**A picture containing text

Description automatically generated**

**School of Computing and Mathematical Sciences**

**CO7201 Individual Project**

**Preliminary Report**

**Stock Control System**

**Name - Apurva Vivek Kulkarni**

**Email Id –** [**avk7@**](mailto:avk7@)**student.le.ac.uk  
Student Id - 239064472**

**Project Supervisor: Anthony Conway**

**Principal Marker: Dr Paula Severi**

**Word Count: [XXXX]**

**[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** 3](#_Toc65510906)

[**2.** **Requirements** 3](#_Toc65510907)

[**3.** **Technical Specification** 3](#_Toc65510908)

[**4.** **Requirements Evaluation Plan** 3](#_Toc65510909)

[**5.** **Background Research and Reading list** 3](#_Toc65510910)

[**6.** **Time-plan and Risk Plan** 3](#_Toc65510911)

[**7.** **References** 3](#_Toc65510912)

# **Aims and Objectives**

A brief discussion on project relevance/motivations (why is it worth doing it?) and its main challenges.  
  
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. 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. Ultimately, this project aims to improve warehouse efficiency, enhance stock visibility, and provide a reliable stock control mechanism that minimizes human errors and enhances operational workflow.

**1. 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.

**2. 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.

**3. 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. This helps in avoiding stock shortages and ensures that critical components are reordered before they run out. 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:

* **Admins** will have full access to stock control, invoicing, and reporting.
* **Warehouse** **Staff** will be able to update stock levels and track inventory movement.
* **Finance** **Teams** will manage invoices and payments. This approach enhances security, prevents unauthorized changes, and reduces errors in inventory handling.

**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:

* Identifying fast-moving and slow-moving items.
* Predicting future stock demand using historical trends.
* 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.

**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:

* Generate invoices for stock sold or purchased.
* Store past invoices for record-keeping.
* Retrieve and track payments from customers and suppliers.
* Integrate tax calculations and payment methods. Invoices will be automatically linked to stock movement, ensuring that stock records are updated whenever a sale or purchase occurs.

**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:

* Computer Maintenance: Managing scheduled servicing and repairs of IT components.
* System Design: Assisting in designing and customizing IT infrastructure.
* Software Development: Keeping track of custom software licenses and installations.
* Software Installation: Managing and recording software installations across different systems.
* Computer Recovery: Implementing disaster recovery services to prevent data loss and ensure business continuity.

A group of yellow rectangular signs

AI-generated content may be incorrect.

# **Requirements**

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

List of requirements, including high-level requirements or aims, and detailed requirements or objectives. You should separate these into essential, recommended and optional.  
  
**2.1 Essential Requirements**

The Essential requirements are fundamental to the system's functionality. A centralized database is required to store and manage stock records efficiently, ensuring real-time updates and accurate inventory tracking. 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. The system should also have an automated stock alert mechanism that notifies managers when inventory falls below a predefined threshold, ensuring smooth operations. 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. Comprehensive reporting and analytics tools should be integrated to generate stock movement trends, demand forecasting, and inventory status reports, helping decision-makers plan effectively. Additionally, an invoice management system must be included to generate, store, and retrieve invoices related to sales and purchases, ensuring seamless financial record-keeping. The system should also support IT asset management services, including maintenance, system design, software installation, and recovery tracking, to manage IT-related stock efficiently. 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. Barcode or QR code scanning should be introduced to streamline stock tracking and reduce human errors. To support teamwork, multi-user collaboration should be implemented, allowing multiple employees to access and update inventory simultaneously. The reporting module can be improved by providing customized reporting dashboards with filter options, multiple export formats (PDF, Excel, CSV), and scheduled automated reports. To further enhance inventory planning, AI-powered demand forecasting should be integrated to predict stock shortages and prevent overstocking. A graphical warehouse mapping system will make stock retrieval more intuitive by visually displaying the locations of inventory items. 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.

* 1. **Optional Requirements**

The Optional requirements introduce advanced features that offer extra convenience but are not critical to the system's core functionality. 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. 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. Additionally, IoT sensors can be integrated to automatically detect stock movement and update inventory records in real-time. 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.

# **Technical Specification**

**3.1 Backend (Python & Flask/Django)**

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. The Django REST Framework (DRF) will be used to develop RESTful APIs, enabling efficient communication between the frontend and backend. To secure authentication, JWT (JSON Web Token) authentication will be implemented, allowing safe login/logout mechanisms. Additional libraries such as bcrypt will be used for password hashing, while django-cors-headers will enable Cross-Origin Resource Sharing (CORS) to allow communication between the backend and the Angular frontend. For background tasks such as sending stock alerts and notifications, Celery and Redis will be integrated. 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)**

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. It ensures high-performance rendering through lazy loading and Ahead-of-Time (AOT) compilation, making the system more efficient. The frontend will include libraries such as RxJS for handling real-time asynchronous data streams and Bootstrap or Tailwind CSS to enhance the visual design and responsiveness of the user interface. To improve user experience, Angular Material will be used for pre-built UI components, while Ngx-Charts or Chart.js will be used for data visualization and reports. The frontend development will be carried out in Visual Studio Code (VS Code) with Chrome DevTools for debugging.  
**3.3 Database – MySQL**

1. For the database, MySQL is chosen over NoSQL alternatives like MongoDB due to its structured relational nature, ACID compliance, and strong transaction management capabilities. Inventory management systems require well-structured data models with defined relationships, constraints, and referential integrity, making MySQL the ideal choice. MySQL supports complex queries, indexing, and JOIN operations, which are essential for efficiently handling inventory data. Additionally, MySQL’s reliable transaction management system ensures that updates to stock levels remain consistent even in multi-user environments. To interact with MySQL, Django ORM (Object-Relational Mapping) will be used, simplifying database interactions without writing raw SQL queries. The MySQL Workbench tool will be used for designing, managing, and querying the database schema, ensuring that database operations are executed smoothly.  
     
   Libraries and ORM Used for MySQL:
2. **MySQL Workbench** – A graphical tool to design and manage the MySQL database.
3. **Django ORM (Object-Relational Mapping)** – Instead of writing raw SQL queries, Django ORM allows developers to interact with the MySQL database using Python.
4. **MySQL Connector / PyMySQL** – Enables Django to communicate with the MySQL database.

MySQL Workbench – A GUI-based tool for designing, managing, and querying 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).
* SMTP or Twilio API – Used for email or SMS notifications when stock levels fall below a threshold.
* 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/Dash – 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 (django-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.
* Docker (Optional) – For containerization and easy deployment.
* Jenkins/GitHub Actions (Optional) – For CI/CD pipeline automation.
* Unit Testing (PyTest & Jest) – Ensuring code reliability before deployment.

**Tools Used for Frontend Development**

Visual Studio Code (VS Code) – The IDE for coding in Angular and TypeScript.

Chrome DevTools – For debugging and optimizing frontend performance.

# **4.Requirements Evaluation Plan**

What criteria to use to evaluate your system or experimental results? Who will be involved in the evaluation? What type of testing will you do to verify the quality of your software? In short, how will you verify that your project achieves what it sets out to do?  
  
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.

# **5. Background Research and Reading list**

Background material (including a reading list and literature review where appropriate);  
  
**5.1 Background Research**

* **Warehouse Management Systems (WMS)** and their role in inventory tracking.
* **Best practices in inventory control and supply chain management.**
* **Modern web technologies for building scalable stock management systems.**

**5.2 Reading List**

1. “Designing Data-Intensive Applications” – Martin Kleppmann.
2. “Flask Web Development” – Miguel Grinberg.
3. “Python for Data Analysis” – Wes McKinney.
4. “JavaScript and TypeScript: Web Development Guide” – Ethan Brown.
5. Research papers on **inventory control techniques** and **warehouse management systems**.

# **Time-plan and Risk Plan**

A detailed timetable and plan for achieving the objectives of the project (this could be tabulated or in the form of a Gantt chart), including the milestones of the project and a risk plan; Explain target dates and amount of time required for the completion of aspects of the project  
  
**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.

A graph with red and orange squares

AI-generated content may be incorrect.

**6.2 Risk Assessment & Mitigation**

| **Risk** | **Impact** | **Mitigation Strategy** |
| --- | --- | --- |
| Security vulnerabilities | High | Implement robust authentication & encryption. |
| Database failures | High | Implement automated backups & failover solutions. |
| Scope creep | Medium | Stick to essential requirements; follow Agile principles. |
| 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, Flask/Django, Angular/React**, 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.

This report clearly defines the project objectives, technical stack, requirements, and a well-structured execution plan, ensuring successful project delivery.