Department of Computer Science and Engineering

B.Tech (CSE)-5th Semester-Aug-Dec2024

UE22CS341A – Software Engineering

PROJECT PLAN DOCUMENT

KIRANA STORE MANAGEMENT SYSTEM

Team #: _

Deliverable 2: Planning, designing, and implementation details

PES1UG22CS478	**Rishith P**
PES1UG222CS443	**Prathik S Hanji**

Project Plan: Inventory Management System (IMS) for Kirana Stores

Introduction

The IMS aims to automate inventory management for Kirana stores, focusing on product management, sales tracking, stock monitoring, supplier management, and reporting. The system will be built using **Agile methodology** with a focus on scalability, security, and usability, utilizing **MySQL** as the core relational database for structured data management.

Deliverables of the Project

1. **IMS Platform**: A web-based system accessible on desktop and mobile devices, built with **React** for the frontend and **Node.js** with **Express.js** for the backend.
2. **User Authentication and Management**: Secure registration, login, and role-based access using **Auth0** and **OAuth2**.
3. **Product and Stock Management**: Manage products, stock levels, and expiry dates, utilizing **MySQL** as the relational database.
4. **Sales Management**: Real-time sales tracking where users manually enter the payment amount. The system records these sales transactions and stores them in the MySQL database for future reference and reporting.
5. **Supplier Management**: Supplier database and automated reordering based on stock levels, leveraging **SMTP-based email notifications**.
6. **Reporting and Analytics**: Generate sales and stock reports using **MySQL Aggregation** and visualized with **Chart.js** on the frontend.

Process Model

The system follows a **microservices architecture**, with services interacting via **RESTful APIs**. Each service handles a specific domain (e.g., sales, product management, supplier management) with a **MySQL** backend for data storage.

1. **User Interaction Process**: Built with **React** for a responsive UI and **Node.js** on the backend. Users perform tasks like adding products, recording sales, entering and managing supplier information. Recording the previous inventory audit, with data stored in **MySQL**. User maintains a customized environment based on his preferences like Theme, notification preferences , Currency ,Re Order Level Threshold for a seamless experience.
2. **Inventory Management Process**: The backend handles product creation,Product quantity updation , stock monitoring , Purchase transactions ,inventory audits and expiry alerts, all managed via **MySQL** queries for CRUD operations.

3. **Supplier Management Process**: Automates supplier management, storing supplier and purchase order data in **MySQL**, and uses **SMTP** email notifications for low-stock reordering.
4. **Reporting and Analytics Process**: Uses **MySQL** to aggregate sales and inventory data, with **Chart.js** visualizing the reports and provides insights or recommendations based on the report.

Organization of the Project

The project follows the **Scrum Agile framework**, with bi-weekly sprints. Team roles include:

1. **Project Manager**: Uses **Gantt and Whiteboard and Sticky Notes** for timeline management and stakeholder communication.
2. **Product Owner**: Prioritizes the backlog and ensures the system meets business needs.
3. **Developers**: Implement system features using **Node.js** (backend), **React** (frontend), and **MySQL** for the database.
4. **QA Engineers**: Perform tests using **Selenium** (UI tests), and **Postman** (API tests).
5. **UI/UX Designers**: Design interfaces using **Figma** and develop them with **React**.
6. **DevOps Engineers**: Handle deployment on **AWS** or **Vercel**.

Standards and Guidelines

1. **Coding Standards**: Use **ESLint** for JavaScript linting and **Prettier** for formatting, following **SOLID principles**.
2. **Agile Practices**: Use **Scrum** for sprint planning and daily standups, with **Git** for version control.
3. **API Design Guidelines:** Follow RESTful API design standards, ensuring scalability, security, and clear documentation.
4. **Testing Procedures**: Testing using **Selenium** and **Postman**.

Management Activities

1. **Sprint Planning and Management**: Sprints managed using **Gantt and Whiteboard and Sticky Notes** for task assignment and tracking.
2. **Resource Management**: Developers, testers, and designers are allocated based on sprint priorities.

3. **Risk Management**: Identified risks such as scalability or user adoption will be tracked and mitigated.
4. **Status Reporting**: Weekly updates on project progress using **Gantt and Whiteboard and Sticky Notes** and regular sprint reviews with stakeholders.
5. **Version Control**: Managed via **GitHub** or **GitLab** with feature branching.

Risks

1. **Scalability Risks**: Increased traffic may require optimized database queries and scaling.
    - **Mitigation**: Use **MySQL performance optimization** (e.g., indexing, query tuning) and **AWS RDS** for scalable SQL infrastructure.
2. **User Adoption**: Poor UI/UX may lead to low engagement.
    - **Mitigation**: Regular user feedback incorporated into UI improvements.
3. **Resource Availability**: Potential delays due to limited availability of developers.
    - **Mitigation**: Cross-train team members and maintain flexibility in the sprint schedule.

Staffing

1. **Development Team**:
    - **Frontend Developers**: Build interfaces with **React**.
    - **Backend Developers**: Develop services using **Node.js** and connect to **MySQL**.
    - **Database Administrators**: Manage **MySQL** schema design, performance tuning, and backups.
2. **Testing Team**:
    - **QA Engineers**: Conduct manual tests with **Postman**, **ThunderBird**.
3. **DevOps Team**:
    - **Engineers**: Deploy applications using Vercel or AWS for deployment**MySQL** database management.

Methods and Techniques

1. **Agile Methodology**: Scrum-based, with iterative sprints and continuous delivery.

2. **Microservices Architecture**: Backend services will communicate via **REST APIs** and store data in **MySQL**.
3. **RESTful API Design**: APIs follow REST standards, documented with **Swagger**.
4. **Automated Testing**: Use **Jenkins** or **GitHub Actions** for automating test builds and deployments.
5. **SQL Database (MySQL)**: **MySQL** will serve as the relational database, designed for structured data storage and optimized query performance.

Quality Assurance

1. **Code Quality**: Enforced with **ESLint**, **Prettier**, and regular peer reviews.
2. **Performance Testing**: Use **JMeter** for load testing to ensure the system handles peak load efficiently.
3. **Security Testing**: Implement regular **SQL injection tests** and use tools like **OWASP ZAP**.
4. **User Acceptance Testing (UAT)**: Involve end-users in testing to validate system usability.

Work Packages

1. **Product and Stock Management**: Implement CRUD operations for product management, integrated with **MySQL**.
2. **Sales Management**: Develop sales tracking with transactions stored in **MySQL**.
3. **Supplier Management**: Automate supplier reordering and manage supplier records in **MySQL**.
4. **Reporting and Analytics**: Build reporting functionality using **MySQL** queries, analysis the inventory, sales, purchases, products using **Simple-Statistics, jstat, statistics** and visualized with **Chart.js**.

Resources

1. **Human Resources:**

   1. Development Team:

   ○ **Front-end Developers:** Focus on building user interfaces using technologies like React or Angular.

○ **Back-end Developers:** Responsible for creating microservices, APIs, and database management, using Node.js, Python, or Java.

○ **Security Engineers:** Implement data protection mechanisms

2. Quality Assurance Team:

○ **Testers:** Focus on manual and automated testing to ensure the platform functions as expected under various conditions.

○ **Performance Engineers:** Handle load and stress testing to confirm the system's ability

3. DevOps Team:

○ **Engineers:** Manage the cloud infrastructure (AWS, GCP), handle CI/CD pipelines, and ensure smooth deployment and scaling**.**

4. UI/UX Designers:

○ **Designers:** Create intuitive and responsive interfaces for users, ensuring ease of use across devices.

5. Project Management:

○ **Project Managers:** Oversee project timelines, sprint planning, and communication with stakeholders.

○ **ScrumMasters:** Ensure Agile processes are followed.

2. **Technology Resources:**
    - **1. Cloud Infrastructure:** Services like AWS or Google Cloud for hosting, storage, and server management.
    - **2. Development Tools:** Version control (Git), code management platforms (GitHub, GitLab), and integrated development environments (IDEs).

Schedule

The project will proceed through the following phases:

1. **Phase 1: Requirements Gathering**: Identify functional and non-functional requirements.
2. **Phase 2: Architecture and Design**: Define the system architecture, focusing on **MySQL** as the core database.
3. **Phase 3: Core Feature Development**: Implement product management, sales, and supplier management modules.
4. **Phase 4: Testing and Quality Assurance**: Conduct testing using **Postman**.

5. **Phase 5: Performance Optimization and Deployment**: Optimize **MySQL** queries and deploy on **Vercel or AWS**.
6. **Phase 6: Final Review and UAT**: Conduct user acceptance testing before final deployment.

Delivery Means

The delivery of the IMS project will be managed through the following steps, ensuring continuous iteration, user feedback, and incremental improvements:

1. Regular Sprint Deliverables

- **Sprint Duration**: Each sprint will last 2-4 weeks, depending on the complexity of the tasks.
- **Sprint Planning**: At the start of each sprint, the team will plan the tasks to be delivered during that sprint. Tasks will be selected from the product backlog based on priority.
- **Sprint Deliverables**: At the end of each sprint, working software or features (such as product management or sales tracking) will be delivered to the product owner for review and feedback. Each deliverable will be fully functional and tested.
- **Incremental Improvements**: Based on user feedback and any discovered issues, small iterative improvements will be implemented in subsequent sprints. This ensures continuous delivery of features, bug fixes, and enhancements.

Example Sprint Deliverables:

- **Sprint 1**: Basic UI for product management and initial database setup.
- **Sprint 2**: Full product CRUD operations integrated with MySQL.
- **Sprint 3**: Real-time sales tracking and inventory updates.

2. User Feedback and Beta Testing

- **Beta Release**: A beta version of the IMS will be released to a limited group of users (e.g., selected Kirana store owners, managers) before the official release. This will allow early feedback and testing of the core functionalities under real-world conditions.
- **User Testing**: Beta users will test the system, focusing on key functionalities like product management, sales, and stock monitoring. Feedback will be collected to identify usability issues, bugs, and potential areas of improvement.
- **Feedback Loop**: All feedback will be gathered in regular sprint reviews and will guide the next iteration's focus. Critical bugs will be addressed immediately, while feature requests or improvements will be added to the backlog.

Example Beta Testing Milestones:

- **Beta 1**: Product management, supplier management, and sales tracking.
- **Beta 2**: Integration of low-stock alerts, reporting, and supplier reordering automation.

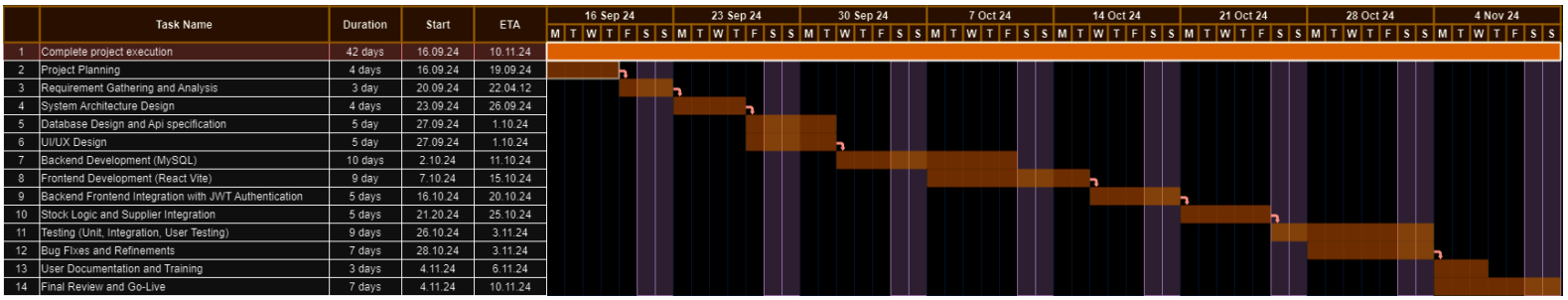
3. Final Deployment

- **Pre-Deployment Testing**: Before final deployment, a thorough end-to-end test will be conducted on all features to ensure that the system meets functional and non-functional requirements (e.g., performance, security, scalability).
- **Cloud Deployment**: The final version of the IMS will be deployed on a scalable cloud infrastructure such as **AWS RDS** for **MySQL**, ensuring the system can handle real-time transactions and increased load as store traffic grows.
- **Documentation**: The delivery will include:
    - **System Documentation**: Including the overall architecture, database schema, and design decisions.
    - **User Documentation**: Step-by-step guides for store owners, managers, and staff on using the system.
    - **Administrative Manuals**: Instructions for system administrators on managing users, monitoring system health, and maintaining the database.

4. Post-Launch Support

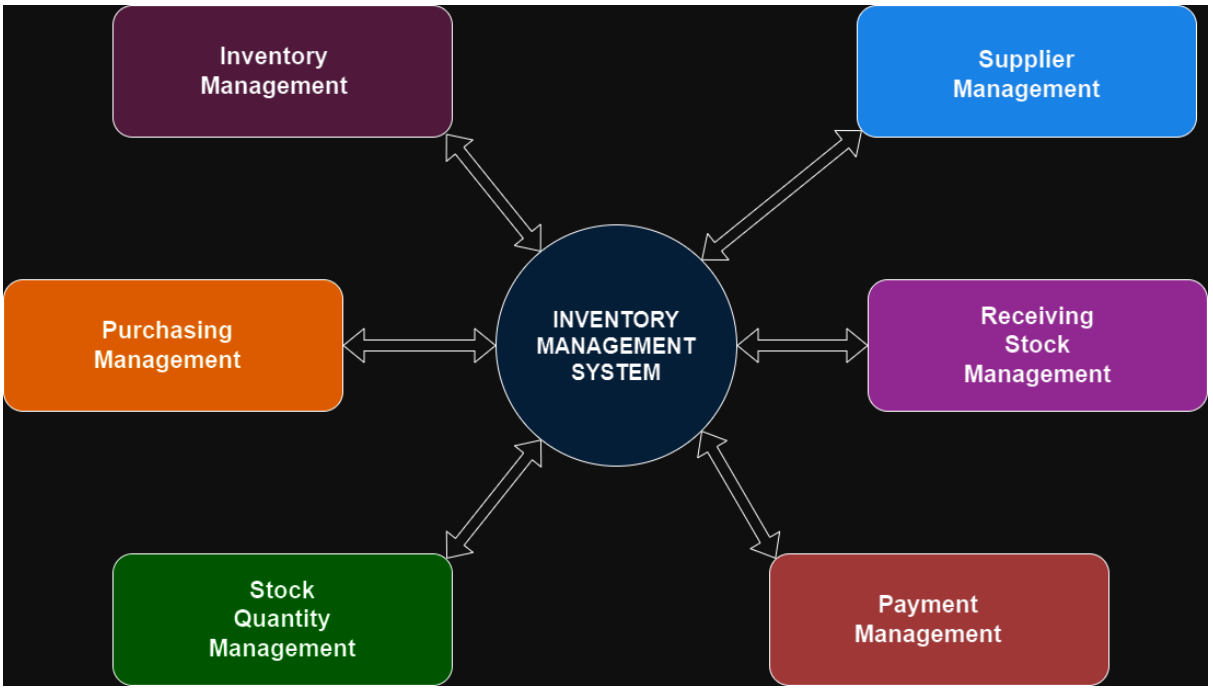
- **Monitoring**: After the system is deployed, **real-time monitoring tools** (such as **Prometheus** or **New Relic**) will be used to track system health, uptime, and performance. Metrics will be gathered on user interactions, API calls, and database performance to ensure smooth operations.
- **Bug Fixes**: Any critical bugs identified after deployment will be prioritized and fixed immediately. A robust error-handling mechanism will be in place to log issues and notify the support team.
- **Performance Optimization**: Periodic performance reviews will be conducted to ensure the system remains scalable and responsive. This may involve query optimization, database indexing, and load balancing adjustments.
- **Minor Releases and Patches**: The system will be regularly updated with patches for bugs, minor feature requests, and security updates. These will be released via **CI/CD pipelines** to minimize downtime.
- **User Support**: A dedicated support channel will be made available for store owners and managers to report issues or ask for assistance.

Gantt Chart

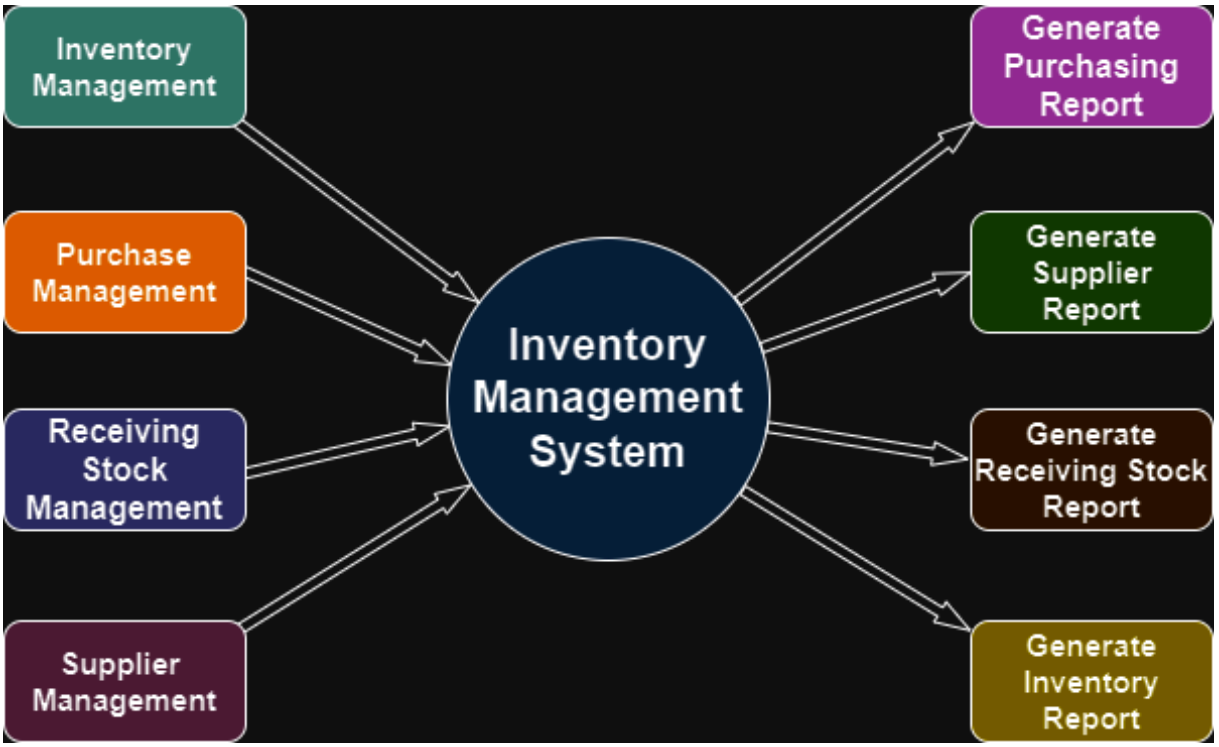


	Task Name	Duration	Start	ETA
1	Complete project execution	42 days	16.09.24	10.11.24
2	Project Planning	4 days	16.09.24	19.09.24
3	Requirement Gathering and Analysis	3 day	20.09.24	22.04.12
4	System Architecture Design	4 days	23.09.24	26.09.24
5	Database Design and Api specification	5 day	27.09.24	1.10.24
6	UI/UX Design	5 day	27.09.24	1.10.24
7	Backend Development (MySQL)	10 days	2.10.24	11.10.24
8	Frontend Development (React Vite)	9 day	7.10.24	15.10.24
9	Backend Frontend Integration with JWT Authentication	5 days	16.10.24	20.10.24
10	Stock Logic and Supplier Integration	5 days	21.20.24	25.10.24
11	Testing (Unit, Integration, User Testing)	9 days	26.10.24	3.11.24
12	Bug Fixes and Refinements	7 days	28.10.24	3.11.24
13	User Documentation and Training	3 days	4.11.24	6.11.24
14	Final Review and Go-Live	7 days	4.11.24	10.11.24

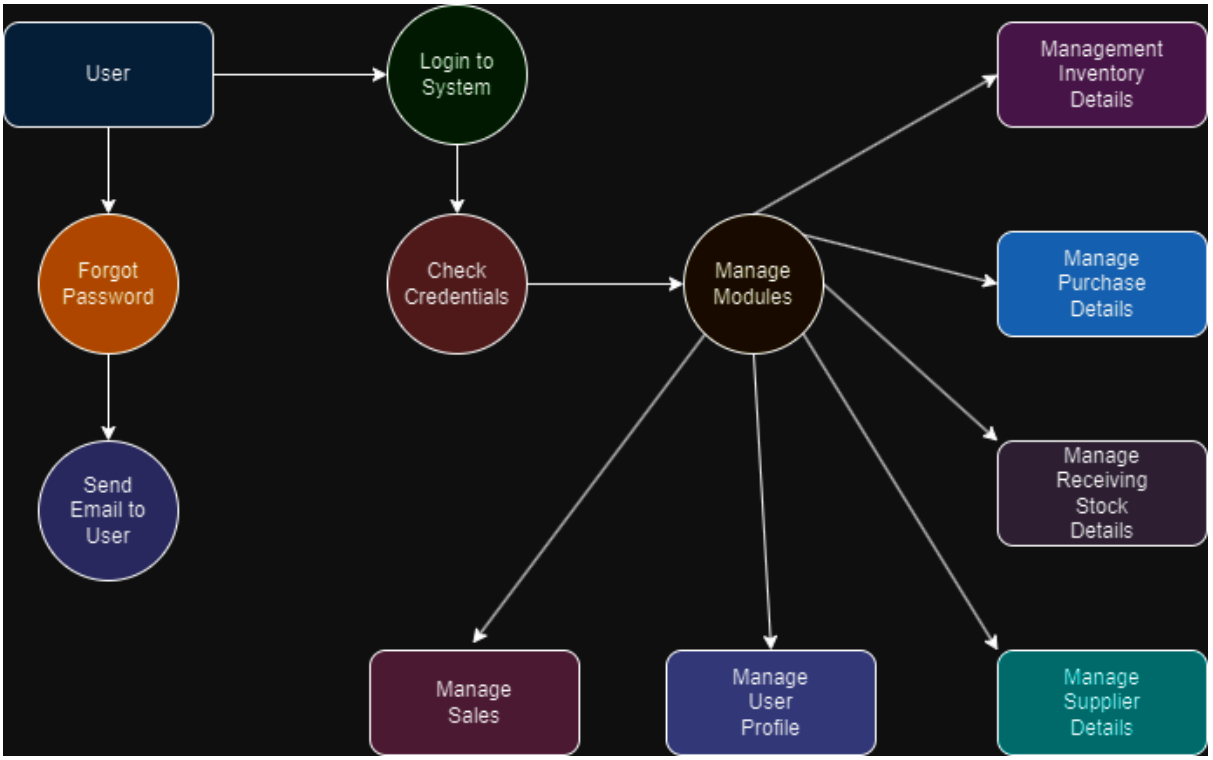
DATA FLOW DIAGRAM



Level Zero



Level One



Level Two

Team members Details:

1. **Rishith P - PES1UG22CS478**
2. **Prathik S Hanji - PES1UG22CS443**