UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

Preliminary Report

Stock Control System

Name - Apurva Vivek Kulkarni

Email Id – avk7@student.le.ac.uk
Student Id - 239064472

Project Supervisor: Anthony Conway
Principal Marker: Dr Paula Severi

Word Count: 2248

[Submission Date]

DECLARATION
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Apurva Vivek Kulkarni
Date: 24-02-2025

Contents

1. Aims and Objectives

The aim of this project is to develop a Stock Control System that efficiently manages and monitors computer component inventory in a warehouse. The system will enable real-time tracking, location identification, stock assembly process management, automated reporting, and invoicing. By incorporating computer services like maintenance, system design, software development, software installation, and disaster recovery, the system will provide an integrated IT management solution.
The project seeks to streamline stock operations, reduce inefficiencies, enhance inventory tracking accuracy, and optimize warehouse workflow through a scalable and secure web application.

1.2 Objectives

To achieve the project aims, the following objectives have been identified:
The **Stock Control System** aims to streamline and optimize **inventory management for computer components** in a warehouse. The system ensures **real-time stock tracking, efficient stock movement monitoring, automated alerts, and accurate invoicing**. By providing an intuitive interface and role-based access control, it enhances user experience and security. The system will also include **reporting and analytics tools** to help businesses make data-driven decisions regarding stock levels, demand forecasting, and replenishment. Furthermore, it integrates **computer services** such as maintenance, system design, software development, installation, and recovery. This will help warehouse managers efficiently manage IT-related inventory and ensure smooth operations. In this system, we are solving different problems affecting direct sales management and purchase management [1]. The system will also Implement **automated stock alerts**, ensuring that stock levels are maintained within optimal limits and reducing the risk of shortages or overstocking.

6. Develop a Centralized Inventory Management System

A centralized inventory system ensures that all stock-related data is stored and managed in one place, providing real-time visibility of stock levels. This prevents overstocking (which leads to wastage) and understocking (which causes delays). It allows warehouse managers to efficiently manage the inflow and outflow of computer components, ensuring that they have an accurate database of stock availability, locations, and transactions.

6. Implement Real-time Stock Movement and Location Tracking

This feature enables real-time tracking of stock movement within the warehouse. Each item is assigned a unique identifier (such as a barcode or QR code) and its location is updated whenever it is moved or used in stock assembly. This reduces misplacement of components, speeds up retrieval, and ensures better inventory control. It also enhances the efficiency of warehouse operations by optimizing the way items are stored and accessed.

6. Integrate Automated Alerts and Notifications

To ensure that stock levels remain optimal, the system will have an automated alert mechanism that notifies the warehouse manager when stock falls below a predefined

threshold. Notifications can be sent via email, SMS, or dashboard alerts, allowing for timely stock replenishment.

4. Develop Role-based Authentication and Access Control

Security is a crucial aspect of any inventory management system. The system will incorporate role-based access control (RBAC), which ensures that only authorized personnel can perform certain actions. For instance:

a. **Admins** will have full access to stock control, invoicing, and reporting.
b. **Warehouse Staff** will be able to update stock levels and track inventory movement.
c. **Finance Teams** will manage invoices and payments.

5. Provide Reporting and Analytics Tools

To help businesses make informed decisions, the system will include advanced reporting and analytics features. Warehouse managers can generate stock movement reports, sales analytics, and demand forecasting reports. These insights will help in:

a. Identifying fast-moving and slow-moving items.
b. Predicting future stock demand using historical trends.
c. Generating monthly and annual reports for financial and operational planning. The reports can be exported in multiple formats (PDF, CSV, Excel) and visually represented using graphs and charts. Data visualization offers a powerful tool to help simplify and communicate complex financial information and data to ease its understanding by various consumers of the data such as investors [5].

6. Implement an Invoice Management Module

A built-in invoice management module will streamline sales and purchase transactions. This feature will allow warehouse managers to:

a. Generate invoices for stock sold or purchased.
b. Store past invoices for record-keeping.
c. Retrieve and track payments from customers and suppliers.

7. Include Computer Services (Maintenance, System Design, Software Development, Installation, and Recovery)

Apart from managing inventory, the system will support **additional computer services** that are crucial for maintaining IT assets. These services include:

a. Computer Maintenance: Managing scheduled servicing and repairs of IT components.
b. System Design: Assisting in designing and customizing IT infrastructure.
c. Software Development: Keeping track of custom software licenses and installations.
d. Software Installation: Managing and recording software installations across different systems.

You can have an accurate forecast and bring the material in such a way to minimize your total cost, but if it is not properly recorded in the computer system, there will probably problems after all, such as:
a. Bringing in unnecessary stock because previous stock receipts weren’t correctly recorded,

resulting in having more inventory than the system reports.
b. unexpected stock outs due to unrecorded material disbursements, substitutes, damaged parts, and other “sloppy” procedures. [2]

8. Ensure Data Security, Backups, and Disaster Recovery Mechanisms

To protect sensitive stock data, the system will implement strong security measures such as:

a. User authentication (JWT-based login)
b. Data encryption to secure transaction records.
c. Automated backups to prevent data loss. Disaster recovery mechanisms to restore stock records in case of hardware failure or cyberattacks.

By ensuring data integrity and security, the system will be reliable, scalable, and resistant to potential failures.

A Visual Representation of all the Objectives:

2. Requirements

The system requirements are categorized into **essential, recommended, and optional** to clearly define the scope of development.

2.1 Essential Requirements

The Essential requirements are fundamental to the system’s functionality.

a. A centralized database is required to store and manage stock records efficiently, ensuring real-time updates and accurate inventory tracking.
b. A real-time stock movement and location tracking system must be implemented to monitor the movement of computer components and their exact locations within the warehouse, reducing errors and improving stock retrieval efficiency.
c. The system will have an automated stock alert mechanism that notifies managers when inventory falls below a predefined threshold, ensuring smooth operations.
d. Role-based authentication and access control must be enforced, allowing different levels of system access for administrators, warehouse staff, and finance teams to enhance security and prevent unauthorized modifications.
e. Random Forest algorithm has great potential to improve e-commerce inventory management, opening opportunities for broader application in the digital business world. [3]
f. Additionally, an invoice management system must be included to generate, store, and retrieve invoices related to sales and purchases, ensuring seamless financial record-keeping.
g. The system will also support IT asset management services, including maintenance, system design, software installation, and recovery tracking, to manage IT-related stock efficiently.
h. Finally, data security, user authentication, encryption, and automatic backups should be implemented to protect stock records and ensure disaster recovery in case of data loss.

2.2 Recommended Requirements

The Recommended requirements include additional features that enhance efficiency and usability.

a. Barcode or QR code scanning should be introduced to streamline stock tracking and reduce human errors.
b. To support teamwork, multi-user collaboration should be implemented, allowing multiple employees to access and update inventory simultaneously.
c. A graphical warehouse mapping system can make stock retrieval more intuitive by visually displaying the locations of inventory items.
d. The system should also support cloud-based deployment, ensuring remote accessibility and multi-location warehouse management. Additionally, integration with third-party accounting and ERP systems would enable seamless synchronization of financial transactions with inventory records, improving operational efficiency.

2.3 Optional Requirements

The Optional requirements introduce advanced features that offer extra convenience but are not critical to the system's core functionality.

a. A mobile-friendly version should be developed to enable warehouse staff to manage stock from tablets or smartphones. Voice-command-based stock retrieval can be added to allow hands-free operation, improving efficiency for warehouse staff.
b. An AI-powered chatbot can assist users by answering queries related to stock levels, order statuses, and reports. To make the system more accessible to a global audience, multi-language support should be provided. There are many approaches related to computer vision-based stocktaking in unstructured warehouses with products stored in bulk, which employ several existing technologies namely Deep Learning, Optical Character Recognition, and 3D model retrieval. [4]

c. Lastly, a self-service portal for customers and suppliers can be introduced, allowing them to check inventory availability, place orders, and download invoices without requiring manual intervention from warehouse staff.

3. Technical Specification

3.1 Backend (Python & Flask/Django)

a. Python is selected due to its simplicity, rapid development capabilities, and extensive library support. The Django framework is used to build the backend because of its built-in authentication system, ORM (Object-Relational Mapping), and scalability. Django provides a structured approach to development, reducing development time while maintaining high security standards.
b. The Django REST Framework (DRF) will be used to develop RESTful APIs, enabling efficient communication between the frontend and backend.
c. To secure authentication, JWT (JSON Web Token) authentication will be implemented, allowing safe login/logout mechanisms.
d. The development environment for the backend will be set up using Visual Studio Code (VS Code) as the primary IDE and Postman for API testing.

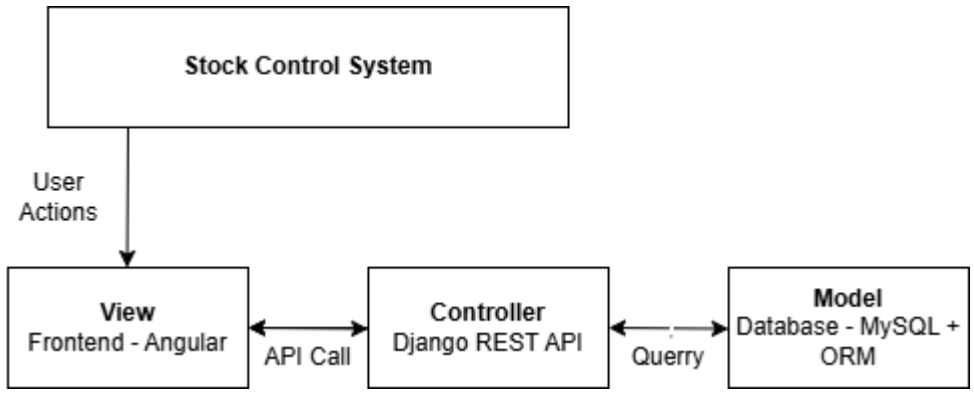
3.2 Frontend (TypeScript & Angular/React)

a. The frontend of the Stock Control System will be built using Angular with TypeScript, providing a structured, scalable, and maintainable user interface. Angular is chosen because of its component-based architecture, two-way data binding, and built-in security features.
b. The frontend will include libraries such as RxJS for handling real-time asynchronous data streams and Bootstrap CSS to enhance the visual design and responsiveness of the user interface.

6.2 Database – MySQL

a. For the database, MySQL is chosen over NoSQL alternatives like MongoDB due to its structured relational nature, ACID compliance, and strong transaction management capabilities.
b. To interact with MySQL, Django ORM (Object-Relational Mapping) will be used, simplifying database interactions without writing raw SQL queries.
c. The database schema will include key tables such as Stock Items, Users, Transactions, and Invoices, each with structured attributes like Warehouse ID, barcode, and SKU (Stock Keeping Unit) to enable precise stock categorization. To facilitate system testing and simulations, dummy datasets will be used to represent real-world inventory data.
d. This dataset will simulate warehouse stock movement, purchase transactions, and stock replenishment cycles.
e. Model View Controller Representation of a Stock Control System:

MODEL VIEW CONTROLLER diagram of Stock Control System:



Libraries and ORM Used for MySQL:

a. **MySQL Workbench** – A graphical tool to design and manage the MySQL database.
b. **Django ORM (Object-Relational Mapping)** – Instead of writing raw SQL queries, Django ORM allows developers to interact with the MySQL database using Python.
c. **MySQL Connector**– Enables Django to communicate with the MySQL database.

3.4. APIs and External Services

The Stock Control System will use several APIs and external services to enhance functionality:

- RESTful APIs – Created using Django REST Framework (DRF) to facilitate communication between the frontend and backend.
- JWT Authentication API – Secures login/logout functionality using JSON Web Tokens (JWT).
- Payment Gateway API (Optional) – To handle online transactions for stock purchases.

3.5. Data Visualization and Reporting

Since the system requires reporting and analytics, various Python-based libraries will be used for data visualization:

- Pandas – Used for data manipulation and generating reports.
- Matplotlib & Seaborn – Used for creating charts, graphs, and visual reports.
- Plotly – Used for interactive dashboards to visualize stock trends.
- ReportLab / FPDF – Used for exporting reports as PDFs.

3.6. Security and Authentication

Security is crucial for an inventory management system, especially for handling sensitive stock and financial data. The system will implement:

- JWT Authentication – Secures API endpoints to prevent unauthorized access.
- Password Hashing (bcrypt) – Ensures secure storage of user credentials.
- CORS Protection (8jango-cors-headers) – Prevents unauthorized frontend applications from making requests to the backend.
- Database Encryption – Protects sensitive data like user credentials and invoices.

3.7. Development Tools and Workflow

The project will follow an Agile development methodology, using tools for version control, testing, and debugging:

- Git & GitHub – Used for version control and collaboration.
- Jenkins/GitHub Actions (Optional) – For CI/CD pipeline automation.

- Unit Testing (PyTest & Jest) – Ensuring code reliability before deployment.
- Visual Studio Code (VS Code) – The IDE for coding in Angular and TypeScript.

4.Requirements Evaluation Plan

The evaluation plan will assess the system based on the following criteria:

- **Functionality Testing:** Ensure that all core functionalities (stock tracking, reporting, invoicing) operate correctly.
- **Performance Testing:** Evaluate system efficiency in handling stock updates and report generation.
- **Usability Testing:** Gather feedback from potential users to improve UI/UX.
- **Security Testing:** Verify authentication and data encryption mechanisms.
- **Scalability Testing:** Assess the system's capability to handle increased stock levels and user interactions.

6. Background Research and Reading list

5.1 Background Research Paper List

1. Stock Inventory Management System
2. Managing stock in warehouse: a case study of a retail industry in Jakarta
3. Optimizing e-commerce inventory to prevent stock outs using the random forest algorithm approach.
4. Towards automating stocktaking in warehouses: Challenges, trends, and reliable approaches.
5. Data Visualization in Annual Reports.

5.2 Reading List

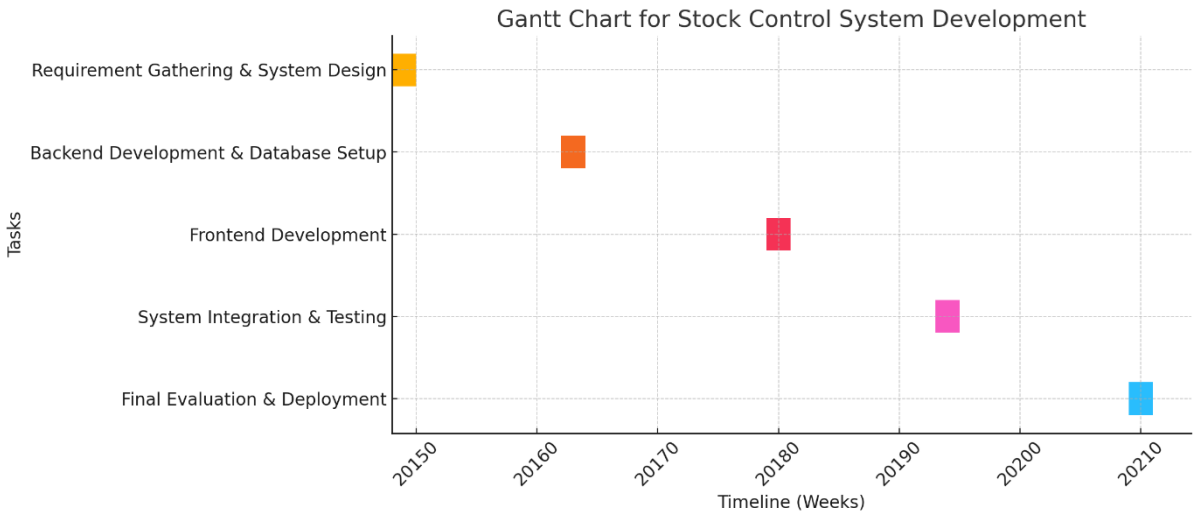
1. "Designing Data-Intensive Applications" – Martin Kleppmann.
2. "Flask Web Development" – Miguel Grinberg.
3. "Python for Data Analysis" – Wes McKinney.

6. Time-plan and Risk Plan

6.1 Project Plan (Gantt Chart)

A detailed Gantt chart will be created to track progress in **project phases**:

- **Phase 1 (Week 1-2):** Requirement gathering and system architecture design.
- **Phase 2 (Week 3-4):** Backend and database implementation.
- **Phase 3 (Week 5-6):** Frontend UI development.
- **Phase 4 (Week 7-8):** System integration and testing.
- **Phase 5 (Week 9-10):** Final evaluation and deployment.



6.2 Risk Assessment & Mitigation

Risk	Impact	Mitigation Strategy
1.Security vulnerabilities	High	Implement robust authentication & encryption.
2. Database failures	High	Implement automated backups & failover solutions.
3. Scope creep	Medium	Stick to essential requirements; follow Agile principles.
4. Performance issues	Medium	Optimize queries and implement caching mechanisms.

7. Overall Impression

This project will deliver a robust, scalable, and efficient Stock Control System that improves warehouse inventory management. By leveraging Python, TypeScript, Django, Angular orReact, and modern APIs, the system will provide real-time tracking, invoicing, and reporting while maintaining security and performance. The project plan ensures timely completion, while the risk plan mitigates potential challenges.

References

1.Sameen, M. A. (2023). *Stock Inventory Management System* (Doctoral dissertation, Soanargaon Universiy (SU)).
2. Paul, Y., & Lestari, Y. D. (2015). Managing stock in warehouse: a case study of a retail industry in Jakarta. *Journal of Business and Management*, *4*(7).

3. Ridwan, A., Muzakir, U., & Nurhidayati, S. (2024). Optimizing e-commerce inventory to prevent stock outs using the random forest algorithm approach. *International Journal Software Engineering and Computer Science (IJSECS)*, *4*(1), 107-120.
4. Daios, A., Xanthopoulos, A., Folinas, D., & Kostavelis, I. (2024). Towards automating stocktaking in warehouses: Challenges, trends, and reliable approaches. *Procedia computer science*, *232*, 1437-1445.
5. Uddin, M. M., Ullah, R., & Moniruzzaman, M. (2024). Data Visualization in Annual Reports–Impacting Investment Decisions. *International Journal for Multidisciplinary Research*, *6*(5).