19/8/25

Page No:
Date.

> Hotel Management System

Problem statement: The hospitality industry faces significant challenges in efficiently managing hotel operations which include reservations, guest services, room allocation, billing and housekeeping. Thus, there is a need for an integrated, automated Hotel Management System that streamlines the operations.

Introduction:

- Purpose: The main objective is to provide a base for the foundation of the project. It provides view of how system is supposed to work and what is to be expected by the end users. Client's expectation and requirements are analyzed to produce specific unambiguous functional and non-functional requirements so they can be used by development team with clear understanding to build a system as per end user needs.
- Scope: The project is intended for the reservations for room that can be made online. The system will be centralized, user-friendly and robust solution for managing hotel operations efficiently. It covers three major functionality.
  - Reservation management
  - Front desk and guest services
  - Room and housekeeping management
  - Billing and payment processing
  - User management.

Page No:
Date.

- Overview: This SRS document includes all the functional and non-functional requirements for the HMS. It is structured into sections that define the system's overall goals, detailed functional and non-functional requirements and design considerations.

General description:

→ Product Perspective

The HMS is an integrated solution designed to streamline hotel operations, including reservations, guest management, room assignments and housekeeping. It will be a web-based platform with a mobile app.

→ Product Features

- Reservation Management
- Guest Management
- Room Management
- Housekeeping Management
- User Management

→ User classes and Characteristics

- Administrator
- Front Desk staff
- Guests
- Housekeeping and Maintenance.

Functional Requirements:

→ Reservation Management

Description - The system shall allow guests to book rooms, view availability and cancel or modify bookings

Use case - Guest can search for available rooms

Requirements:
- Support for single and group reservations
- Automated email confirmations.

→ Guest Management
Tracks guest profiles, preferences and history for personalized service
→ Room Management
Display room availability and status and assign rooms
→ Billing and Payments
Generates invoices, processes payments and integrates with payment gateways
→ User Management
Admin can manage user roles and access.

Interface requirements
→ User Interfaces
- Web interface: Responsive design
- Mobile App: iOS/Android
- Guest Portal

→ Hardware interfaces
- POS Systems: Integration with payment
- Keycard Systems
- Room Devices

→ Software interfaces
- Payment Gateways
- Sync with platforms like Airbnb
- CRM Systems.

Performance Requirements
- Response Time: System should load pages within 3 seconds
- Concurrent Users: System should support 500 simultaneous users
- Transaction Processing
- Data Storage
- System availability

Design Constraint
- Compliance: Must adhere to regulations
- User interface: Should be intuitive
- Scalability: Must support large chains
- Integration: Needs to integrate with third-party systems.

Non-Functional Attributes
- Performance: Pages should load within 3 seconds, support 500 users
- Security: Encript sensitive data with GDPR
- Usability: Intuitive interface with minimal training
- Scalability: Must handle growing data
- Availability

Preliminary Schedule and Budget
Schedule
Phase 1: Requirements gathering & system design — 2 months
Phase 2: Development & integration — 4 months
Phase 3: Testing and deployment — 2 months
Phase 4: Training and go-live — 1 month

Budget
Development: $150,000
Integration: $30,000
Testing: $20,000
Training and support: $10,000

> Credit Card Management System

Problem Statement: The system addresses inefficiancies, security risks and scalability issues in managing credit card transactions and accounts. It will streamline card issuance, real-time processing, fraud detection and reporting

Introduction:
- Purpose: The purpose of this document is to define software requirements for a Credit Card Processing System (CCPS). The system will handle authorization, authentication, transaction routing, settlement and reporting.

- Scope: The CCPS will:
- Support credit card payments
- Provide real-time authorization and settlement
- Include fraud detection and prevention mechanisms
- Offer API interfaces for third-party integration.

Overview
The system consists of
- Merchant interface: To initiate transaction
- Payment Gateway: To securely route requests
- Settlement Module: To transfer funds between banks.

General Description:
- Users: Merchants, Customers, Banks, Admins
- Environment: Web & Mobile Apps, secure APIs, integration with banking networks
- Constraints: Must comply with PCI DSS, GDPR.
- Assumptions: Users have valid credit cards.

Functional Requirements:
1. User Authentication: Merchants and admins must log in securely
2. Credit Card Authorization: Verify credit card, available balance, expiration date
3. Transaction Processing: Approve or decline transactions, support refunds
4. Fraud detection: Detect unusual patterns
5. Settlement: Transfer approved funds
6. API Integration: REST APIs for third-party platforms.

Interface Requirements
- User Interface (UI): Web dashboard
- API Interface: RESTful APIs with JSON response
- External Interface: Integration with card networks.

Performance Requirements
- Process 10,000 transactions per second.
- Authorization response time < 2 seconds
- 99.99% uptime

Design Constraints
- Must follow PCI DSS compliance
- Database must support ACID properties
- Support scalability on cloud platforms

Non-Functional Attributes
- Security: End-to-end encryption.
- Reliability: High availability with failover
- Scalability: Support millions of users.
- Usability: Simple UI for admins

Preliminary Schedule

Phase	Duration (weeks)	Deliverables
> Requirement analysis	2	SRS
> System Design	3	Architecture
> Development	8	Core Modules
> Testing	4	Unit, Integration
> Deployment	2	Production release
> Maintenance	Ongoing	Updates

Total Estimated Time: ~19 months.

Budget

> Development Team	$1000,000
> Cloud Infrastructure	$20,000
> Licensing & Compliance	$15,000
> Testing & QA	$10,000
> Maintenance	$25,000
Total	$170,000

> Library Management System

Problem statement
Many educational institutions and public libraries still manage their resources manually, leading to inefficiency, errors in record-keeping, difficulty in tracking issued/returned books. A digital library is required to automate cataloging, book issue/return, accuracy and accessibility.

Introduction
- Purpose: Purpose is to define functional and non-functional requirements for a library management system. This helps administrators, librarians and students.

- Scope:
The LMS will:
- Maintain a digital catalog of all books & resources
- Support member registration and authentication
- Enable book search, issue, return
- Automatically calculate & manage fines
- Be accessible via web and mobile interfaces.

- Overview: The system consists of:
  - Admin/Librarian Module: Add/remove books
  - User Module: Search catalog, view history
  - Database: Stores books, members
  - Reports Module: Summarizes book circulat° & member activity.

General description
- Users: Librarians, Students/Members
- Environment: Web-based interface
- Constraints: Must work with limited internet bandwidth.
- Assumptions: Users have valid library membership.

Functional Requirements
1. User Authentication: Login/registration for members and staff
2. ~~Data~~ Book Catalog Management: Add, edit, remove book entries.
3. Book Issue/Return: Issue books to members, support book renewal.
4. Search & Browse: Search by title, author, subject, ISBN
5. Fine Management: Auto-calculate fines for overdue returns.
6. Reports & Analytics: Track most borrowed books & active users.
7. Notifications

Interface Requirements
- User Interface: Web dashboard and mobile-friendly design.
- API Interface: REST APIs for integration with student portals.
- External Interfaces: Barcode scanners, student database.

Performance Requirements
- Support 1,000+ concurrent users
- Book search results displayed in < 1 seconds
- Issue/return transactions
- 99% uptime requirement

Design constraints
- Database must support ACID transactions
- Must comply with privacy regulations
- Secure authentication.
- Should run on standard platforms.

Non-functional Attributes
- Security: Role-based access, encryption of user credentials.
- Reliability: Backup and recovery support
- Scalability: Can expand to include e-books
- Usability: UI both for librarians & students

Preliminary Schedule

Phase	Duration
Requirement Analysis	2 weeks
System Design	2 weeks
Development	6 weeks

Testing 3 weeks
Deployment 2 weeks
Maintenance Ongoing
Total Estimated Time: ~15 weeks

Preliminary Budget:
- Development Team $60,000
- Cloud Infrastructure/ Servers $8,000
- Database licensing & Tools $5,000
- Testing & QA $7,000
Total Estimated Budget $80,000

> Stock Maintainence System

Problem Statement:
Organisations face difficulties in tracking stock levels, avoiding overstocking/understocking, and managing inventory efficiently. A Stock Maintenance System is required to automate stock tracking, monitor availability and generate alerts.

Introduction:
- Purpose: To digitally manage stock items ensuring real-time updates on availability, incoming and outgoing stock, and automated notifications for restocking
- Scope:
  - Maintain a digital inventory of items
  - Track incoming (purchases) and outgoing stock.
  - Generate alerts for low stock levels.
  - Provide role-based access
- Overview: Components include:
  - Stock Database: Stores items details, quantities
  - Transaction Module: Records incoming/outgoing stock.
  - Alert System: Triggers notifications for critical stock levels.

General Description:
- Users: Admin, Staff
- Environment: Web or desktop application
- Constraints: Secure access, consistent data accuracy.
- Assumptions: Items are uniquely identifiable

Functional Requirements:
1. Add/Edit/Delete stock items
2. Record stock purchase & consumption.
3. Auto-update current stock levels.
4. Generate daily/weekly/monthly stock reports
5. Set minimum stock threshold & trigger alerts.

Interface Requirements:
- UI: Web dashboard with graphs & tables
- API: Integration with sales/purchase system
- External

Performance Requirements:

- Handle 100,000+ products
- Support 1,000 concurrent users
- Transact" update within 2 seconds

Design Constraints
- Must comply with data integrity rules
- Should app. support multi-locat" warehouse
- Should run on standard web/cloud infrastructure.

Non-functional Attributes
- Security: Role-based access, encrypted data storage.
- Reliability: Auto-backup & recovery
- Scalability: Expandable to multi-branch systems.
- Maintainability: Modular design

Preliminary Schedule

Requirements	2 weeks
Design	2 weeks
Development	6 weeks
Testing	3 weeks
Deployment	2 weeks
Total:	~15 weeks

Preliminary Budget

Development Team	$55,000
Cloud Infrastructure	$3,000
Testing & QA	$5,000
Maintainence	$10,000
	$77,000

> Passport Automation System

Problem Statement
Passport processing is traditionally slow, manual and prone to errors, causing delays and inefficiency. An automated system is required to digitize applicat" submission, verificat", approval & passport issuance.

Introduction
- Purpose: To define requirements for a Passport Automation System that streamlines passport applicat", verificat", status tracking & issuance.
- Scope:
- Allow applicat applicants to apply online and upload documents
- Support document verificat" & approval by officials
- Provide status tracking for applicants
- Enable appointment booking for verifica
- Ensure data security & compliance with government rules.
- Overview:
The System includes:
Applicant Module - Registration, application submission, tracking
Admin/Official Module - Document verificati approvals
Database - Store applicant and passport records
Notification Module - SMS/Email updates

General Description:

- Users: Applicants, Passport Officials, Admins
- Environment: Government web portal, Secure servers.
- Constraints: Must comply with government data policies.
- Assumptions: Applicants provide valid details and documents

Functional Requirements:

1. User Registration/Login - Secure applicant access
2. Online Application - Form submission with documents.
3. Document Verification - Officials verify ID, address proof, etc.
4. Appointment Booking - For physical verification/interview.
5. Status Tracking - Applicants can view application progress
6. Passport Issuance - Generate & update passport record.

Interface Requirements:

- UI: Responsive web portal
- API: Integration with national ID database.
- External Interfaces

Performance Requirements:

- Handle 10,000+ applications per day
- Applicant Application response time < 3 seconds

Design Constraints

- Must comply with e-Governance, GDPR, and local regulations
- Strong authentication
- Use AES-256 encryption for data.

Non-Functional Attributes

- Security: Two-factor authentication, encrypted storage
- Reliability: Failover support, daily backups
- Scalability: Can expand to millions of users
- Maintainability: Easy updates.

Preliminary Schedule

Requirements	3 weeks
Design	3 weeks
Development	10 weeks
Testing	5 weeks
Deployment	3 weeks
Total	~ 24 weeks

Preliminary Budget

Component	Cost (USD)
Development Team	$ 150,000
Secure infrastructure	$ 40,000
Testing & Security Audits	$ 20,000
Maintenance	$ 30,000
Total	$ 240,000.