Software Requirement Specification (SRS)

Hotel Management System

Problem Statement

Hotels often face challenges in managing bookings, check-ins, check-outs, and billing when done manually. This results in errors like double booking, loss of records, and customer dissatisfaction. The problem is the absence of an efficient, automated system that can handle hotel operations smoothly.

1. Introduction

1.1 Purpose of this Document

The purpose of this SRS document is to describe the Hotel Management System in detail. It defines the system's objectives, requirements, and functionality, which will help the development team and also act as a reference for testing.

1.2 Scope of this Document

The Hotel Management System will automate tasks such as reservations, customer check-in/check-out, staff management, billing, and reporting. This reduces manual work, increases efficiency, and ensures customer satisfaction.

1.3 Overview

The HMS is a user-friendly application that handles hotel activities. It consists of modules for room booking, billing, customer records, and reporting. The system reduces errors and increases efficiency for hotel management.

2. General Description

The HMS is meant for receptionists, hotel managers, and administrators. It provides features such as viewing available rooms, managing staff, handling payments, and generating reports. The objective is to reduce paperwork and improve accuracy.

3. Functional Requirements

1. Customer Registration
2. Room Booking and Cancellation
3. Check-in and Check-out Processing
4. Billing and Payment Management
5. Staff Management
6. Report Generation (daily, monthly, yearly)

4. Interface Requirements
The system will provide a GUI for users with options like booking, billing, and staff management. It will also include a backend database (MySQL) for storing records.

5. Performance Requirements
The HMS should process bookings quickly and allow multiple users at the same time. It should handle up to 5000 records efficiently, and responses should be under 2 seconds.

6. Design Constraints
1. Developed using Java/Python and MySQL.
2. Should meet data security standards for customer details.
3. Should run on Windows and Linux OS.

7. Non-Functional Attributes
1. Security: Protect customer data with encryption.
2. Reliability: Should be available 24/7 with 99% uptime.
3. Scalability: Must support hotel expansion.
4. Usability: Easy for staff to use.
5. Portability: Compatible with different platforms.

8. Preliminary Schedule and Budget
Development is estimated to take 4 months with a budget of $20,000. This includes design, coding, testing, and deployment.

Credit Card Processing System

Problem Statement

In today’s world, most payments are done using credit cards. However, manual processing of credit card payments causes delays, errors, and sometimes fraud. Customers face problems like declined transactions even when they have sufficient balance, while banks and merchants face the

risk of fraudulent activities. There is a need for a software-based system that can process credit card transactions automatically, securely, and quickly to improve customer experience and trust.

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to describe the Credit Card Processing System in detail. It explains the objectives, features, and functionalities that the system should provide. This document will act as a guideline for developers, testers, and stakeholders to design and implement the system correctly.

1.2 Scope of this Document

The system will automate the entire process of credit card transactions such as customer authentication, authorization, billing, and fraud detection. It will improve the speed and reliability of transactions, reduce manual work, and provide security against fraud. This system will be used by banks, merchants, and customers.

1.3 Overview

The Credit Card Processing System will allow merchants to process payments securely in real-time. It will include modules for customer authentication, authorization, billing, fraud detection, and reporting. The main goal of the system is to ensure accurate, fast, and secure handling of transactions.

2. General Description

The system will serve customers, merchants, and banks. Customers will use their credit card for payments, merchants will initiate the transaction, and the bank will authorize and settle it. The system will ensure quick verification, secure payment, and record maintenance. It will also provide reporting facilities for auditing and analysis.

3. Functional Requirements

1. Customer Authentication (PIN/OTP)
2. Transaction Authorization (check credit balance, validity)
3. Fraud Detection and Prevention (suspicious activity check)
4. Billing and Settlement between bank and merchant
5. Transaction History Management
6. Report Generation for banks and merchants

4. Interface Requirements

The system will provide a secure web interface for merchants and administrators. Customers will authenticate using OTPs or PINs during transactions. The system will also integrate with banking servers through secure APIs for authorization and settlement.

5. Performance Requirements

- The system should handle thousands of transactions per second.
- Response time for authorization should be less than 1 second.
- The system should ensure 99.9% uptime for continuous service.

6. Design Constraints

1. The system must follow **PCI-DSS** security standards.
2. Encryption methods such as **AES** and **RSA** must be used.
3. The system must integrate with different banks' existing servers.

7. Non-Functional Attributes

1. **Security**: Transactions must be encrypted and safe.
2. **Reliability**: System must not fail during high transaction loads.
3. **Scalability**: Should support increasing number of users and merchants.
4. **Usability**: Interface should be simple for merchants and banks.
5. **Compatibility**: Must work with different banking and merchant systems.

8. Preliminary Schedule and Budget

The development of the system is expected to take **6 months** with an estimated budget of **$50,000**. This includes requirements gathering, design, coding, security testing, and deployment.

Library Management System

Problem Statement

In many colleges and institutions, library management is still carried out manually using registers and paper records. This causes problems such as loss of records, difficulty in tracking issued/returned books, and delay in updating student accounts. Students often face inconvenience in finding whether a book is available or not. Therefore, there is a need for a computer-based Library Management System that automates the entire process of book issue, return, and catalog maintenance.

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to provide a detailed description of the Library Management System (LMS). It defines the system objectives, requirements, and features in order to help developers and testers in creating a complete and efficient application.

1.2 Scope of this Document

The LMS will automate the functions of a library such as book cataloging, member registration, book issue/return, fine calculation, and report generation. The system will save time for librarians, reduce errors, and make it easier for students to access information. The users of the system will include librarians, students, and administrators.

1.3 Overview

The system will provide a central database to maintain records of all books, members, and transactions. It will also provide search facilities to check book availability. The application will make the library operations faster, more reliable, and more transparent.

2. General Description

The system will be used by:

- **Students**: To search for books and check their library accounts.
- **Librarians**: To manage book issue/return and maintain records.
- **Administrators**: To oversee overall functioning and generate reports.

The objective of the system is to provide a user-friendly solution for managing all library-related tasks in a single platform.

3. Functional Requirements

1. Member Registration and Management
2. Book Catalog Management (add, update, delete books)
3. Book Issue and Return Management
4. Fine Calculation for late returns
5. Search Facility for books by title/author/category
6. Report Generation (books issued, overdue books, fines collected)

4. Interface Requirements

- The system will provide a **Graphical User Interface (GUI)** for librarians and students.
- Students will be able to log in to check their account status.
- Database connectivity (e.g., MySQL) will be used to store all library records.

5. Performance Requirements

- The system should handle up to **10,000 book records** efficiently.
- It should process book issue/return transactions in less than **2 seconds**.
- It should support multiple users accessing the system at the same time.

6. Design Constraints

1. Should be developed using Java/Python with a relational database.
2. Must follow basic security rules for student and book records.
3. Should work on college computer labs (Windows/Linux systems).

7. Non-Functional Attributes

1. **Security**: Only registered users can access the system.
2. **Reliability**: Records should not be lost or corrupted.
3. **Scalability**: Should handle increasing number of books and students.
4. **Usability**: Easy-to-use interface for students and librarians.
5. **Portability**: Should run on multiple operating systems.

8. Preliminary Schedule and Budget

The development of the LMS is expected to take **3 months** with a budget of **$15,000**. This includes requirement gathering, design, development, testing, and deployment.

Stock Maintenance System

Problem Statement

In many shops, warehouses, and organizations, stock management is still handled manually using registers or spreadsheets. This often leads to errors such as incorrect stock counts, overstocking, or shortage of items. It also becomes difficult to generate reports and track sales or purchases efficiently. To solve these problems, there is a need for a computerized Stock Maintenance System that automates inventory tracking, stock updates, and reporting.

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the requirements for the Stock Maintenance System (SMS). It provides details about the system's objectives, features, and functionalities so that developers and testers can design and implement the application effectively.

1.2 Scope of this Document

The Stock Maintenance System will help organizations keep accurate records of items available, purchased, or sold. It will allow managers to track stock levels, generate reports, and predict shortages in advance. The system will reduce human error, save time, and increase efficiency in managing inventory.

1.3 Overview

The SMS will maintain a central database of stock items, purchases, and sales. It will include modules for updating stock, managing suppliers/customers, and generating reports. The system will support real-time stock tracking and ensure that the data is always accurate and up to date.

2. General Description

The system will be used by:

- **Warehouse/Store Managers**: To update stock and track inventory.
- **Employees**: To enter sales and purchase records.
- **Administrators**: To generate detailed stock and sales reports.

The system's objective is to provide a fast and reliable way to maintain stock information and reduce manual paperwork.

3. Functional Requirements

1. Item Registration (add/update/delete stock items)
2. Purchase Entry and Update in stock
3. Sales Entry and Stock Reduction
4. Alert System for Low Stock Levels
5. Supplier and Customer Management
6. Report Generation (stock status, sales, purchases, profit/loss)

4. Interface Requirements

- The system will have a **GUI** for users to add/update/view stock.

- The backend database (e.g., MySQL) will maintain stock records.
- Managers will have access to generate reports, while employees will have limited access.

5. Performance Requirements

- The system should support up to **20,000 stock items**.
- It should process updates within **2 seconds**.
- It should allow multiple employees to work simultaneously without conflicts.

6. Design Constraints

1. Should be developed using Java/Python with a database like MySQL.
2. Must follow basic security measures to protect inventory records.
3. Should run on standard desktop computers with Windows/Linux OS.

7. Non-Functional Attributes

1. **Security**: Only authorized users can modify stock.
2. **Reliability**: System should not lose data even during crashes.
3. **Scalability**: Must support future expansion of items and users.
4. **Usability**: Easy to use for employees with minimal training.
5. **Portability**: Compatible with different operating systems.

7. Preliminary Schedule and Budget

The Stock Maintenance System is expected to take **4 months** to develop with an estimated budget of **$18,000**. This includes requirement analysis, design, coding, testing, and final deployment.

Passport Automation System

Problem Statement

The traditional passport application process is often slow and inefficient due to manual handling of applications, long queues, and paperwork. Applicants face problems such as delays in approval, loss of documents, and lack of transparency in the process. Similarly, authorities struggle to manage large volumes of applications effectively. To overcome these issues, there is a

need for a **Passport Automation System** that digitizes the entire process, making it faster, secure, and more transparent.

---

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to describe the requirements of the Passport Automation System (PAS). It defines the objectives, features, and system functionalities that will help developers and testers build the application in an efficient and reliable manner.

1.2 Scope of this Document

The Passport Automation System will allow applicants to submit applications online, upload required documents, track the status of their application, and receive notifications. For authorities, the system will provide tools for verifying documents, processing applications, and issuing passports. The system will reduce paperwork, save time, and provide better services to citizens.

1.3 Overview

The PAS will consist of modules for application submission, verification, payment processing, and status tracking. The system will ensure security of applicant data, reduce manual effort, and provide transparency throughout the process.

2. General Description

The system will be used by:

- **Applicants**: To apply for a passport, upload documents, and check application status.
- **Passport Officers**: To verify documents and approve/reject applications.
- **Administrators**: To oversee the entire system and generate reports.

The main objective of the system is to digitize and simplify the passport issuance process.

3. Functional Requirements

1. Online Application Submission with document upload
2. Payment Gateway for application fees
3. Verification Module for officers to check documents and details
4. Status Tracking and Notifications for applicants
5. Appointment Scheduling for interviews/verification

6. Report Generation for administrators

4. Interface Requirements

- A **web-based interface** for applicants to apply online.
- A secure portal for officers to verify and process applications.
- Database connectivity (e.g., MySQL/Oracle) to store applicant records and documents.

5. Performance Requirements

- The system should handle **thousands of applications per day**.
- The response time for each request should be **under 2 seconds**.
- The system should be available **24/7** for applicants.

6. Design Constraints

1. Must follow government rules and policies for passport applications.
2. Must ensure data security using encryption techniques.
3. Should support integration with government ID databases.

7. Non-Functional Attributes

1. **Security**: Sensitive applicant data must be encrypted.
2. **Reliability**: The system should have 99.9% uptime.
3. **Scalability**: Must support future increases in number of applicants.
4. **Usability**: Easy to use for applicants with minimal computer knowledge.
5. **Portability**: Compatible with web browsers on multiple platforms.

8. Preliminary Schedule and Budget

The Passport Automation System is expected to take **6 months** for development with an estimated budget of **$60,000**. This covers requirements gathering, design, coding, testing, security validation, and deployment.