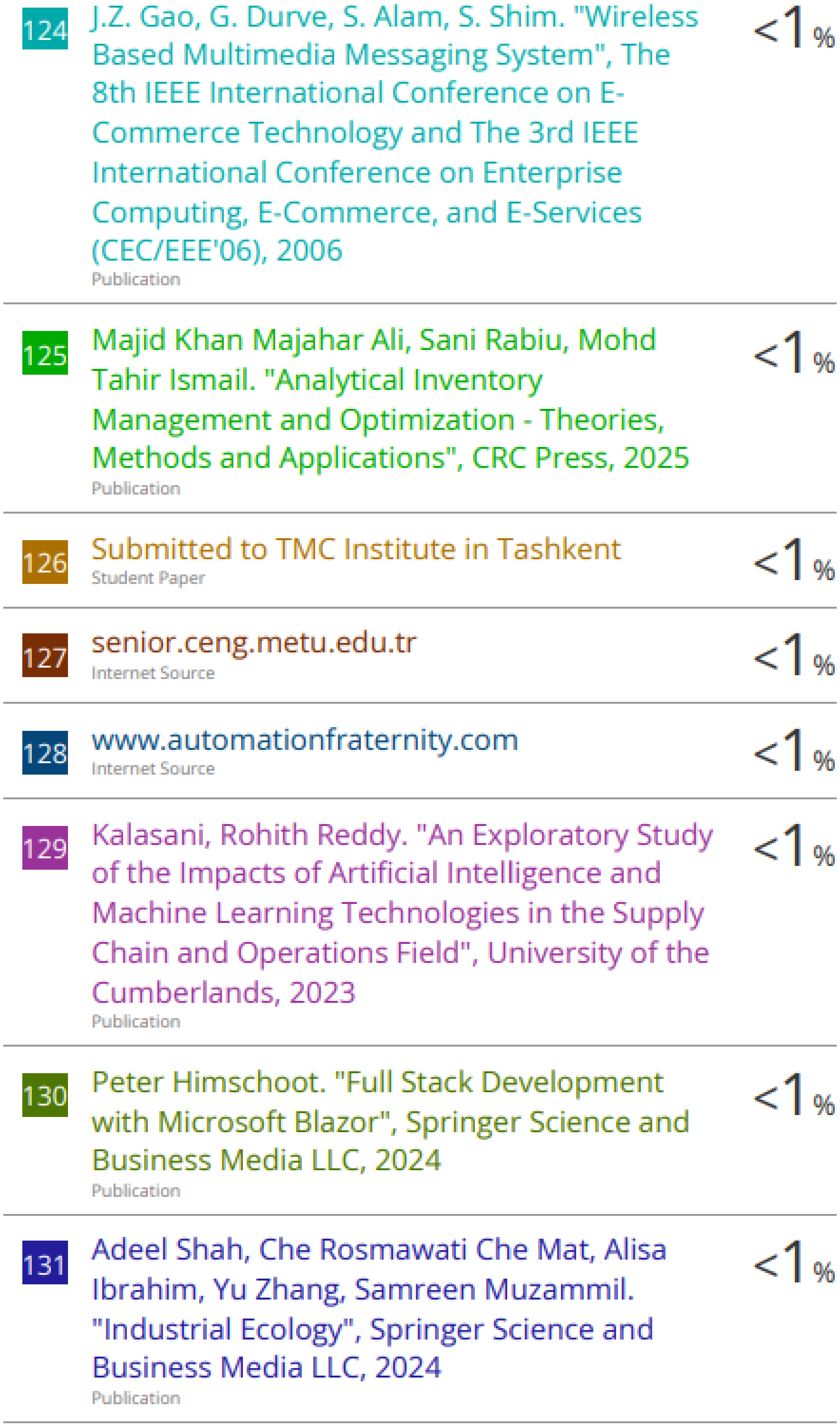
81	Submitted to Texas A&M University, Texarkana Student Paper	<1 %
82	Submitted to University College Birmingham Student Paper	<1 %
83	fastercapital.com Internet Source	<1 %
84	Submitted to Monash University Student Paper	<1 %
85	Sufyan bin Uzayr. "Bootstrap - The Ultimate Guide", CRC Press, 2022 Publication	<1 %
86	Submitted to Technical University of Cluj-Napoca Student Paper	<1 %
87	Submitted to University of Auckland Student Paper	<1 %
88	Submitted to Yerevan State University Student Paper	<1 %
89	fdocuments.net Internet Source	<1 %
90	good-time-invest.com Internet Source	<1 %
91	www.apta.com Internet Source	<1 %
92	www.pearson.com Internet Source	<1 %
93	www.slideshare.net Internet Source	<1 %
94	Submitted to Edith Cowan University Student Paper	<1 %



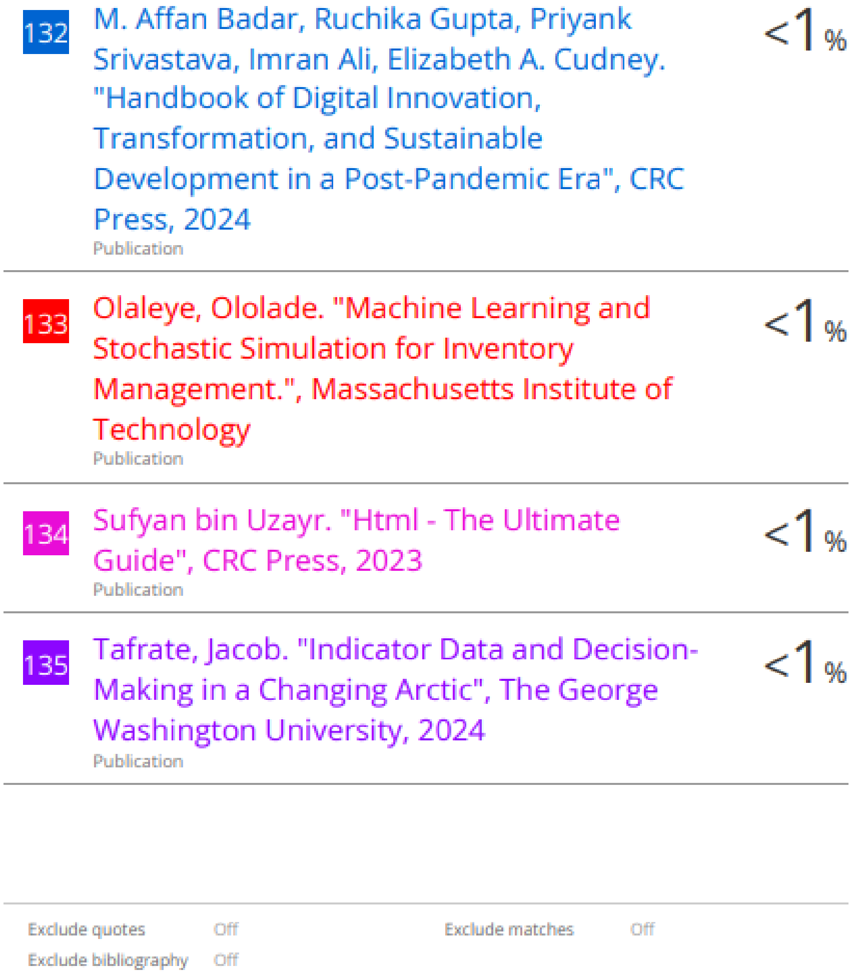
95	Submitted to Richfield Graduate Institute of Technology Student Paper	<1 %
96	arxiv.org Internet Source	<1 %
97	ds.libol.fpt.edu.vn Internet Source	<1 %
98	jiansenlu.blogspot.com Internet Source	<1 %
99	php-steps.blogspot.com Internet Source	<1 %
100	www.egnite.de Internet Source	<1 %
101	www.mdpi.com Internet Source	<1 %
102	www.pluginvulnerabilities.com Internet Source	<1 %
103	www.preprints.org Internet Source	<1 %
104	www.unphp.net Internet Source	<1 %
105	www.waterboards.ca.gov Internet Source	<1 %
106	Submitted to AIMST University Student Paper	<1 %
107	Submitted to Middle East College of Information Technology Student Paper	<1 %
108	aiforsocialgood.ca Internet Source	<1 %
109	dl.ucsc.cmb.ac.lk Internet Source	<1 %



110	docshare.tips Internet Source	<1 %
111	feeds.feedburner.com Internet Source	<1 %
112	tools.egbc.ca Internet Source	<1 %
113	Submitted to CTI Education Group Student Paper	<1 %
114	Sunil Kumar. "Python for Accounting and Finance", Springer Science and Business Media LLC, 2024 Publication	<1 %
115	docslib.org Internet Source	<1 %
116	eatatboccata.com Internet Source	<1 %
117	huggingface.co Internet Source	<1 %
118	portesfenetressolux.com Internet Source	<1 %
119	www.h2kinfosys.com Internet Source	<1 %
120	www.icsid.org Internet Source	<1 %
121	www.ijirset.com Internet Source	<1 %
122	"Evaluation and Decision Models with Multiple Criteria", Springer Science and Business Media LLC, 2015 Publication	<1 %
123	Submitted to Brunel University Student Paper	<1 %



124	J.Z. Gao, G. Durve, S. Alam, S. Shim. "Wireless Based Multimedia Messaging System", The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), 2006 Publication	<1 %
125	Majid Khan Majahar Ali, Sani Rabiu, Mohd Tahir Ismail. "Analytical Inventory Management and Optimization - Theories, Methods and Applications", CRC Press, 2025 Publication	<1 %
126	Submitted to TMC Institute in Tashkent Student Paper	<1 %
127	senior.ceng.metu.edu.tr Internet Source	<1 %
128	www.automationfraternity.com Internet Source	<1 %
129	Kalasani, Rohith Reddy. "An Exploratory Study of the Impacts of Artificial Intelligence and Machine Learning Technologies in the Supply Chain and Operations Field", University of the Cumberlands, 2023 Publication	<1 %
130	Peter Himschoot. "Full Stack Development with Microsoft Blazor", Springer Science and Business Media LLC, 2024 Publication	<1 %
131	Adeel Shah, Che Rosmawati Che Mat, Alisa Ibrahim, Yu Zhang, Samreen Muzammil. "Industrial Ecology", Springer Science and Business Media LLC, 2024 Publication	<1 %



132	M. Affan Badar, Ruchika Gupta, Priyank Srivastava, Imran Ali, Elizabeth A. Cudney. "Handbook of Digital Innovation, Transformation, and Sustainable Development in a Post-Pandemic Era", CRC Press, 2024 Publication	<1 %
133	Olaleye, Ololade. "Machine Learning and Stochastic Simulation for Inventory Management.", Massachusetts Institute of Technology Publication	<1 %
134	Sufyan bin Uzayr. "Html - The Ultimate Guide", CRC Press, 2023 Publication	<1 %
135	Tafrate, Jacob. "Indicator Data and Decision-Making in a Changing Arctic", The George Washington University, 2024 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		