Hotel Management System

problem statement :- Managing a hotel manually or using disparate systems can lead to inefficiencies, errors, and poor customer experience. Tasks such as booking reservations, managing check-ins / check-outs, handling room availability, billing, and maintaining customer records are time-consuming and prone to mistakes. There is a need for an automated Hotel Management System (HMS) to streamline operations, improve accuracy, and enhance customer satisfaction by providing a centralized platform for managing all hotel-related activities.

Introduction

Purpose :-

The purpose of this document is to specify the requirements for hotel management system the system will automate the management of hotel operations such as room booking, guest check-in / check out billing and reporting.

Scope :-

The HMS will provide functionalities for managing hotel rooms, reservations, guest details, billing and housekeeping activities. It will generate reports for management to analyze business performance

Overview :-
The SRS document describes the functional and non-functional requirements for HMS. It includes system features, external interfaces and constraints.

general Description :-
A web-based system to automate hotel operations including room booking, guest check-in / check-out billing, housekeeping, and reporting. Used by receptionists, managers, and housekeeping staff to improve efficiency and accuracy.

Functional Requirements :-
1) User login with role-based access
2) Manage rooms (add / edit / status)
3) Create, update, and cancel reservations
4) Guest check-in and check-out processing
5) Billing with multiple payment options and invoice generation
6) Housekeeping updates for room status

Performance Requirements :-
1) Support 100+ concurrent users
2) Response time : under 3 seconds per action
3) 99.5% system uptime
4) Daily data backups

Design Constraints :-
1) Must comply with data privacy laws
2) Secure data transmission via HTTPS

3) Modular design for easy updates
4) Operate on standard web servers and browsers

Non functional Attributes :-
Security : encrypted passwords, role-based access
usability : Intuitive UI, minimal training needed
Reliability : high availability, backup and recovery
maintainability :- easy to modify and enhance

Preliminary schedule :-
Requirements gathering : 2 weeks
Design phase : 3 weeks
Development : 8 weeks
Testing : 3 weeks
Deployment and training : 2 weeks
Total = 18 weeks

Preliminary budget :-
1) Personnel (developers, testers, project manager) : $40,000
2) Software licenses & tools : $5,000
3) Hardware and hosting : $3,000
4) Training and support : $2,000
Estimated total : $50,000

Credit card Processing system

Problem statement:-

current Credit card transaction processing involves multiple intermediaries and manual verifications, leading to delay, security risks, and errors. there is a need for an automated, secure, and efficient system to authorize, process, and settle credit card payments in real-time.

general description:-

A secure, real-time system to authorize and process credit card transactions between merchants, issuing banks, and payment networks. It supports transaction validation, fraud detection, settlement, and reporting.

Functional Requirements:-

1) User authentication and authorization for merchants and administrators
2) Real-time transaction authorization and validation
3) Fraud detection and prevention mechanisms.
4) Transaction settlement and clearing between banks
5) Refund and dispute management.
6) Reporting of transaction and settlement status.

Interface Requirements :-

1) Secure API for merchants to submit transaction requests
2) Interface with issuing banks and payment gateways
3) Admin dashboard for monitoring and managing transactions
4) Integration with external fraud detection services.

Performance requirements :-

1) Process up to 1,000 transactions per second
2) Authorization response within 2 seconds
3) System uptime of 99.9%
4) Real-time fraud detection with minimal false positives

Design constraints :-

1) Must comply with PCI-DSS security standards
2) Use encrypted communication (TLS/SSL) for all data transfers
3) Scalable to support growing transaction volumes.

Non-functional attributes :-

1) Security :- Data encryption, role-based access, audit trails
2) Reliability: High availability with failover and backup
3) Usability: Clear error messages and

easy-to-use admin UI

Compliance:- Adherence to finance and the data protection regulation

Preliminary schedule:-
Requirements gathering: 3 weeks
System Design: 4 weeks
Development: 10 weeks
Testing: 4 weeks
Deployment: 2 weeks
Total: 23 weeks

Preliminary budget:-
Development team: $70,000
Security audits & compliance: $10,000
Infrastructure & hosting: $5000
Training and support: $5,000
Estimated total: $90,000

Library Management

Manual library systems are inefficient, error-prone, and time-consuming, there is a need for an automated solution to manage book inventory, user activity, and transactions more accurately and efficiently.

Introduction:-

The library management system is a software application designed to manage the operations of a library efficiently. It provides features for managing book inventories, user registrations, book lending and returns, fines, and reports. It ensures smooth operation of library processes and improves the overall library experience for both librarians and patrons.

Purpose

The purpose of this document is to define the software requirements for the library management system. It serves as a guide for developers, testers, and stakeholders to understand the system's functionality, constraints, and performance criteria. The main goal is to automate manual library operations to improve speed, efficiency, and accuracy.

Scope

The library management system will:

1) Maintain records of books, users, and transactions
2) Handle book issues, returns and renewals.
3) Track overdue books and calculate fines.
4) Support user registrations and authentication.
5) Allow administrators to manage inventory and users.
6) Generate various reports for analysis.
The system will be accessible to two user types: Librarians/Administrators and Students/Users.

Overview
1) General system description and features
2) Functional and non-functional requirements
3) Constraints and assumptions
4) Performance requirements
5) Budget and schedule estimates

General Description
1. Product Perspective
The LMS is standalone web or desktop application with a relational database backend. It can be integrated with barcode scanners or RFID systems.

2. Product Functions
1) User authentication
2) Catalog management
3) Book search functionality

4) Issue/return/renewal tracking
5) Fine calculation
6) User profile and history
7) Reports generation (daily, monthly)

3. User Classes and Characteristics
1) Administrators: Full system access, manage books and users.
2) Users: Search books, view history, borrow/return.

4. Assumptions and Dependencies
1) The system requires internet access (for web version)
2) Users will have basic computer literacy
3) Barcode scanner is optional but recommended.

Functional requirements
1. User login/logout
- User must be able to login in securly with credentials
2. User Management
- Admins can add, delete, and update user information
3. Book Inventory Management
- Add/update/delete & search books
- View current books status
4. Issue/Return/Renew books
- Users can borrow and return books
- Admin tracks due dates and calculate fines
5. Fine Management
- System auto-calculates fines for overdue

6. Search and Filter Books
- Search by title, author, category, ISBN etc.

7. Generate Reports
- Daily/weekly/monthly reports on issued books, fines, etc.

8. Notifications
- Notify users of due dates, overdues, and announcements.

Performance Requirements
- The System should support up to 100 concurrent users.
- Average response time for queries should be < 2 seconds.
- System uptime should be 99% monthly.

Design Constraints
- Backend: MySQL / PostgreSQL
- Frontend: HTML/CSS/JavaScript
- Language: Python / Java / PHP / C
- Framework: Optional (eg Django, MFT, Laravel)
- The System must be accessible via standard web browsers

Non-functional requirements
1. Reliability
- System must perform consistently under normal load.

2. Security
- Role based access control
- Password encryption and secure authentication.

3. Maintainability
- Code must follow standard conventions for easy updates.

4. Scalability:
- Database and application should
with processing records.

5. Usability
- Clear, intuitive interface for non-technical users.

6. Portability
- Should open on Windows / Linux/ mac
- Should be mobile-responsive (for web

Preliminary Schedule		
Phase	Duration	Start date
Requirement Analysis	1 week	Sep 9, 2025
System Design	1 week	Sep 16, 2025
Implementation	3 weeks	Sep 23, 2025
Testing and Debugging	1 week	Oct 14, 2025
Deployment and training	1 week	Oct 21, 2025
Final review and delivery	2 days	Oct 28, 2025

Preliminary Budget	
Item	Estimated cost
Developer Salaries	$6,000 - $10,000
Software licenses	$200
Hosting / Domain	$100
Hardware	$300

Date: / /

Miscellaneous $150

Total Estimated Cost $6750-$10,700

Stock Maintance System

Problem Statement:
Manual stock management often leads to inaccurate inventory records, delays in stock updates, overstocking, or stock A digital Stock Maintance System is needed to automate inventory tracking, reduce errors, and ensure efficient stock control and decision-making.

Introduction:
The Stock Maintance System is designed to streamline and automate the process of tracking, managing, and updating stock levels in real time. It provides a centralized platform to monitor inventory, generate reports, and manage suppliers and transactions efficiently, helping businesses maintain optimal stock levels and improve operational efficiency.

Purpose:
This document defines the requirements for developing a stock maintance system. It is intended for stakeholders, developers, and testers as a reference to ensure the system meets business needs. The system will support inventory tracking, restocking, sales recording, and reporting.

Scope
track incoming and outgoing stock items

- Manage product information
- Alert for low stock levels
- Handle supplier and purchase order details.
- Provide real-time reporting and analytics.
- Support multi-user roles (admin, manager, staff)
- Be scalable for small to medium-sized businesses.

Overview

- System main features and architecture.
- Functional and non-functional requirements
- Development constraints and assumptions
- Timeline and estimated budget.

General Description

1. Product perspective

The system will be a standalone or web-based applications with a user-friendly interface and a backend database to store inventory records.

2. Product functions

- Product and category management
- Stock-in and stock-out operations
- Supplier and purchase order tracking.
- Low-stock and expiry date alerts.
- User authentication and access control.
- Reporting and analytics

3 User classes.

Admin: Full access to all system features.

Inventory Manager: Manage stock levels and transactions.
Staff: Views stock and process stock-in/out operations.

4. Assumptions & Dependencies.
- Stable internet for online version
- Users have basic system knowledge
- Barcode / RFID integration is optional

5. Functional Requirements
1. User Authentication
- Secure login/logout with role-based access
2. Product Management
- Add, edit, delete products and assign categories
3. Stock Transactions.
- Record stock-in and stock-out activities
- Update stock levels in real time
4. Low Stock Alerts
- Notify users when items fall below minimum quantity
5. Supplier Management
- Maintain supplier information and link with stock purchases
6. Reports
- Generate reports on current stock, transaction history, and reorder level
7. Search and Filter
- Search stock by name, category, or ID
8. Audit Logs
- Track all user actions for accountability

Performance Requirements
- Should support upto 50 concurrent users.
- Response time for stock updates must be under 2 seconds
- The system should process 1000 inventory items efficiently.

Design constraints
- Backend: MySQL / PostgreSQL
- Frontend: React / Angular
- Programming language: Python / Java
- Platform: Web-based or desktop
- Data backup must be scheduled daily

Non-Functional Attributes
1. Security
Encrypted passwords, role-based access control, secure database
2. Usability
Easy-to-use interface with minimal training
3. Reliability
System must be available 99.5% of the time
4. Maintainability
Modular code for easy updates and debugging.
5. Scalability
Should be scale to handle growing inventory and users
6. Portability
Web version should support major browsers. Desktop version should be able to run on Windows and Linux.

paperang

Date: / /

Preliminary schedule

Phase	Duration	Start date	End date
Requirement Gathering	1 week	Sep 9, 25	Sep 15, 25
System Design	1 week	Sep 16, 25	Sep 22, 25
Development Phase	3 weeks	Sep 23, 25	Oct 13, 25
Testing & QA	1 week	Oct 14, 25	Oct 20, 25
Deployment & training	1 week	Oct 21, 25	Oct 27, 25
Final Review & Handover	2 days	Oct 28, 25	Oct 30, 25

Preliminary Budget

Item	Estimated Cost
Developers	$5000 - $9000
Hosting / Server	$100
Database and Backup setup	$200
Optional Hardware	$300
Miscellaneous	$150
Total Estimated Cost	$5,750 - $9,750

Passport Automation System

Problem Statement

The manual passport application and processing system is time-consuming, error-prone, and inefficient. There is a need for an automated system that simplifies the passport application process, reduces human intervention, improves transparency, and accelerates passport issuance.

Introduction

The Passport Automation System is a software application designed to automate the entire process of applying for and processing passports. It enables officials to submit applications online, upload necessary documents, track their application status, and receive timely updates. For government officials, it simplifies data verification, application management, and passport approval processes.

Purpose

Streamline the passport application and verification process.

Eliminate paperwork and manual data entry.

Enhance efficiency, transparency, and user experience.

Provide an interface for both applicants and passport office staff.

Scope

- Allows users to register and submit passport applications online.
- Enable uploading of identity and address proof
- Facilitate appointment scheduling for document verification.
- Provide tracking and status updates.
- Include an admin panel for processing applications, verifications, and approvals.
- Be scalable for use in regional and national passport centers.

Overview

- System architecture and models
- Features for both end-users and passport staff
- functional and performance requirements
- Development and Deployment plan
- Budget and time estimates

General Description

1. Product Perspective

This is a web-based platform with two main user roles: Applicants and Passport Office staff. It will interface with third party APIs for identity verification and payment gateways.

2. Product Functions

- Users registration and login
- Online passport application form
- Document upload and verification
- Application status tracking

- Admin dashboard for processing
- Notifications via Email / SMS

3. User Classes
- Applicants: Apply for passports, upload documents or track status
- Passport Staff / Admin: Verify applications, schedule appointments, approve or reject applications

4. Assumptions & Dependencies
- Applicants have Internet access and valid documents.
- Staff will have credentials and secure access
- Integration with government databases.

Functional requirements
1. User Registeration & Authentication
- Secure signup & login with OTP or email verification
2. Online Application Submission
Fill out online passport application form
3. Document Upload
Upload scanned copies of required documents
4. Application Tracking
Users can view their application status in real time
5. Appointment Scheduling
Book appointment slots for in-person verification
6. Admin Panel
View, verify, and manage applications.

Update status and provide approval rejections

7. Notifications
Email / SMS notifications for key status

8. Report Generation
generate report for submitted, approved, rejected, and pending applications

Performance Requirements
system should support upto 1000 concurrent users.
Applications submissions and updates should be processed in <3 seconds.
Applications uptime must be 99.9% monthly.

Design Constraints
Backend: MySQL
Frontend: React / Angular
Language: Java / Python
Hosting: Cloud-based (AWS, Azure, etc.)
Must Comply with data privacy regulations

Non-Functional Attributes
1. Security
Secure data storage (encryption)
HTTPS, user authentication

2. Usability
clean, interactive interface for both users & staff

3. Reliability:
Minimal downtime; automated backups

4. Maintainability
modular design for easy maintenance and future updates

Scalability:
can scale to handle national
- less books.

Maintainability:
should work on major browsers and
mobile devices

Preliminary Schedule

Phase	Duration	Start Date	End Date
Requirement gathering	1 week	Sep 9, 25	Sept 15, 25
System Design	1 week	Sep 16, 25	Sept 22, 25
Development Phase	4 weeks	Sep 23, 25	Oct 20, 25
Testing & QA	1 weeks	Oct 21, 25	Oct 27, 25
Deployment & training	1 week	Oct 28, 25	Nov 3, 25
Final reviews & delivery	2 days	Nov 4, 25	Nov 5, 25

Preliminary Budget

item	Estimated cost (USD)
Developer Hours (2 devs, 2 months)	$8,000 - $12,000
Server Hosting & Domain (1 year)	$200

SMS / Email Notification services	$100
Security audit & testing	$400
Miscellaneous	$200
Total Estimated cost	₦9000 - $13,0