Module 3

Requirement Analysis and Software Estimation and Scheduling

1. What are Functional Requirements?

- These are the **features or functions** that a system **must perform**.
  They answer the question:
  “What should the system do?”

 Example:

- In a library system: *Borrow and return books*
- In a banking system: *Transfer money*
- In a shopping site: *Add products to cart*

 Characteristics:

- Specific tasks, operations, or behaviors
- Directly visible to users
- Easy to test (does it work or not?)

2. What are Non-Functional Requirements?

- These are the **qualities or constraints** of the system.
  They answer the question:
  “How should the system work?”

 Example:

- In a library system: *System should be available 24/7*
- In a banking system: *Transactions must be secure with OTP*
- In a shopping site: *Website must load within 3 seconds*

 Characteristics:

- Not about *what* the system does, but *how well* it does it
- Related to performance, security, usability, scalability, etc.
- Harder to measure but very important

Example : Car

- **Functional Requirements (What it does):**
  - Car must start with a key.
  - Car must accelerate when pressing the pedal.
  - Car must stop when brakes are applied.
- **Non-Functional Requirements (How it does it):**
  - Car must start within 2 seconds.
  - Car must give mileage of 15 km/l.
  - Car must be safe with airbags and seat belts.

 Without functional requirements, the system is **useless**.

 Without non-functional requirements, the system is **unreliable, slow, or unsafe**.

Functional Vs Non-Functional Requirements

Aspect	**Functional Requirement**	**Non-Functional Requirement**
Meaning	What system should do	How system should work
Focus	Actions, tasks, features	Quality, performance, constraints
Example (Library)	Issue & return books	Must be available 24/7
Example (Banking)	Transfer money	Transaction must be secure
Analogy (Car)	Start, run, stop	Speed, safety, comfort

Library Management System (LMS)

- **Functional Requirements (FR):**
- **User Management** – Admin can add, update, and delete librarian, students, and faculty accounts.
- **Book Management** – Add, update, remove, and search books by title, author, ISBN.
- **Borrowing & Returning** – Students/faculty can borrow and return books, system tracks due dates.
- **Fine Management** – Automatic calculation of late fees.
- **Reservation System** – Users can reserve a book if currently issued.
- **Report Generation** – Daily/weekly/monthly reports for issued, returned, and overdue books.
- **Notification System** – SMS/Email reminders for due dates and reservation availability.

Non-Functional Requirements (NFR):

- **Performance** – System should handle at least 1000 concurrent users.
- **Reliability** – Ensure no book is double-issued.
- **Security** – Role-based access (admin, librarian, student).
- **Usability** – User-friendly interface for quick book search.
- **Scalability** – Able to add large book databases (millions of records).
- **Availability** – 99.9% uptime for online library portals.

Student Management System (SMS)

- **Functional Requirements (FR):**
- **Student Enrollment** – Add/update student details, assign roll number.
- **Course Management** – Assign subjects, courses, and faculty.
- **Attendance Management** – Record daily attendance.
- **Grade & Exam Management** – Store exam results, generate report cards.
- **Fee Management** – Track student fee payments, generate receipts.
- **Communication** – Notifications to students/parents via email/SMS.
- **Library/Hostel/Transport Integration** – Manage student services.

Non-Functional Requirements (NFR):

- **Performance** – Quick response for searching student details.
- **Scalability** – Should handle thousands of students across different branches.
- **Security** – Data protection (marks, personal details encrypted).
- **Maintainability** – Easy update of courses, subjects, and semesters.
- **Reliability** – System should not lose data during failures.
- **Accessibility** – Mobile and web-based access for parents and students.

Banking System

- **Functional Requirements (FR):**
- **Account Management** – Create, update, close accounts.
- **Transaction Management** – Deposit, withdraw, transfer funds.
- **Balance Inquiry** – View current balance and mini statements.
- **Loan Management** – Apply, approve, and track loan repayment.
- **ATM & Online Banking Integration** – Provide secure access to customers.
- **Customer Support** – Handle queries, complaints, and service requests.
- **Audit & Compliance** – Generate regulatory reports for RBI/Government.

Non-Functional Requirements (NFR):

- **Security** – High-level encryption (SSL, OTP, biometric authentication).
- **Reliability** – Ensure accurate transactions, no double withdrawals.
- **Performance** – Must process thousands of transactions per second.
- **Availability** – 24/7 system availability (critical for banking).
- **Compliance** – Must follow banking regulations and standards (e.g., PCI DSS).
- **Disaster Recovery** – Data backup and failover mechanisms.

Employment Management System

- **Functional Requirements (FR):**
- **Employee Records Management** – Add, update, delete employee details.
- **Payroll Management** – Calculate salary, tax deductions, generate payslips.
- **Leave Management** – Apply, approve/reject leave requests.
- **Recruitment Management** – Job postings, applications, interview scheduling.
- **Performance Management** – Track employee goals, appraisals, and promotions.
- **Training Management** – Assign courses, monitor progress.
- **Reports** – Attendance reports, salary reports, HR dashboards.

Non-Functional Requirements (NFR):

- **Security** – Role-based access (HR, Admin, Employee).
- **Performance** – Should quickly fetch employee records.
- **Usability** – Intuitive UI for HR and employees.
- **Scalability** – Should support thousands of employees across multiple branches.
- **Reliability** – Salary calculation must be 100% accurate.
- **Maintainability** – Easy updates for HR policies, salary structures.

Online Shopping System (E-Commerce)

- **Functional Requirements (FR):**
- **User Management** – Register, login, profile management.
- **Product Catalog** – Browse products by category, search filters.
- **Shopping Cart** – Add/remove products, quantity updates.
- **Order Management** – Place, track, cancel orders.
- **Payment Gateway Integration** – Secure online payments via UPI, cards, wallets.
- **Delivery Management** – Address entry, shipping tracking.
- **Review & Rating System** – Customers can rate/review products.
- **Admin Panel** – Manage products, offers, discounts, and inventory.

Non-Functional Requirements (NFR):

- **Performance** – Must handle peak loads during sales (Black Friday, Diwali).
- **Security** – Secure transactions (SSL, PCI DSS compliance).
- **Availability** – 24/7 availability, minimal downtime.
- **Scalability** – Able to support millions of products and users.
- **Usability** – Easy navigation, responsive design for mobile and desktop.
- **Reliability** – Orders should not be lost, payments must always confirm correctly.
- **Personalization** – Product recommendations using AI.

Software Documentation

1. What is Software Documentation?

- Software documentation is the **written text, diagrams, or guides** that explain how a software system is **designed, developed, and used**.
- It acts like a **manual or reference** for developers, testers, users, and customers.
- Without documentation, software becomes very hard to **understand, maintain, or use**.

2. Types of Software Documentation

A. User Documentation (for end users)

- Explains how to **use the software**.
- Written in **simple, non-technical language**.
- Examples:
  - User manuals
  - Online help guides
  - FAQs
  - Tutorials

 Example: In Microsoft Word, the “Help” section is **user documentation**.

B. Technical Documentation (for developers & engineers)

 Explains how the software is **built, designed, and maintained**.

- Written in **technical language**.
- Examples:
  - Software requirements specification (SRS)
  - Design documents (UML diagrams, architecture)
  - API documentation
  - Test plans and reports
  - Maintenance manuals

 Example: In Java, the **Javadoc** is technical documentation for programmers.

3. Importance of Software Documentation

- **Understanding** – New developers or users can easily learn the system.
- **Maintenance** – Helps in fixing bugs and updating software.
- **Knowledge Sharing** – Acts as a reference for teams.
- **Training** – Useful for teaching new users or employees.
- **Quality** – Good documentation improves reliability and usability of software.

4. Example (Library Management System Documentation)

- **User Documentation:**
  - Steps to log in, search a book, issue/return a book.
- **Technical Documentation:**
  - Database design for books and students.
  - Class diagrams of Book, Student, Librarian.
  - API endpoints for online library access.

5. Simple Analogy

- Think of software documentation like a **recipe book** 🍲:
- The **recipe (steps)** = Functional requirements / User guide
- The **ingredients & method details** = Technical documentation
- Without a recipe, cooking becomes confusing. Same with software.

Software Requirements Specification (SRS)

1. What is SRS?

- SRS is a **formal document** that describes **what the software should do** and **how it should behave**.
- It is like a **blueprint** of the software project.
- Prepared **before development starts**, so developers, testers, and clients have the same understanding.

 In short: -**SRS = Agreement between client and development team.**

2. Purpose of SRS

- To clearly **define requirements** (functional & non-functional).
- To **avoid misunderstandings** between client and developers.
- To act as a **reference** during development & testing.
- To help in **maintenance and future updates**.

3. Structure / Format of SRS

A typical SRS has the following sections:

1. Introduction

- Purpose of the system
- Scope of the system
- Definitions, acronyms, abbreviations
- References

2. Overall Description

- Product perspective (where this software fits)
- Product functions (summary of features)
- User characteristics (who will use it)
- Constraints (limitations like hardware/software)
- Assumptions and dependencies

3. Functional Requirements

- Detailed description of **what the system should do**
- Example: *“The system shall allow the user to borrow a book by entering book ID and student ID.”*

3. Structure / Format of SRS

4. Non-Functional Requirements

- Performance requirements
- Security requirements
- Usability requirements
- Reliability and availability

5. External Interface Requirements

- User interface (screens, forms, menus)
- Hardware interface (devices, printers, scanners)
- Software interface (with other applications)
- Communication interface (network protocols, APIs)

6. Other Sections (Optional)

- ER diagrams, UML diagrams
- Data dictionary
- Test cases

4. Example (Mini SRS for Library Management System)

1. Introduction:

- Purpose: To develop a system for managing library books, members, and transactions.
- Scope: Students can search, borrow, and return books. Librarians can add/remove books.

2. Overall Description:

- Users: Students, Librarians, Admin
- Constraints: Runs only on Windows/Linux with MySQL database.

3. Functional Requirements:

- System shall allow students to search for books by title or author.
- System shall allow librarians to issue and return books.
- System shall generate a fine if books are returned late.

4. Non-Functional Requirements:

- System should support at least 500 users at a time.
- System should be available 24/7.
- System should ensure secure login with password protection.

5. Interfaces:

- Web interface for users.
- Database connectivity with MySQL.

Summary

- SRS is a detailed document that describes the purpose, features, and behavior of software before development starts.