

## LAB 1

- Q. Develop a complete IEEE standard SRS document with several requirements.

### HOTEL MANAGEMENT SYSTEM

#### 1. Introduction

a. Purpose: This document outlines the functional & non-functional requirements for the Hotel Management System. It will serve as a guide for developers, testers & stakeholders to understand the expected features & performance of the system. The goal is to automate hotel operations & improvement customer service.

#### b. Document Conventions

The document follows IEEE, all headings are bold & properly numbered for easy reference.

Requirements are labeled as functional, non-functional or external interface.

#### c. Intended Audience & Reading suggestions

This SRS is intended for developers, testers, project manager, & hotel administrator. Developers should focus on the requirements section. Testers should refer to functional & interfaces while managers & clients may focus on scope & user roles.

#### d. Project Scope

The HMS will manage hotel operations such as reservations, room availability, customer check-in/out, staff scheduling, billing & reporting.

### c. References

- IEEE 830 Software Requirements

MySQL by Java Spring documentation

- APIs for online payments by email notifications

### d. Overall Description

#### a. Product Perspective

The HMS is a standalone application but may be extended to integrate with payment gateway, email services. It is intended for both local use by anyone access via a browser or client applications.

#### b. Product Functions

Main functions include room reservation, check-in/check-out, billing, etc. It also allows administrative control for staff management, room configuration.

#### c. User Classes & Characteristics

Administrator: full access to all modules of the system.

Receptionist: can manage bookings and billings.

Housekeeping Staff: can view & update room cleaning status.

External customer: can book rooms online through a portal.

Customer: can book rooms online through a portal.

#### d. Operation Environment

The system will run on Windows or Linux platforms.

It uses MySQL for the backend by Java on a web interface for the frontend.

e. Design & Implementation constraints -  
The system should use only open-source technology.  
It must be scalable, secure, & support integration with external services.

#### f. User documentation -

User manuals & admin guides will be provided with include system navigation, features, FAQs & troubleshooting steps to help users interact with the system easily.

#### g. Assumptions & Dependencies -

Assumes users have basic computer knowledge. Online features depend on internet availability & third-party APIs. The system also depends on the correct setup of the hosting environment.

### 3. Specific Requirements -

#### a. Functional Requirements -

- Users can book, modify & cancel reservations
- Receptionist manages check-ins & check-outs
- System generates & prints bills
- Admin can add/edit rooms & view reports
- Email confirmations are sent after booking

#### b. non-functional requirements -

- System should respond within 2 sec.
- 99.9% uptime is expected
- Secure login & encrypted data storage
- Must support at least 20 users at the same time.
- Interface should be user-friendly & fast.

- cc. External Interface Requirements
- API for admin, staff & customers
  - Integration with payment gateways
  - Email server for sending notifications
  - Optional barcode / ID scanner support at reception

Appendices

Glossary of terms used in this document.

Screenshots / mockups of interfaces.

- Use Case, ER & DFD diagrams
- Sample test cases for each feature

- Business logic

- Business logic constraint

and other terms to follow, had not been

defined yet so will be defined here.

Business logic constraint

is a rule that must be followed by

business logic and is determined by

the requirements itself or by

the requirements themselves.

Business logic constraint

is a rule that must be followed by

business logic and is determined by

the requirements themselves.

Decline

Reasons to decline:

1. No further budget available

2. Requirements are unclear

# CREDIT CARD PROCESSING (CSRST)

## I. Introduction

### a. Purpose

This system securely processes credit card payments, including validations, authorization, fraud checks, and transaction recording. It ensures fast, reliable, and compliant payment handling.

### b. Document Conventions

follows IEEE 830 format using clear section numbering & labels like PRR functional, non-functional external interface requirement.

### c. Intended Audience & Reading suggestions

for developers, testers, security analysts, & stakeholders. Each should focus on relevant sections like requirements, security features & performance criteria.

### d. Project Scope

The system handles online & offline credit card transactions, detects fraud, validates card details & provides transaction history to banks & merchants.

### e. References

IEEE 830, 2019 edition, version 1.0

- PCI DSS Guidelines
- VISA Merchant API Doc
- ISO Smart Card Standards

## 2. Overall Description

### a. Product Perspective

Acts as middleware between merchants, banks, card networks, payment gateways, and merchant platforms for secure transactions.

### b. Product Functions

Performs card validation, payment approval, fraud detection, transaction logging, reporting. Admin can review & manage merchant profiles.

### c. User Classes

- Merchant: Initiates & monitors transactions
- Cardholder: Makes payment requests to merchant
- Admin: Manages user by logs, negotiation of security team: monitor fraud alerts
- Security Team: monitor fraud alerts

### d. Operating Environment

Runs on secure servers using HTTPS, supports

web/mobile/POS system, uses encrypted

connections via (SSL/TLS)

Networks of certified or certified merchants, automatically

### e. Constraints

can't store sensitive card data without compliance  
Must integrate with API's from banks & card networks to ensure high performance

### f. User Documentation

Includes guides for merchants, admin, panels, API integration & security options for monitoring & compliance.

## 9. Assumptions

Requires internet, valid card network APIs, verified users, & third-party fraud detection systems for smooth operation.

## 3. Specific Requirements

### a. Functional requirements

- Validate and details of appearance/reject payments
- Record transactions by notifying users
- Detect & block suspicious activity
- Provide summary reports

### b. Non-functional requirements

- Response time less than 3 secs for transactions
- 24/7 System availability with 99.99% uptime
- End-to-end encrypted data transmission
- System must comply with PCI-DSS standards
- Scalable to handle peak load traffic

### c. External Interface Requirements

- Integration with banking / payment gateway APIs
- GWI dashboards for merchants / admins
- Supports POS system, web, & mobile interfaces
- Communication via REST APIs & secure sockets

### d. Appendix

- Glossary, timeline etc.
- We use sequence diagrams for transactions
- API request/response formats examples
- Sample test cases & validation criteria

Meeting  
19/8/25

# Stock Maintenance System

## 1. Introduction

Purpose: To manage & track stock digitally with real time updates.

Convention: "Shall used for requirements"

Audience: Managers, Developers, Testers, Staff

Scope: Automates Stock tracking & reports

& alerts

References: IEEE Std. 830, Inventory guide

## 2. Overall Description

Characteristics of the System will include:

Perspective - Standalone, future integration with POS / ERP

Product functions - Add / update / delete items, Stock alerts, reports, search (filter, role-based access)

Users: Admin (full control), staff (update reports), viewer (view only)

Environment: Windows / Linux, MySQL / Oracle DB, Web/Desktop

Documentation - User manual, online help

Assumptions: Users are computer literate, data entered correctly

### 3. Specific Requirements

#### 1. Functional Requirements

The system manages stock by adding, updating, deleting items, showing real-time availability, generating low stock alerts, reports, & keeping transaction logs with role-based access.

#### 2. Non-functional Requirements

It ensures fast response less than 2 sec scalability up to 10,000 items, security with authentication / encryption, user-friendly design, high uptime (99.9%), & easy maintenance.

#### 3. External Interface Requirements

The system provides a API with dashboards, connects to databases like MySQL, works on standard OS/ hardware, communicates via AWS & supports future POS/ ERP integration.

#### 4. Appendices

Glossary: Stock, Report, Threshold

Future enhancement: Mobile app, barcode scan, auto supplier ordering.

# Passport Management System

1. Introduction - ~~Introduction and background~~

Purpose: The system automates passport application, verification & issuance online to reduce delays of manual work.

Document convention: All requirements use the keyword shall, categories are functional, non-functional & interface.

Intended Audience by reading suggestions applicants, government offices, develops, tests & manages each should focus on their relevant sections.

Project scope: The system makes online applications, payments, tracking, & verification of final passport delivery in a secure manner.

References: IEEE std 830, official govt. E-Governance policies & existing passport guidelines.

2. Overall Description

Product Perspective: A web-based solution replacing manual processes; designed for integration with govt databases.

Product Functions : users apply online, upload documents, pay fees, schedule appointments, track status, officials issue passports.

User classes & characteristics : Applicants submit requests, officials check & approve documents, & admins control the system.

Operating environment - works on windows/linux server with MySQL | Oracle DB accessible via connection with browser through API.

Design & Implementation constraints : must follow strict data protection, security standards, & handle large user traffic.

User documentation : provides user manual, online help & FAQ's for applicants & officials

Assumptions & Dependencies : user have internet access & valid documents, system depends on secure payment & verification services.

### 3. Specific Requirements :

1. Functional Requirements : Applicants can register, apply, upload, pay & track, applications, officials verify, update, & issue passports.

## 2 Non-functional Requirements

The system shall be fast (≤ 2 secs response), secure, reliable, scalable, by easy to use 24/7.

## 3 External Interface Requirements

Provides browser-based GUI, connects to database, integrates with payment / Revision System over Internet / LAN.

## 4 Appendices

Glossary : Applicant, citizen user, User, Govt office, States, current stage of passport processing, etc.

Future Enhancement - Integration with mobile app, biometric verification, AI based fraud detection.

## 1. Introduction

- 1) Purpose - The system automates library operations including book issue, return, cataloguing, & members management to save time & reduce errors.

## 2. Document Conventions:

All the requirements use the keyword shall, categories are FK, NFR, IFR, MFR, etc.

## 3. Intended Audience by reading suggestions:

Librarians, students, faculty, developers, users of project manage showed focus

## 4. Project scope -

The system manages book records, member accounts, transaction by queries, reports, vehicle, ensuring of easy search & access.

## 5. References:

IEEE std 830, library automation standards by guidelines such.

## 2. Overall Description -

### 1. Product Perspective -

A standalone of web-based solution replacing manual systems, with options for future integration with digital libraries, loans, fines, etc.

2. Product Functionality: Includes adding / removing books, issuing / returning, managing members, generating reports, & providing search functionality.
3. Use classes by characteristics:  
librarians manage books & members  
faculty by students borrow books &  
admin oversee the system
4. operating environment: Works on  
Windows / Linux with MySQL  
& accessible via web browser.
5. Design & Implementation constraints:  
must ensure data consistency, prevent  
multiple issue of the same copy.
6. User Documentation: Includes manual  
for staff, FAQs for students &  
training guides
7. Assumptions & Dependencies - Classes  
have valid library accounts, Systems  
depend on reliable database

### 3. Specific Requirements

Functional Requirements - The system shall allow book issue

MARCH 22/23

Feature: Manage due dates & fines,  
maintain member accounts & generate  
reports.

2. Non-functional Requirements - It shall  
provide responses within 2 days, ensure  
security, maintain 99.5% uptime &  
offer a user friendly interface.

3. External Interface Requirements -  
Provides a GUI with search by email &  
menus. connects to external databases &  
communicates over LAN. Retains if  
outage.

#### 4. Appendices

Glossary : member = student / faculty,  
librarian = staff, Transaction = issue / return

~~future enhancement~~: Barcode / QR report  
e-book access, mobile app, auto fine  
payment

## CLASS DIAGRAM

## I Hotel Management System

<b>Hotel</b>	<b>Room</b>
+ hotel-id: int	+ room-id: int
+ name: String	+ room-num: String
+ address: String	+ type: String
+ phone: String	+ status: String
+ add-room()	+ price: double
+ gets-available-room()	+ check-availability()
	+ update-status()

<b>Staff</b>
+ staff-id: int
+ name: String
+ phone: String
+ task: String
+ perform-duty()

<b>Reservation</b>
+ rid: int
+ check-in-date: Date
+ check-out-date: Date
+ status: String
+ coupon-reserve
+ cancel-reserve

<b>Receptionist</b>
+ qualification: String
+ check-in-guest()
+ check-out-guest()

<b>Manager</b>
+ dept: String
+ generate-report()

<b>Payment</b>
+ id: int
+ amount: double
+ date: Date
+ method: String
+ process-payment()

<b>Guest</b>
+ id: int
+ name: String
+ address: String
+ phone: String
+ email: String
+ mark-reservation
+ book-service

<b>Service</b>
+ service-id: int
+ type: String
+ cost: double

# Credit card processing

Customer	
id: int	
name: string	
email: string	
phone: string	
linkcard()	
View transactn history()	
( )	

Credit card
no.: string
type: string
expiry date: Date
validate card()
getcard details()

Merchant	
id: int	
name: string	
area	
tr: int	
slab: float	
stock: float	
rate: float	
( )	
initialrepayment()	
refund()	
( )	

Transaction
id: int
amount: double
date: date
status: string
authorizetransaction()
updatestatus()

Bank	
id: int	
name: string	
branch: string	
verify account()	
( )	

## payment gateway

id: int
name: string
process payment()
check status()

Issuing Bank	
Card limit: int	
approvemethod()	
( )	
void transaction()	

Acquiring Bank
accno: int
settrepayment()

## Fraud Detection System

id: int
risklevel: int
analyzetransaction()

# Smart Library Management System

## Library

id: int
name: String
address: String
addBook()
getAvailableBook()

## Staff

id: int
name: String
phone: String
roll: String
performDuty()

## Book

id: int
title: String
author: String
ISBN: int
status: String
updateStatus()
getBookInfo()

## Reservation

id: int
date: Date
status: String
cancelReservation()
confirmReservation()

## Loan

id: int
issueDate: Date
dueDate: Date
returnDate: Date
calculateFine()
markAsReturned()

## Librarian

qualification: String
addMember()
removeMember()
getMemberList()

## Assistant

shiftTiming: String
arrangeBooks()

## Member

id: int
name: String
email: String
no: String
borrowedBook()
returnedBook()

## Fine

id: int
amount: double
duedate: Date
status: String
payFine()

# STOCK MANAGEMENT SYSTEM

## Inventory

id: int  
 location: string  
 add-product()  
 getStockDetails()

## Staff

id: int  
 name: string  
 role: string  
 performDuty()

## Product

id: int  
 name: string  
 price: double  
 quantity: int  
 category: string  
 updateStock()  
 getStockDetails()

## Manager

dept: string  
 generateReport()

## Clerk

shift: string  
 updateInventory()

## Stock Transaction

id: int  
 date: Date  
 quantity: int  
 type: string  
 recordTransaction()

## Order

id: int  
 Date: Date  
 status: string  
 placeOrder()

## Customer

id: int  
 name: string  
 phone: string  
 purchaseProduct()

## PASSPORT AUTOMATION SYSTEM

User
id : int
name : string
DOB : Date
Address : string
Age : int
email : string
register()
login()
authentication()

Passport
no : int
issue date : Date
name : string
nationality : string
DOB : date
expiry date : date
get details()
renewal()

Auth. system
id : int
user id : int
pass no : int
authentication()
get details()

Staff
id : int
dept : string
work()
help user()

Biometrics
fingerprint
password : string
face-recognition
match biometric()

Verification office
id : int
name : string
designation : string
approve()
reject()

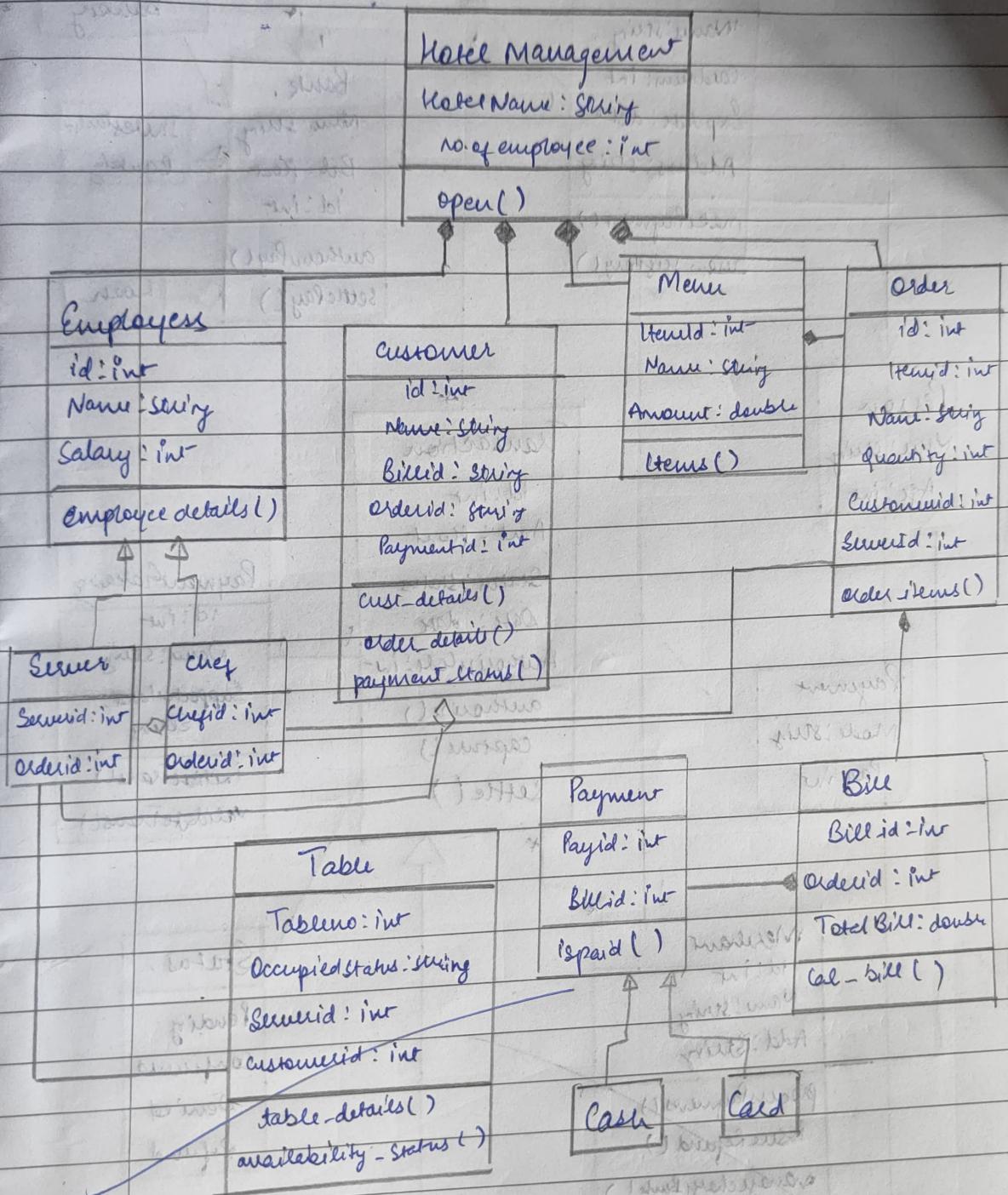
Customer care
id : int
name : string
designation
help()

User profile board
record id : int
user id : int
pass no : int
receipt date : date
update system()
set record()

Payment
user id : int
bank : string
amount : double
pay()

# HOTEL MANAGEMENT SYSTEM

## Advanced class diagram



# CREDIT CARD PROCESSING

Date \_\_\_\_\_  
Page \_\_\_\_\_

## WTF2D TRANSACTION

### Customer

Name: string  
Cardnum: int  
Expdate: date  
Address: string  
makePayment()  
viewHistory()

### Bank Account

(initialization)  
(update)  
(query)

### Commercial banking

Bank,  
Name: string  
Bal: float  
id: int  
authenPay()  
settlePay()

### Investment Banking

### Loans

### Person

Name: string  
Age: int

### Transaction

id: int  
Amt: float  
Status: status  
Date: date  
AuthenCode: int

### Payment Gateway

id: int  
Name: string  
SupportCard: string  
initiateTrans()  
validateTrans()

### Payment

Mode: string  
Paytm

authorise()

capture()

settle()

### Status

Pending  
Confirmed  
Denied  
Refund

### Merchant

Id: int  
Name: string  
Add: string  
processPayments()  
issueRefund()  
manageBank()

### Authorisation

Expiry date: Date  
Fraud check: Boolean  
validateFunds()  
Check AVV()

### Refund

Amt: float  
Reason: string  
initiateFund()  
notifyCust()

6/9/16