18/8/2025

Lab-1

→ Develop a problem statement and complete IEEE SRS document with several requirements.

1> Hotel management system

Problem statement: Develop a hotel management system that handles reservations, checkin/check-outs, payments, staff management, ~~and~~ and reporting efficiently.

SRS document:

1. Introduction

1.1 Purpose of this document

The purpose of this document is to define the requirements for the development of a Hotel Management System. This system will automate hotel operations such as reservations, checkin/out, billing, staff management, food ordering, taxi/cab services, customer queries and reporting. It serves as a reference for developers, testers, project managers, stakeholders throughout the project life cycle.

1.2 Scope of this document

This system will,

- Automate hotel operations like room booking, billing, staff scheduling, guest services
- Provide online/offline booking facilities with secure payment integration.
- Enable food ordering from rooms/restaurant, cab/taxi bookings, service requests.
- Allow guests to submit feedback and queries.
- Generate reports on occupancy, revenue, staff performance, customer satisfaction

- Provide role-based access for Admin, Receptionist, Staff, Guest.
- Be scalable from small to large hotels, with support for multi branch operations.

1.3 Overview
The system will provide user-friendly web application for hotel operations. It will integrate room management, services, billing payments, and reporting to a single platform.

2. General description
→ Users : Admin, Receptionist, Staff (housekeeping, restaurant, transport) guests.

→ Operating environment : Website with cloud / server deployment
→ Assumptions : Internet connectivity is available to users.
→ Dependencies : Payment gateways, SMS/email API, cab service.

3. Functional requirements
3.1 Room & reservation management
→ Add, update, delete room details
→ track availability (vacant, booked, ~~ac~~ under maintainence).
→ booking cancellation, modification.
→ SMS/email confirmation messages.

3.2 Guest check-in/out.
→ register/update guest details, assign room & give key.
→ settle bills and update room status
3.3 Billing & Payments
→ auto genarate invoices, multiple payment modes, history.
3.4 Staff, Food and Cab services
→ assigning staff duties, track attendance, scheduling, payroll.
→ guest order food, track order, update billing.
→ guest request taxis, interate cab services, billing.
3.5 Customer feedback and queries
→ Guests raise queries, can submit feedback and rate.

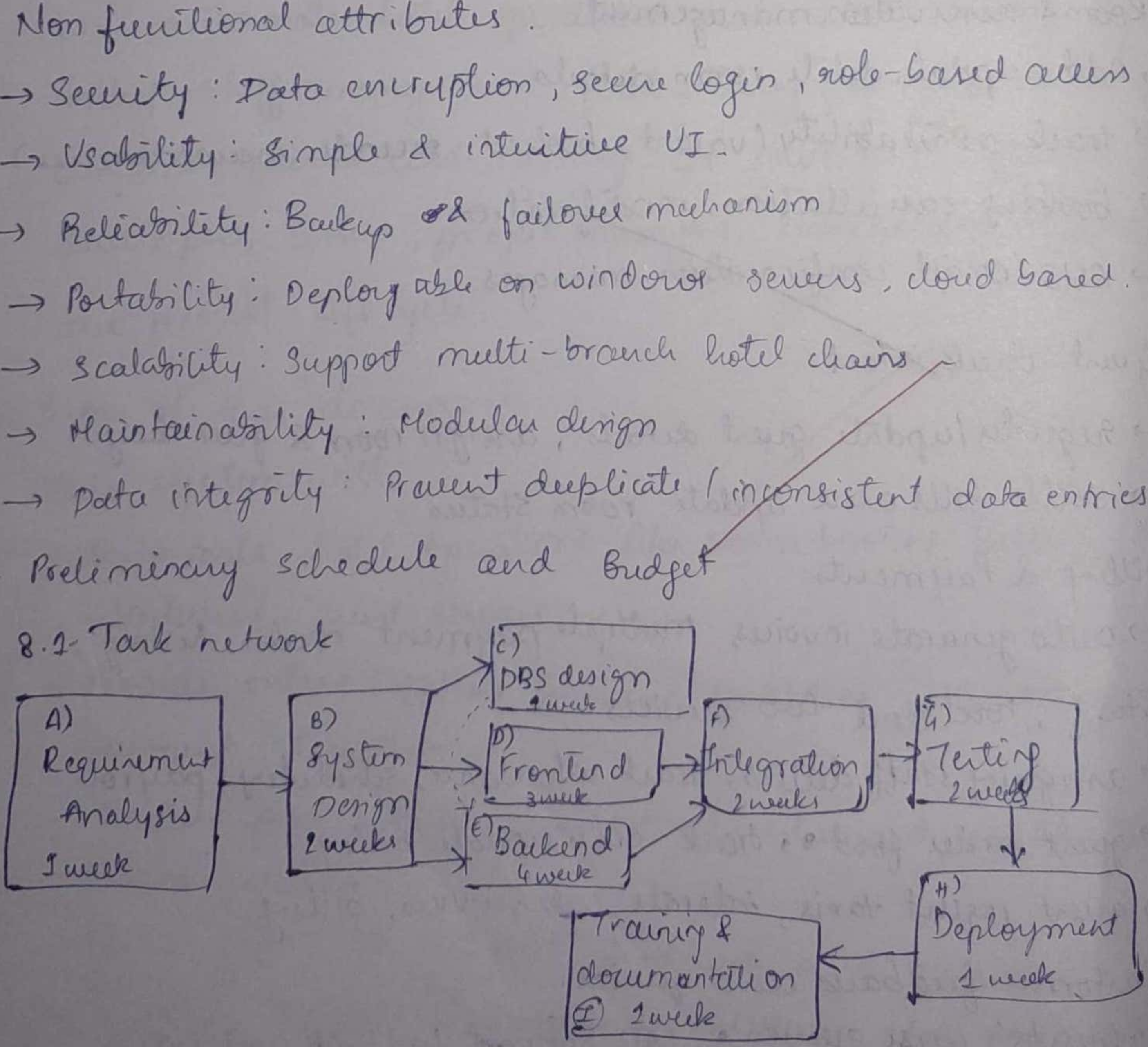
4. Interface requirements
→ UI :
  • Responsive design, multi-language support
  • Dashboards for admin, receptionist, staff, guest.
→ System interface :
  • Payment gateway, SMS/email API for notification

5. Performance requirements:
  • Handle 500+ users efficiently
  • Response time < 3s for major operations.
  • Database storage : 5+ years of guest & financial records.
  • Uptime : 99.5% or higher

6. Design constraints :
  • Backend : Java / Python / PHP with MySQL / PostgreSQL
  • Frontend : HTML, CSS, JS
  • Browser : Chrome / Firefox / Edge / Safari
  • Security : PCI DSS compliance for payments.

7. Non functional attributes.
→ Security : Data encryption, secure login, role-based access
→ Usability : Simple & intuitive UI.
→ Reliability : Backup & failover mechanism
→ Portability : Deployable on windows servers, cloud based.
→ Scalability : Support multi-branch hotel chains
→ Maintainability : Modular design
→ Data integrity : Prevent duplicate / inconsistent data entries.

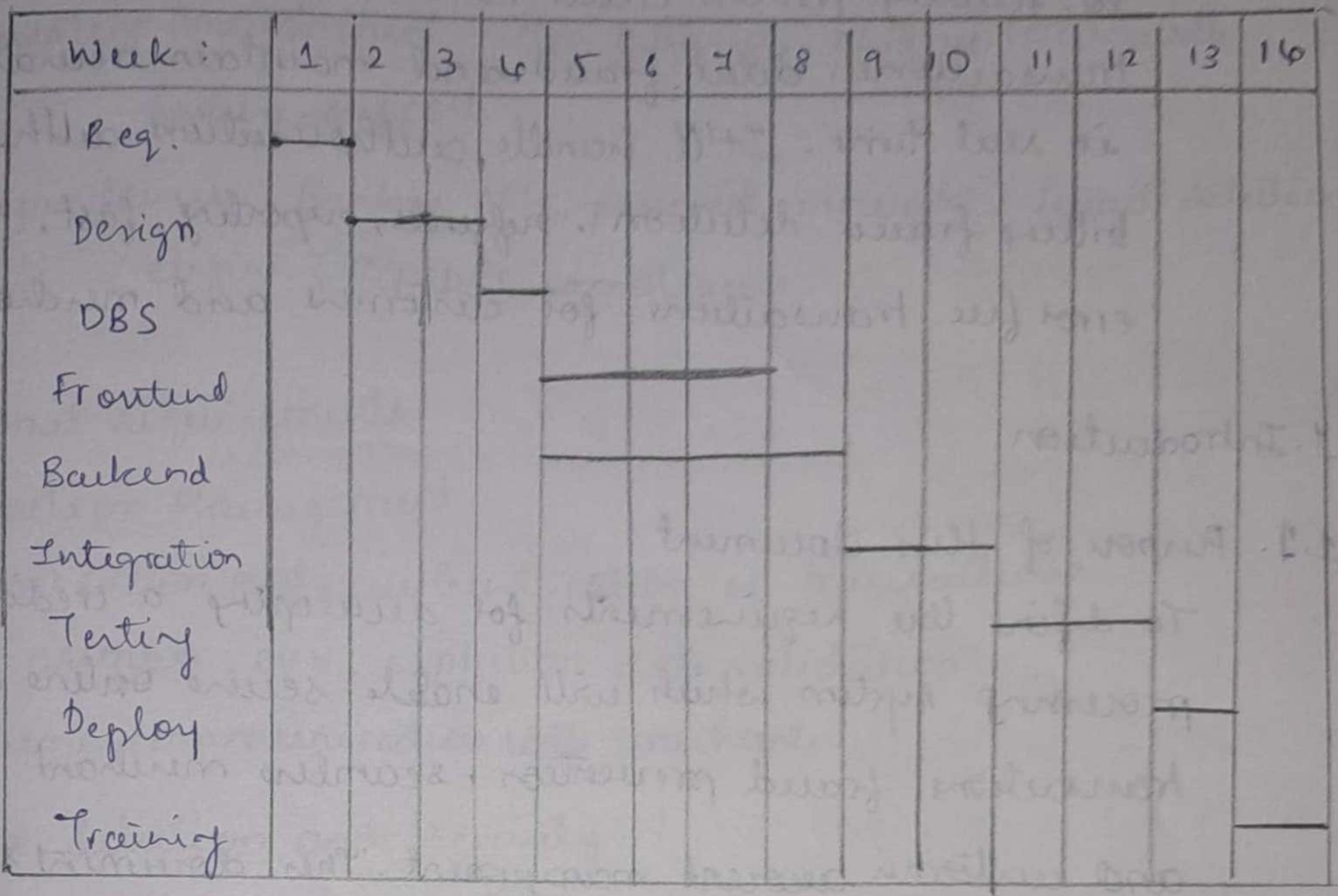
8. Preliminary schedule and Budget
8.1 Task network



Critical path: A → B → C → E → F → G → H I.

Total duration ~ 16 weeks

8.2 Timeline Chart



8.3. Effort estimation & cost calculation.

Total LOC ≈ 10,300

Productivity rate ≈ 2500 LOC/person-month

Effort = LOC / productivity = 10300/2500 = 4.12 ≈ 5 person-month

// Labour cost.

Total cost = effort * cost per-person month

= 5 × 75,000 = ₹3,75,000.

27 Credit Card Processing System

Problem statement: We are developing a system that is automat to securely process credit card payments, verify transactions, detect fraud and maintain accurate record in real time. It'll handle authorization, authentication billing fraud detection, refunds, reporting, fast, secure, error free transactions for customers and merchants.

1. Introduction

1.1. Purpose of this document

To define the requirements for developing a credit card processing system which will enable secure online ~~and~~ transactions, fraud prevention, seamless merchant integration and customer account management. This document serves as a guideline for all stakeholders.

1.2 Scope of this document

The system will:

- Process credit card transactions (authorization, authentication)
- Verify card details and detect fraudulent activities.
- Support payments across multiple merchants.
- Provide services like, statements, refunds, dispute handling
- Ensure compliance with PCI DSS and banking standards

1.3 Overview

This system will act as a secure payment gateway and transaction processor between customer, merchants and banks. It will integrate with banking networks, ensure encrypted communication for all transactions.

2. General Description

→ Users : customers (credit card holders), merchants (businesses accepting card payments), banks, admins.

→ Operating environment : Cloud based, 24/7 uptime with failure support.

→ Dependencies : Banking APIs, payment networks, fraud detection engine ; PCI DSS compliance.

3. Functional requirements

3.1 Transaction Management

- authorization and authentication of transactions.
- card number, CVV, expirition date validation
- real time communication with merchants.

3.2 Fraud detection and security

- detect suspicious transactions
- OTP/2FA authentication
- Block stolen or invalid cards
- Encrypted transaction logs

3.3 Merchant Services

- APIs ~~with~~ for integration with merchant websites, real time payment confirmation, refund and cancellation.

3.4 Customer services

- Secure login for customers, transaction history, monthly statements, dispute handling, notifications

3.5. Admin and reporting

- Dashboard for system monitoring, Fraud alerts, compliance checks, revenue reports.

4. Interface requirements
→ UI : Web portal for customers & merchants, responsive, multi-language
→ System interface :
· Integration with Visa, MasterCard, RuPay, ~~R~~ Amex networks
- Bank API integration

→ Communication interface :
· Encrypted communication (SSL/TLS), EMail/SMS alerts.

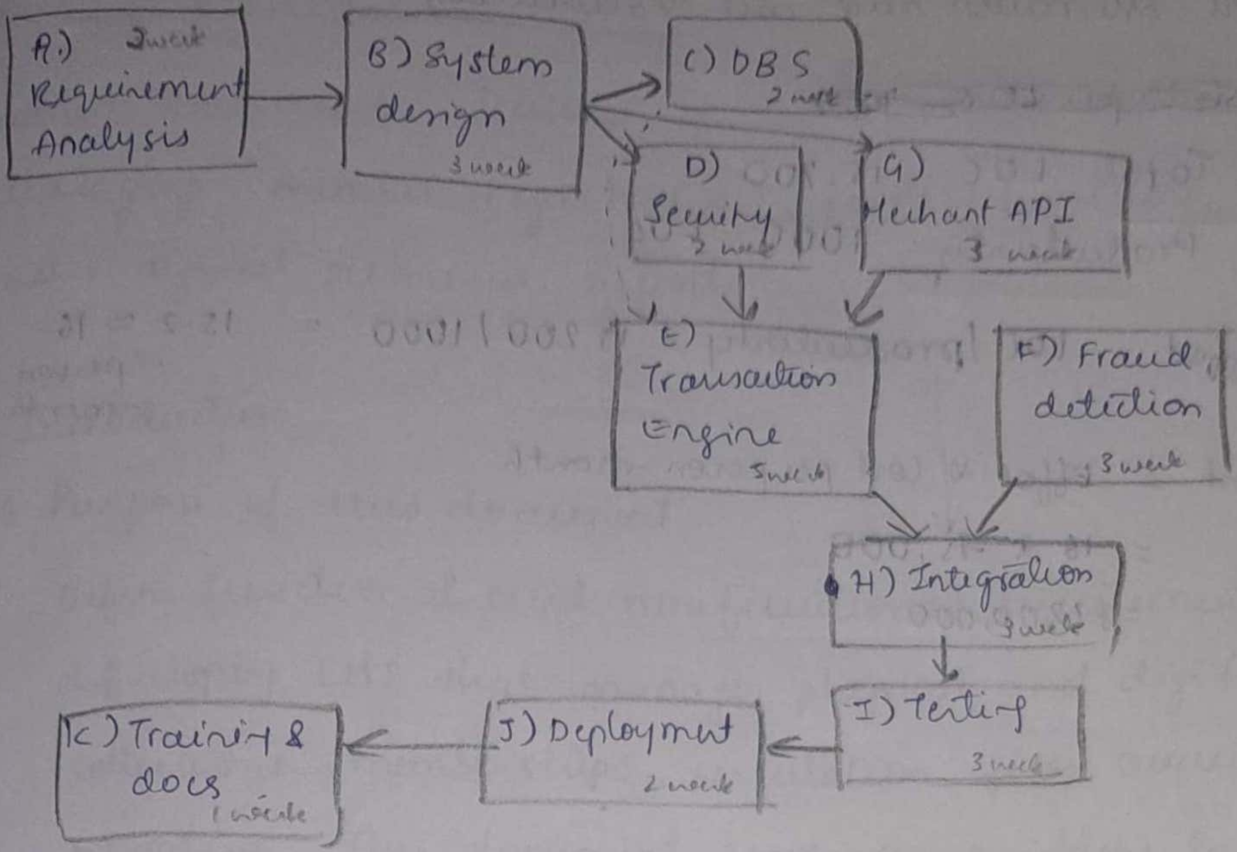
5. Performance Requirements
· Must handle atleast 50000 transactions per minute
· response time < 2s , system uptime : 99.99%
· fraud detection system accuracy > 98%

6. Design constraints
· Must comply with PCI DSS, ISO 8583 standards
· Backend : Java / python + frameworks
· Database : Encrypted SQL/NoSQL
· Compatible with major browsers and mobile platforms.
· Strict regulatory compliance with RBI / International banking laws.

7. Non-functional attributes
· Security : End-to-end encryption, tokenization, firewall.
· Scalability : Supports millions of users and transactions.
· Reliability : Fraud-tolerant with automatic failover.
· Usability : Simple interfaces for customer and merchant
· ~~Q~~ Portability : Cloud-ready, deployable across regions.
· Auditability : Full transaction logs for regulators.

8. Preliminary schedule and budget

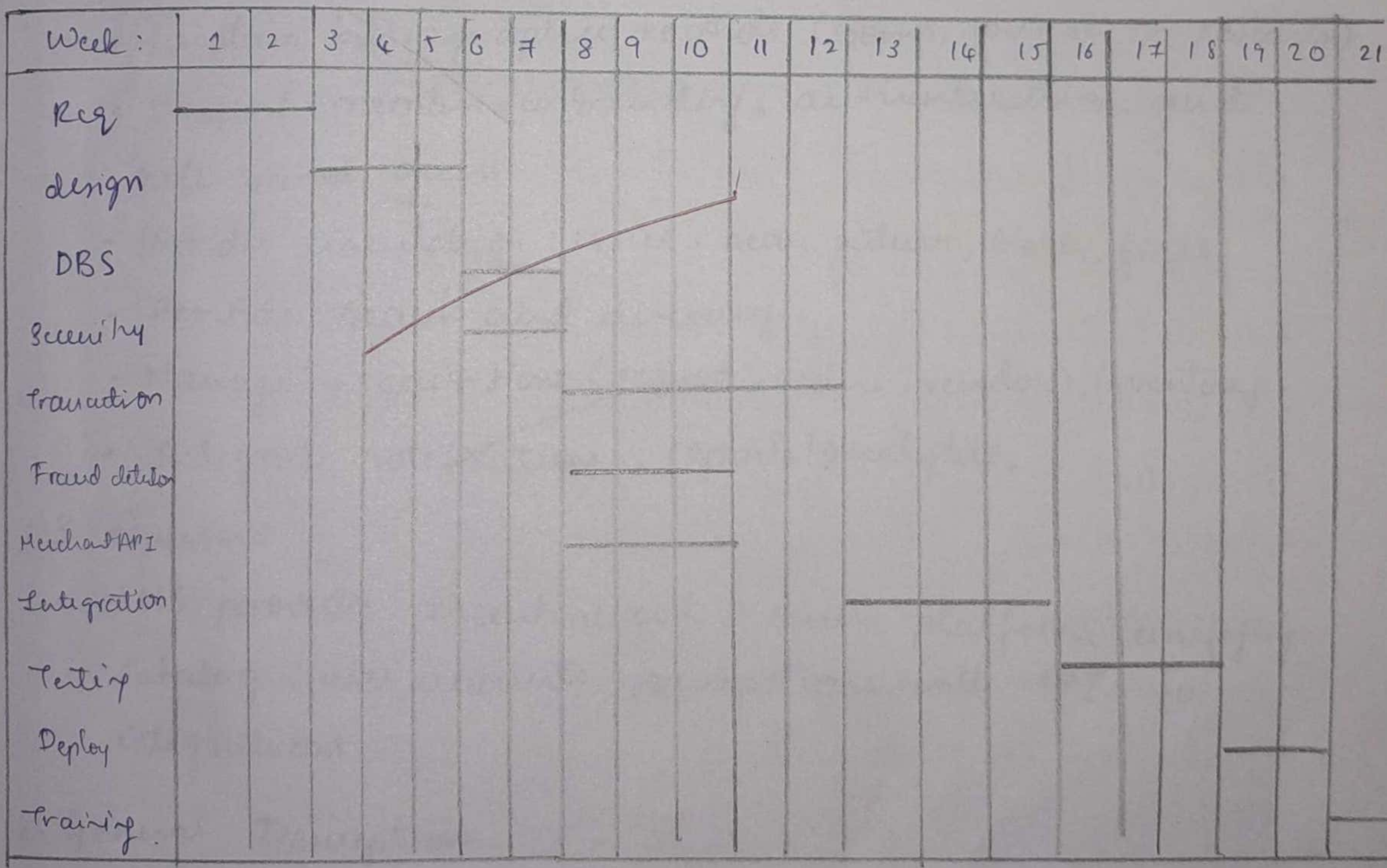
8.1. Task Network



Critical path : A→B→C→E→H→I→J→K.

Total duration ≈ 26 weeks

8.2 Timeline Chart



8.3 Effort Estimation and cost calculation

~~[illegible]~~

Total LOC 15,200

Productivity : 1000 LOC

Effot = LOC /productivity = 15200 / 1000 = 15.2 ≈ 16 person-month

Cost = Effot * cost per person-month

= 16 * 75,000

= ₹12,00,000

3) Library Management System

Problem statement : Develop a system ~~that~~ [illegible] to manage cataloging, member registration, lending / returns, reservations, fines, digital resources, reports.

1. Introduction

1.1 Purpose of this document

Define functional and nonfunctional requirements for developing LMS that manages physical and digital collections, memberships, circulation, fines, acquisitions, reporting. This document serves as guideline to all stakeholders.

1.2. Scope of this document

This system will

- Maintain bibliographic records (books, journals, e-resources).
- Support member onboarding, authentication, and role-based access
- Handle circulation : issue, new, return, holds, fines.
- Provide search and discovery.
- Manage acquisitions (request, orders, vendors), inventory.
- Integrate notifications, reports/analytics.

1.3 Overview

LMS provides a centralized, secure platform unifying catalog, user accounts, transactions with APIs for integrations.

2. General Description

→ Users : Admin : system config, policies, roles.
Librarian : cataloging, acquisition, circulation, fines.
Library assistant : circulation desk operations.
Member : search, place holds, borrow/renew, fines, history

→ Operating environment :
- Web app / browser (chrome, firefox, edge, safari)
- server (Linux, windows, cloud ready)
- DB (MySQL / Postgre SQL).

→ Assumptions & dependencies
- Reliable network for online ops ; offline fall back
- Email / SMS notification , payment gateway, barcode / RFID devices.

3. Functional Requirements.

3.1 Catalog management
- Add / edit / delete bibliographic records with ISBN, title, authors, subjects, classification, tags.
- Manage item / copy records
- Import / export details
- Attach cover images, TOC.

3.2 Member management
- Register members, verify identity, assign categories..
- Manage fees, deposits, memberships validity, account status,

3.3 Search and discovery.
- Basic / advanced Search on title / author / subject
- Relevance ranking, autocomplete

3.4 Circulation
- Issue, renew, return items, compute due dates per policy
- Reservations / Holds with pickup deadline and queing
- Fine and fees : overdue, damage / lost

3.5 Acquisitions & vendors
- Title suggestion / requests , purchase order, receiving invoicing

3.6 Policies and Security.
- Access based on role, borrowing limits , holidays , fine rules
- Data backup , encryption at rest & in transit.

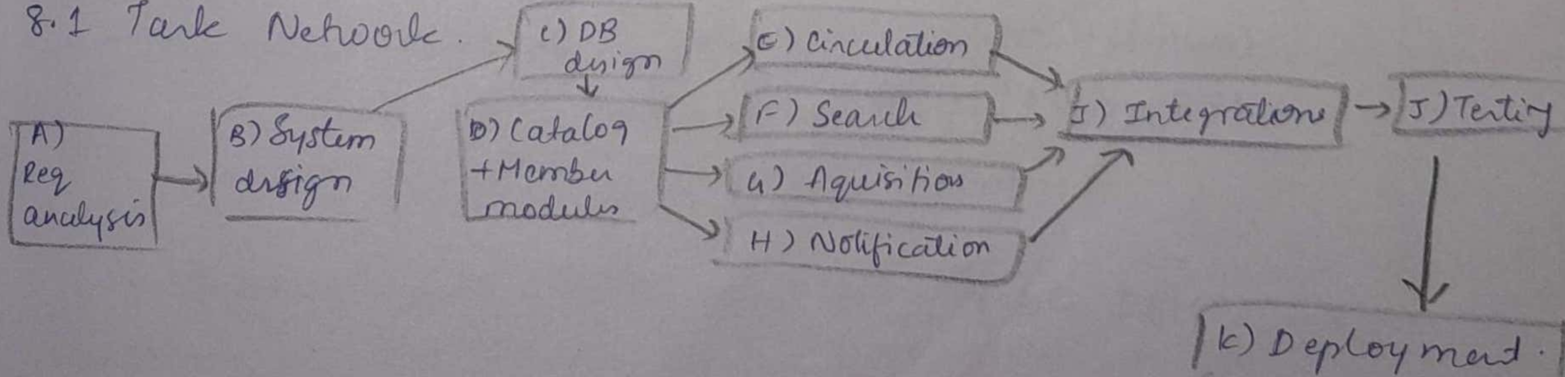
4. Interface requirements
→ UI : Responsive, key board shortcuts, dashboard per role.
→ System interfaces :
· Email/SMS gateway, payment gateway for fines
→ Hard interfaces:
· Barcode scanners, reciept printers.

5. Performance requirements.
- concurrency : ≥ 300 users
- response time ≤ 2s for search, ≤ 1s for checkout/return
- 99.5% availability
- database scale : ~~1M~~ ≥ 5,00,000 records.

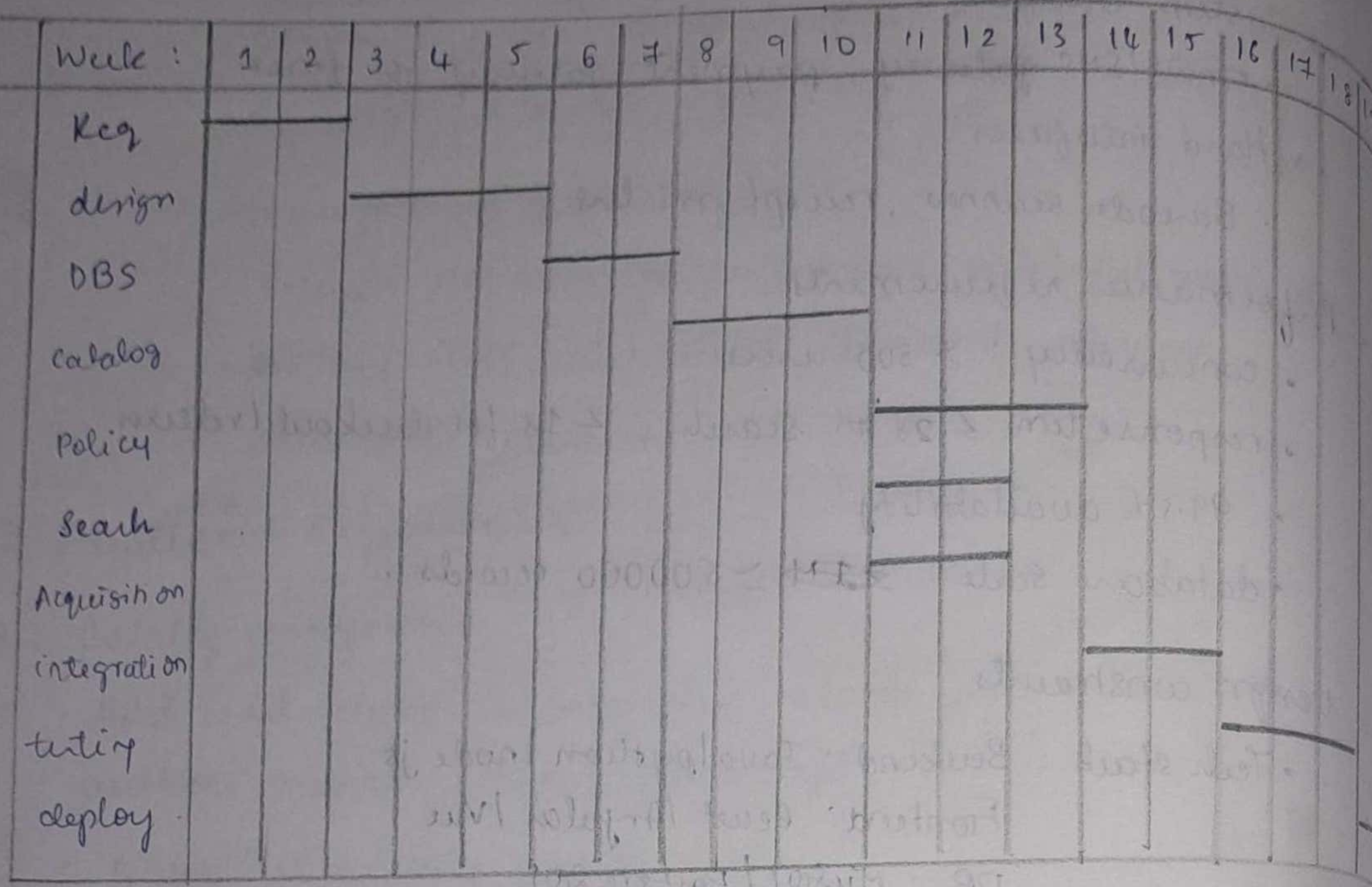
6. Design constraints
- Tech stack : Backend : Java/python/node.js
  Frontend: React/Angular/Vue
  DB : MySQL/PostgreSQL
- Privacy & compliance: local data-protection rules.

7. Non-functional attributes
- Security : TLS, hashed passwords, IP/device restrictions
- Usability : minimal clicks for desktops.
- Reliability : Automated backups, replication, health checks.
- Maintainability : modular services, clean APIs.
- Scalability : scalable to bigger libraries.
- Portability : containerised deployments.
- Data integrity : constraints & transactions to prevent inconsistent circulation states.

8. Preliminary schedule and Budget.
8.1 Task Network.



8.2 Timeline chart



8.3 Effort estimation and cost calculation

Total LOC = 10600

Productivity = 1000100

Effort = LOC / productivity = 10.6 ≈ 11

cost = Effort $*$ cost per person mont

= 11 $*$ 75000

= ₹8,25,000

4) Stock maintainence system

Problem statement: Develop a system to track stocks, monitor sales, prevent shortages, avoid overstocking, integrated with billing. System allows multi-user access, real-time updates, customer query handling.

1. Introduction
1.1. Purpose of this document
This document defines the requirements for Stock MS that helps businesses manage inventory, suppliers, sales. This serves as guidelines to all stakeholders

1.2. Scope of the system
This system will allow businesses to
- maintain centralised inventory database
- provides features of stock entry, updates, deletions.
- record purchases, transactions. generate reorder alerts.
- Ensure data security, reliability and backup.

1.3 Overview
This system will provide an intuitive user interface for staff and managers, while ensuring secure data storage & ~~rea~~ real time updates.

2. General Description
→ Users : Admin : manages users, system settings, high level reports
Staff : handle day to day stock entry, sales, supplier info
manager : monitor updates, authorizes reorder.
→ Operating environment: Server OS (windows / linux)
Database : MySQL / PostgreSQL
Client : Web browser

→ Dependencies : Reliable internet, stable DBMS.

3. Functional Requirements

3.1 Stock management

- add new stock, update stock levels after purchase / sale
- Remove discontinued item

3.2 Supplier management

- maintain supplier details, generate purchase order and track

3.3 Sales and Billing integration

- record sales transaction, integrate with POS system for automatic stock detection.
- generate bills and reciepts for customers.

3.4 Reorder alerts

- automatically generate reorder alerts for low-stock items
- notify about stock expiry

3.5 Reports and analytics

- generate sales report
- view inventory status

3.6 User and role management.

- Multi-user login with access control.

3.7. Search & query system.

- check stock by ID, name or category.
- query suppliers and sales record.

4. Interface requirements.

- UI : Intuitive role based dashboards, forms for sales, entries, repo
- ~~Sys~~ Hardware : Support barcode scanner / POS hardware.
- Software : Database connectivity

5. Performance requirements

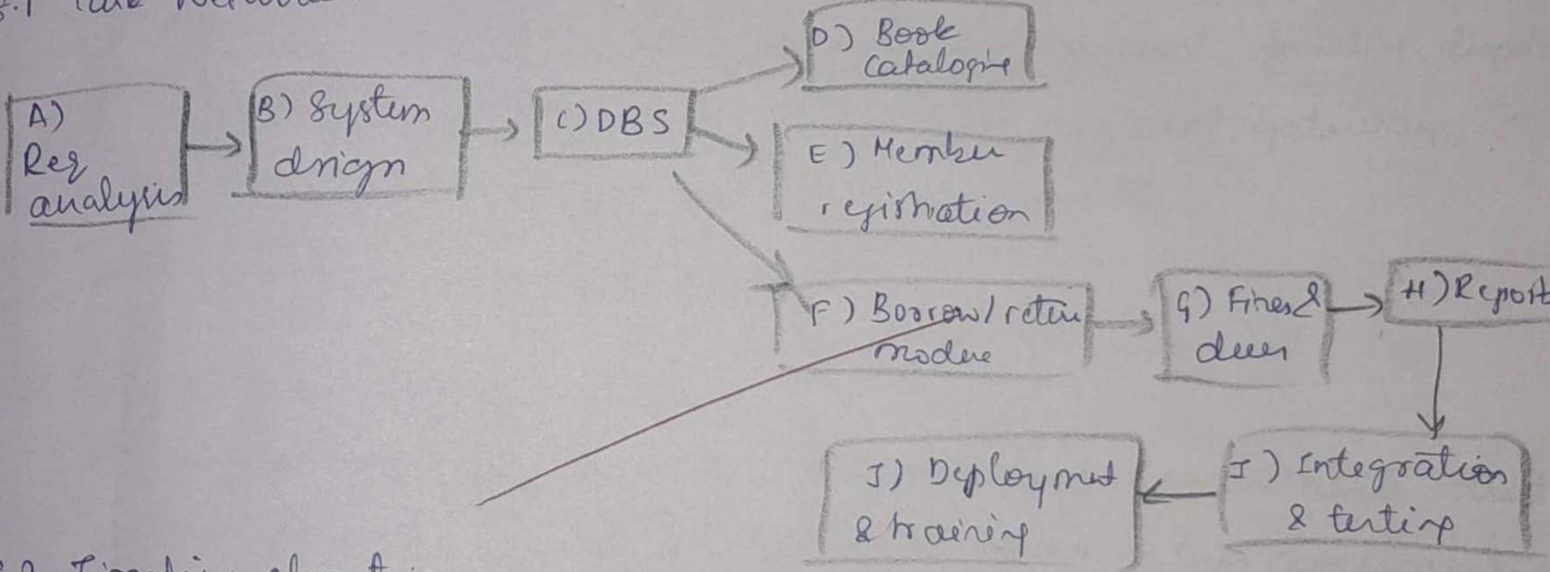
- system must handle atleast 10000 items efficiently.
- respons time ≤ 2s. Support atleast 100 concurrent users.

6. Design constraints
- Must support Windows/Linux OS
- Database must be SQL-based for portability
- Must include data backup and recovery features
- Model-view-controller design pattern

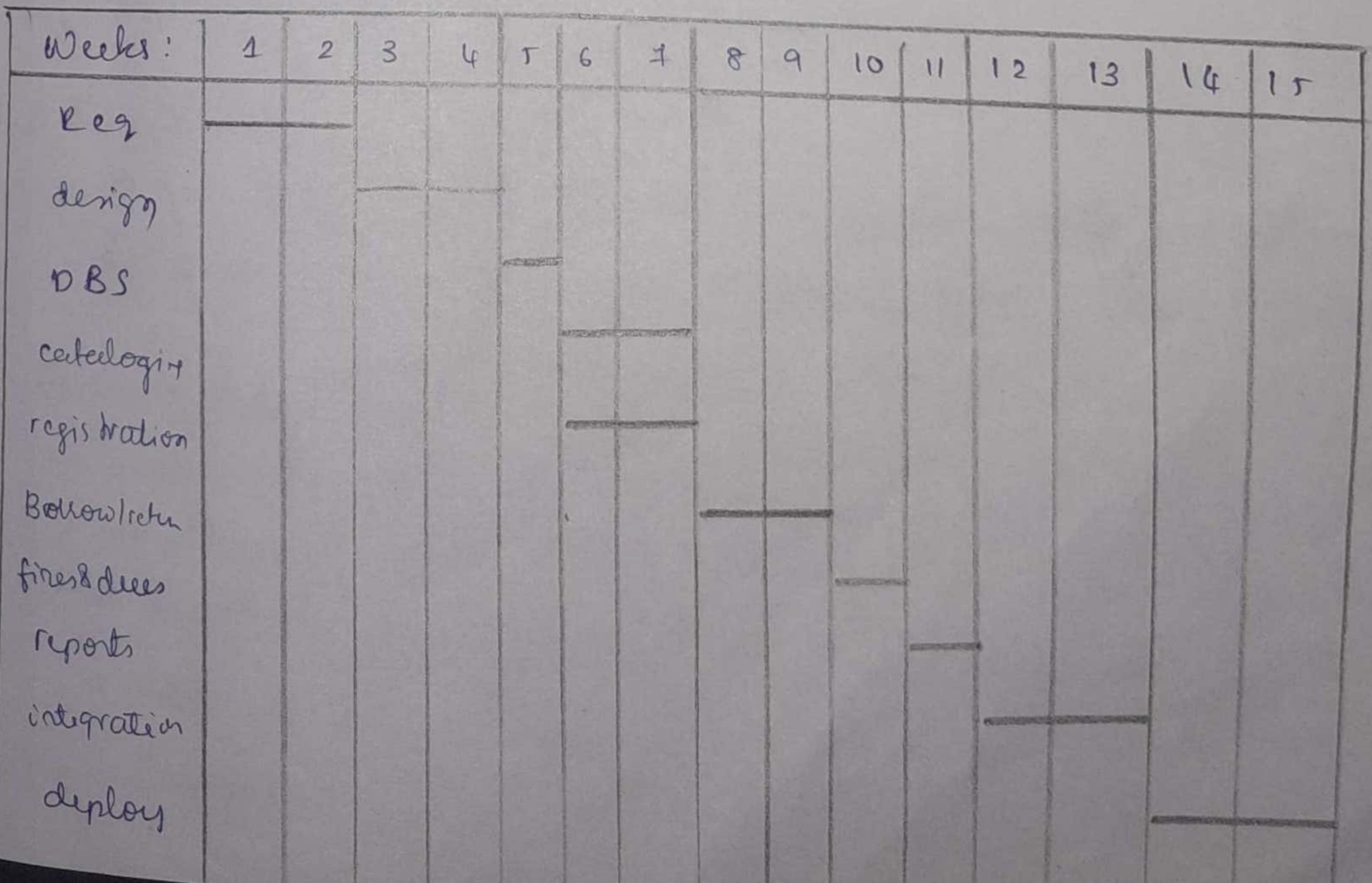
7. Non-Functional attributes
- Security : Role based access control, password encryption
- Usability : Easy to use UI
- Reliability : 99% uptime, data backup.
- Scalability : support multi branch expansion
- Maintainability : Modular design

8. Preliminary schedule and Budget

8.1 Task Network.



8.2 Timeline chart



8.3 Effort estimation and cost calculation.

Total LOC = 12,500

Productivity = 1000

Effort = LOC / productivity = 12.5 ≈ 13

Cost = Effort * cost per person-monthly

= 13 * 80,000

= ₹ 10,40,000

5) Passport auto mation system

problem statement : Develop a system that'll digitalize endtoend operations - user registration, application filling, document upload, appointments, biometrics, fee payment, police verification, decisioning, printing & dispatch, tracking and grievances - with strong security, auditability.

1. Introduction

1.1 ~~Purpose~~ Scope of this document :
This system will provide.
- Citizen portal for registration, application, document upload, fee payment, slot booking, application tracking, feedback.
- Office portal for document scrutiny, biometric capture, police verification workflow, risk/fraud checks, approval, printing, dispatch
- Integrations with identity verification, payment gateway, SMS/email, police verification, courier dispatch.
- Role based access to the system.

1.2. Purpose of this document.
To define funtional and non-functional requirements that automates all the tasks mentioned above. This document guids all the stakeholders.

1.3 Overview
A web-first mobile responsive system with centralized databases, secure api's to partner agencies, real-time monitoring

2. General description
→ Users : • Applicant : create/submit application, pay fee, book slot, track
- Front desk operator : scan/verify originals, capture biometrics
- Verifier/Scrutiny officer : document checks, raise queries.
- Police officer : recieves verification tasks, submits reports.
- Dispatch : printing, lamination, packaging, handoff to courier.

→ Operating environment : · Web application, window/linux server
· containerized deployment.

→ Assumptions and dependencies
- reliable internet, external services for police verification, payment, SMS/email, courier.

3. Functional requirements

3.1 Registration & Authentication
- applicants register with mobile/email OTP.
- password reset, ~~do~~ session management, CAPTCHA

3.2 Application management.
- create/edit application
- dynamic forms with validation.
- auto save drafts

3.3 document upload & verification.
- upload required documents
- client side checks, server side antivirus.
- operator scan & marks originals verified.

3.6 Appointment scheduling.
- Real time calendar by center & counter
- reschedule / cancel within policy.

3.5 Biometrics and photo capture
- capture face photo ~~on~~, fingerprints, signature.

3.6 Payments and fees.
- Fee calculation by, service type, booklet pages
- Payments via card UPI, cancellation per policy
- E-reciept & reconcilation with reports.

3.7 Verification, desisioning, issuance
- officers run data, verify, police verification, risk/fraud checks.
- application will be approved, held, rejected.
- on approval, print order, sealing, dispatch through courier service

3.8 Tracking, Notifications & Administration.

- End to end status tracking. SMS/email updates. feedback.
- User (roles, permissions, fee tables, holidays, checklists.

4. Interface requirements

→ UI: Responsive, multilingual.

→ System interface: Identity / ID verification API, police verification, payment gateway, SMS /Email providers, courier tracking.

5. Performance requirements

- support ≥ 5000 concurrent applicants
- Response time ≤ 2s
- appointment allocation < 300ms.
- 99.99% uptime and backup.

6. Design Constraints

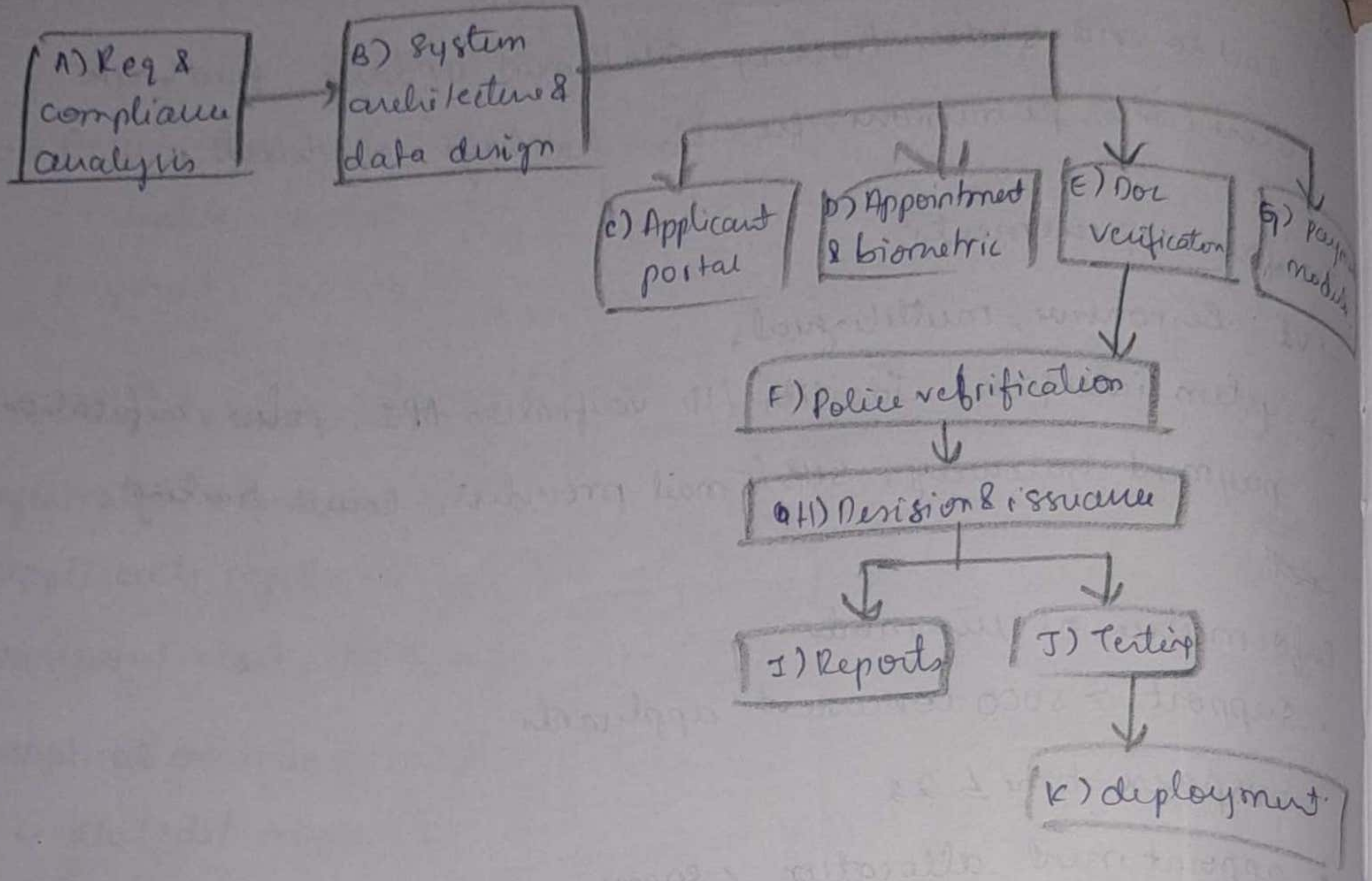
- compliance with applicable data protection laws and govt security guidelines; audit trails
- Secure coding, encryption at rest and in transit
- Use of containerised microservices
- Multilingual content

7. Non functional attributes

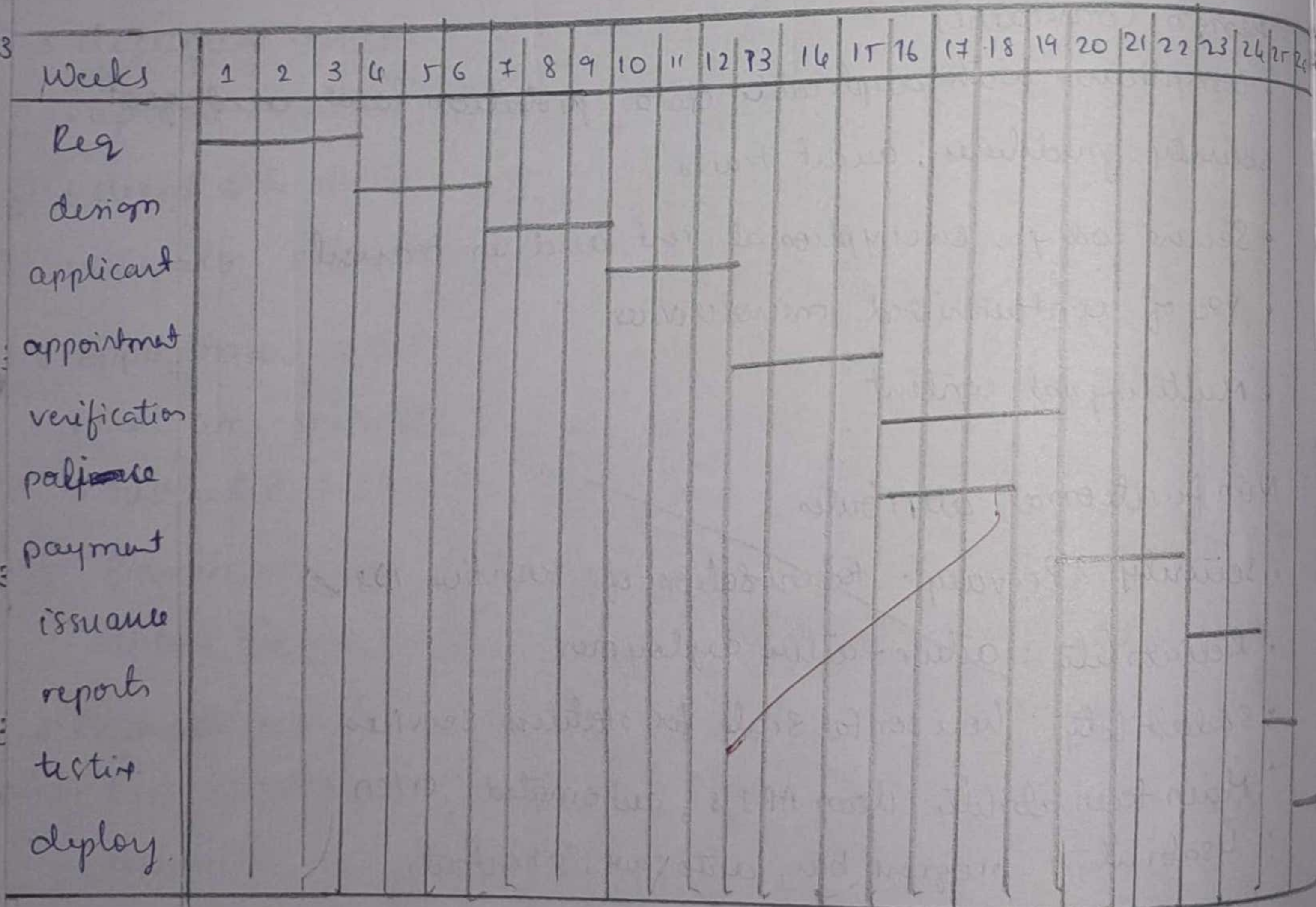
- Security & Privacy: tokenisation of sensitive IDs, e
- Reliability: active-active deployment.
- Scalability: horizontal scale for stateless services.
- Maintainability: clean API's, automated CI/CD
- Usability: progress bar, autosave, shortcuts.
- Data integrity: transactional updates, referential constraints.

8. Preliminary schedule and budget.

8.1 Task network.



8.2 Time line Chart



8.3 : Effort estimation and cost calculation

Total LOC = 18,750

Productivity = 1000

Effort = LOC / productivity = 18.75 ≈ 19

Cost = effort * cost per person-month

= 19 * 1,00,000

= ₹ 19,00,000

[illegible]
25-8-2025