Inventory Management System

Planning & Management

1. Project Proposal

Overview of the project:

The **Inventory Management System** is a web-based software application designed to efficiently organize and track inventory within a warehouse or business environment. The system will be developed using the **ASP.NET Core MVC** framework, ensuring a structured separation of concerns: **Model (business logic), View (user interface), and Controller (flow control).**

The application will enable businesses to:

- Manage stock levels and track product movements.
- Generate insightful reports to enhance decision-making.
- Maintain a secure, scalable, and user-friendly inventory system.

Objectives:

- Develop an **intuitive user interface** for inventory management (adding, editing, retrieving, and deleting products).
- Implement **real-time stock tracking** to prevent overstocking or stockouts.
- Generate **analytical reports** on inventory trends, such as **high-demand and low-stock products**.
- Enhance operational efficiency through automation and optimized workflows.
- Ensure **scalability** to support business growth and additional features in the future.

Scope:

- **Product Management (name, quantity, price, category, stock tracking).**
- **Supplier & Customer Management for tracking transactions.**
- **Order Processing with inventory adjustments.**
- **User Authentication & Role Management (Admin, Manager, Staff).**
- **Inventory Reports & Low-Stock Alerts for better decision-making.**
- **Comprehensive Dashboard with real-time insights into key performance indicators.**

2- Project Plan

Timeline (Gantt Chart)

Phase	Duration	Weeks
Requirements Analysis	3 days	Week 1
Database Design	4 days	Week 1
Core Development	3 weeks	Weeks 2 - 4
- Model Development	1 week	Week 2
- Controller Development	1 week	Week 3
- View Development	1 week	Week 4
Testing & Refinement	1 week	Week 5
Final Delivery & Review	1 week	Week 6

Milestones:

Milestone	Expected Completion
Completion of Requirements Analysis	End of Week 1
Database Design & Setup	End of Week 1
Model Development Completion	End of Week 2
Controller Development Completion	End of Week 3
View (Interface) Development Completion	End of Week 4
System Testing & Refinement	End of Week 5
Final Delivery & Review	End of Week 6

Deliverables:

Deliverable	Description	Timeline
Requirements Document	Detailed system requirements & specifications	Week 1
Functional Database	Designed & implemented database (SQL Server)	Week 1
Model Code	Backend logic & database interaction (ASP.NET Core MVC)	Week 2
Controller Code	Handles business logic & request processing	Week 3
View (UI) Components	User interface & front-end pages (Bootstrap/Tailwind)	Week 4
System Test Report	Testing results, bug fixes, and performance improvements	Week 5
User Manual & Documentation	Guide on system usage and features	Week 6

Resource Allocation:

Role	Responsibilities
Project Manager	Tracks progress, coordinates team, ensures milestone completion
Business Analyst	Defines requirements & validates project scope
Database Developer	Designs schema, manages SQL Server & Entity Framework
Backend Developer	Develops Models & Controllers (ASP.NET Core MVC)
Frontend Developer	Develops Views, UI components, and dashboard (Bootstrap/Tailwind)
QA Tester	Performs system testing, reports bugs, and ensures functionality
Deployment Engineer	Handles deployment, hosting, and server configuration

Tools & Technologies:

- **IDE:** Visual Studio
- **Database:** SQL Server
- **Version Control:** GitHub/Git
- **Testing Tools:** Postman, Selenium (for UI Testing)

Task Assignment & Roles

Project Manager

The project manager oversees the entire development process, ensuring that milestones are met on time. They coordinate between team members, track progress, and resolve any issues that may arise throughout the project. Their main focus is ensuring the smooth execution of tasks and maintaining the project timeline.

Business Analyst

Responsible for gathering and analyzing business requirements, the business analyst defines the project's functional and non-functional requirements. They work closely with stakeholders to document expectations and ensure the system aligns with business needs. This role is crucial during the first week for defining the project scope.

Database Developer

The database developer designs and implements the database schema using SQL Server. They ensure data integrity, optimize queries, and set up relationships between tables. Their work is primarily focused on the first week, ensuring that a solid foundation is established for the backend development.

Backend Developer

The backend developer is responsible for implementing the system’s logic using ASP.NET Core MVC. They develop models and controllers to handle business processes, ensuring efficient interaction between the database and the user interface. Their work starts in **Week 2**, focusing first on model development, followed by controller implementation in **Week 3**.

Frontend Developer

The frontend developer is responsible for creating a user-friendly and responsive interface using Bootstrap and Tailwind CSS. They ensure that the UI is intuitive and properly connected to the backend. Their work begins in **Week 4**, focusing on implementing the view layer and styling the application.

QA Tester

The QA tester ensures that the system functions correctly by conducting unit, integration, and system testing. They identify and report bugs, verify system performance, and ensure that all features meet the requirements. Testing starts in **Week 5**, allowing time for any necessary refinements before final deployment.

Deployment Engineer

The deployment engineer is responsible for hosting and deploying the system. They configure servers, manage security settings, and ensure smooth deployment. Their work takes place in **Week 6**, ensuring the application is ready for production use.

Risk Assessment & Mitigation Plan

1. Requirement Misalignment

- **Risk: Miscommunication or unclear requirements may lead to a system that does not fully meet business needs.**
- **Mitigation: Conduct detailed requirement analysis sessions in Week 1, involve stakeholders early, and document everything properly. Regular feedback loops will be maintained.**

2. Database Performance Issues

- **Risk: Poorly optimized queries can slow down the system, affecting performance.**
- **Mitigation: Implement indexing, optimize queries, and use Entity Framework best practices. Load testing will be performed during Week 5 to assess database performance.**

3. Development Delays

- **Risk: Unforeseen issues may cause delays in coding, testing, or deployment.**
- **Mitigation: Follow the structured timeline, allocate buffer time, and hold weekly progress reviews to identify and resolve bottlenecks early.**

4. Security Vulnerabilities

- **Risk: Unauthorized access or data breaches may compromise system security.**
- **Mitigation: Implement secure authentication (ASP.NET Identity), encrypt sensitive data, and conduct security testing in Week 5 before deployment.**

5. System Scalability Limitations

- **Risk: The system may not handle increased data or users in the future.**
- **Mitigation: Design a scalable database architecture, use caching mechanisms, and ensure modular code that supports future enhancements.**

6. User Adoption Challenges

- **Risk: Employees may find it difficult to use the new system, leading to low adoption rates.**
- **Mitigation: Provide training sessions, user manuals, and support to help users adapt. Gather user feedback to refine the UI/UX.**

7. Deployment & Hosting Issues

- **Risk: Technical problems during deployment could cause downtime.**
- **Mitigation: Test deployment in a staging environment before going live. The deployment engineer will handle server setup and security measures in Week 6**.

KPIs (Key Performance Indicators)

Performance Metrics

Response Time: System should process requests within **< 2 seconds**.
System Uptime: Maintain **99.9% uptime** to ensure reliability.
Database Query Execution Time: All queries should be executed in **< 500ms** under normal load conditions.

User Engagement Metrics

User Adoption Rate: At least **80% of targeted users** should actively use the system within the first **3 months**.
User Satisfaction Score: Gather feedback from users; target a **90% satisfaction rate** through surveys.

Operational Efficiency Metrics

Inventory Accuracy Rate: Ensure inventory data has a **99% accuracy level** compared to actual stock.
Low-Stock Alert Efficiency: System should notify users **immediately** when stock reaches reorder levels.
Error Rate in Transactions: Should be **< 1%** after deployment.