Requirements

ROJO, J/ALIWATE, D/EDO, A EDO/UY, ULRICH/ODARVE, C

- 1. Requirements Elicitation
- 2. Requirements Documentation
- 3. Requirements Maintenance & Management
- 4. Modeling Tools and Methodologies Using UML
- 5. Testing of Requirements

- **Requirements Elicitation** is the process of identifying, gathering, and understanding the needs and expectations of stakeholders for a software project. It is a critical phase in the Software Development Life Cycle (SDLC) as it lays the foundation for defining the system's functionality, constraints, and objectives.

A. Requirements Elicitation

- **S**takeholder Identification

  -Determining key users, clients, and decision-makers.

- **T**echniques for Gathering Requirements

  1. Interviews & Surveys - Direct interaction with stakeholders through interviews allows for a deep dive into their needs. Surveys, on the other hand, can gather quantitative data from a larger audience, providing a broader perspective.

  ex. business analyst might use structured interviews to gather detailed user requirements

2. User Stories and Use Cases- These narrative tools help in visualizing the end-user's interaction with the system, making it easier to understand and communicate requirements.

- ***ex.*** user stories that describe how a customer will use an online shopping platform can help developers understand the features needed.

3. Prototyping – Creating mock-ups for validation.

ex. Clickable prototype of a mobile app interface can help stakeholders visualize the end product and suggest changes early in the development process.

3. Requirement Workshops: These collaborative sessions bring together various stakeholders to discuss and reconcile different needs and viewpoints

ex. Conducting a workshop with end-users, developers, and quality assurance teams to define the acceptance criteria for a new feature.

5. Observation & Job Shadowing - Analyzing existing workflows.

ex. Shadowing a nurse during their shift to understand the requirements for a healthcare management system

6. Document Analysis-Reviewing existing documentation can provide insights into current processes and systems, helping to identify areas for improvement.

e.g. Analyzing user manuals to identify gaps in current software functionalities

7. Brainstorming Sessions - Collaborative sessions to explore ideas.

e.g. A brainstorming session that leads to the idea of integrating AI

Effective Use of Requirements Gathering Tools

B. Requirement Classification

Functional vs. **Non-functional** Requirements

1. Functional Requirements (FR) These define the specific features, functionalities, and behaviors that the system must perform.

Examples:

- User authentication and authorization
- Data input and validation
- CRUD operations (Create, Read, Update, Delete)
- Payment processing in an e-commerce system
- Generating reports and dashboards

Key Documents:

Use Cases, User Stories, Process Flows

2. Non-Functional Requirements (NFR)

These define the quality attributes, constraints, and system performance that do not directly relate to specific functionalities.

Types of NFRs:

Performance Requirements – System response time, load capacity

Scalability – Ability to handle growing user loads

Security Requirements – Data encryption, authentication standards

Usability Requirements – User experience, accessibility

Reliability & Availability – System uptime, error handling

Maintainability & Portability – Ease of updates, system migration

Example:

The system should respond to user requests within 2 seconds under normal load conditions.

Functional Requirements

- **Describes** what the system should do, i.e., specific functionality or tasks.
- **Focuses** on the behavior and features of the system.
- **Defines** the actions and operations of the system.
- **User authentication** data input/output, transaction processing

VS

Non Functional Requirements

- **Describes** how the system should perform, i.e., system attributes or quality.
- **Focuses** on the performance, usability, and other quality attributes.
- **Defines** constraints or conditions under which the system must operate
- **Scalability** security, response time, reliability, maintainability.

Aspect	Functional Requirements	Non-Functional Requirements
Definition	Describes what the system should do, i.e., specific functionality or tasks.	Describes how the system should perform, i.e., system attributes or quality.
Purpose	Focuses on the behavior and features of the system.	Focuses on the performance, usability, and other quality attributes.
Scope	Defines the actions and operations of the system.	Defines constraints or conditions under which the system must operate.
Examples	User authentication, data input/output, transaction processing.	Scalability, security, response time, reliability, maintainability.

Aspect	Functional Requirements	Non-Functional Requirements
Measurement	Easy to measure in terms of outputs or results.	More difficult to measure, often assessed using benchmarks or SLAs.
Impact on Development	Drives the core design and functionality of the system.	Affects the architecture and overall performance of the system.
Focus on User Needs	Directly related to user and business requirements.	Focuses on user experience and system performance.

Aspect	Functional Requirements	Non-Functional Requirements
Documentation	Typically documented in use cases, functional specifications, etc.	Documented through performance criteria, technical specifications, etc.
Evaluation	Can be tested through functional testing (e.g., unit or integration tests).	Evaluated through performance testing, security testing, and usability testing.
Dependency	Determines what the system must do to meet user needs.	Depends on how well the system performs the required tasks.

How to Gather Functional and Non-functional Requirements

Gathering requirements involves multiple approaches and collaboration between the development team, stakeholders, and end-users:

1. Functional Requirements:

Interviews: Talk to stakeholders or users to understand their needs.

Surveys: Distribute questionnaires to gather input from a larger audience.

Workshops: Host sessions to brainstorm features and gather feedback.

2. Non-functional Requirements:

Performance Benchmarks: Consult with IT teams to set expectations for performance and load.

Security Standards: Consult with security experts to define the best practices for data protection.

Usability Testing: Test the system to find areas where users might struggle and refine the interface.

2. **Requirements Documentation**

- Software Requirements Specification (SRS)

Software requirements specification lists out all the requirements stated by the user inconsistent manner.

- User Stories and Use Case Diagrams

- Business Requirement Document (BRD) vs. Functional Requirement Document (FRD)
- IEEE Standard for SRS
- Traceability Matrix

3. Requirements Maintenance & Management

- Change Management in Requirements
- Version Control for Requirements
- Impact Analysis of Requirement Changes
- Handling Requirement Conflicts

4. **Modeling Tools and Methodologies Using UML**

- UML Diagrams (Use Case, Class, Sequence, Activity, State Machine)
- Data Flow Diagrams (DFD)
- Entity-Relationship Diagrams (ERD)
- BPMN (Business Process Model and Notation)
- Agile Requirement Modeling (User Stories, Epics)

5. **Testing of Requirements**

- Requirements Validation and Verification
- Test-Driven Development (TDD) and Behavior-Driven Development (BDD)
- Requirement-based Test Case Design
- Ensuring Test Coverage of Requirements
- Acceptance Criteria and User Acceptance Testing (UAT)

Sample User Stories

Scenario: Online Library Management System

User Story 1: User Registration

1. As a new user,
2. I want to register with my email and password,
3. So that I can access my account and borrow books.

Acceptance Criteria:

✓ The system should allow users to register with a valid email and password.

✓ Passwords should be encrypted for security.

✓ Users should receive a confirmation email after successful registration

Sample User Stories

Scenario: Online Library Management System

User Story 2: Borrowing a Book

1. As a registered user,
2. I want to search for a book and borrow it,
3. So that I can read it within the allowed time.

Acceptance Criteria:

✓ Users should be able to search for books by title, author, or category.
✓ The system should check book availability before allowing a borrow request.
✓ The user should receive a due date for returning the book.

Sample User Stories

Scenario: Online Library Management System

User Story 3: Managing Books (Librarian Role)

1. As a librarian,
2. I want to add, update, or remove books from the system,
3. So that I can keep the library's book catalog up to date.

Acceptance Criteria:

✓ The system should allow librarians to add book details (title, author, ISBN, etc.).

✓ Books should have a status (available, borrowed, reserved).

✓ The system should restrict book management actions to librarian users only.

Sample Use Case Diagram

The Use Case Diagram visually represents interactions between users and the system.

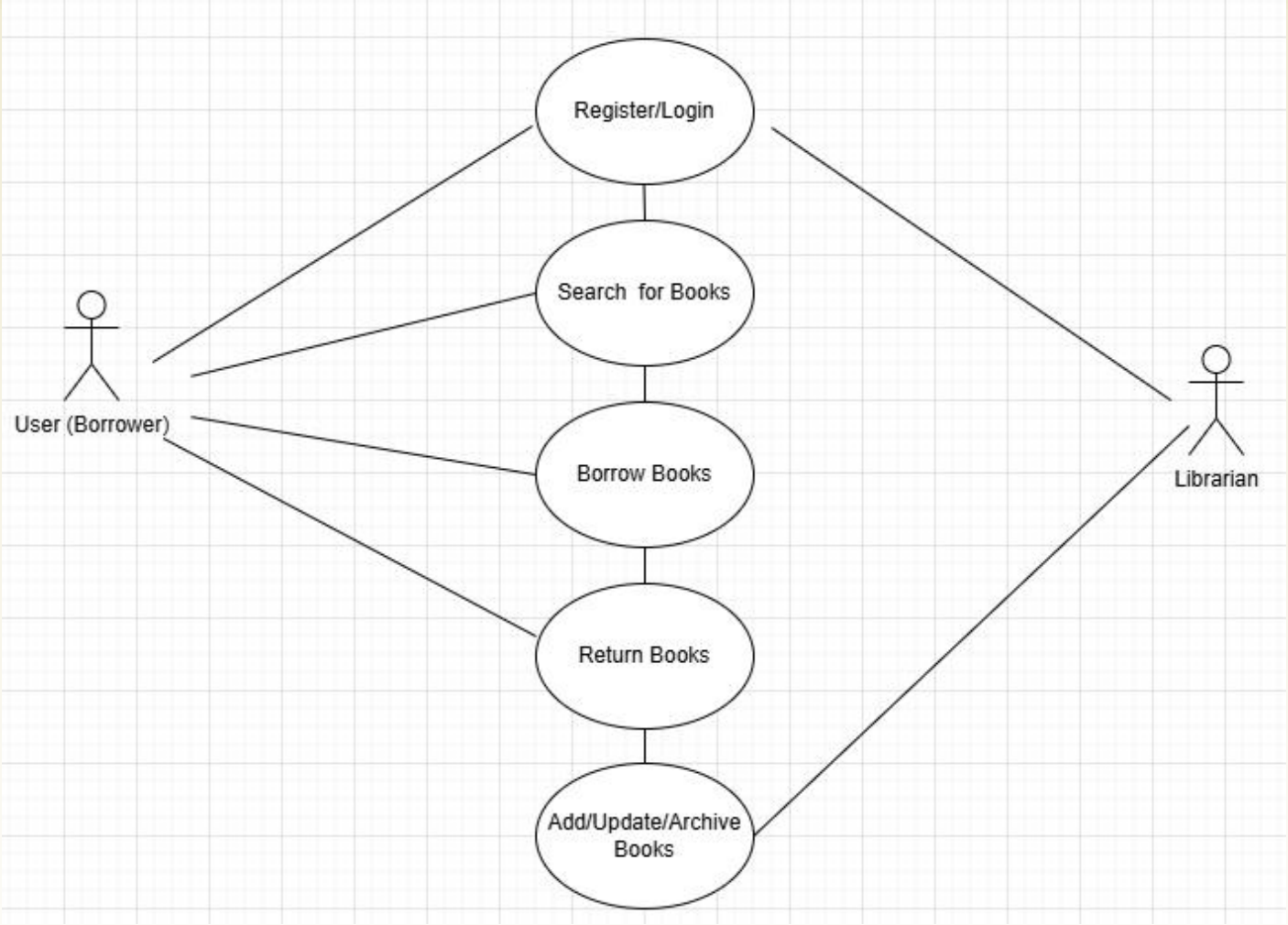
Actors:

- User (Borrower)
- Librarian (Admin)

Use Cases:

- Register/Login
- Search for Books
- Borrow Books
- Return Books
- Add/Update/Delete Books (Librarian only)

Sample Use Case Diagram



Note: *This is a sample use case diagram. User management module and other system components were not included. (created using Draw.io)*

Processes (Functionalities) – These are actions or operations performed in the system (e.g., user registration, book search).
Modules (Components of the System) – These are structural divisions of the system that group related functionalities (e.g., User Management Module, Book Management Module).
Use Cases (User Interactions) – These describe how users interact with specific functionalities (e.g., "Register/Login" represents the process of authentication).

User/Borrower transaction module

Use Case	Process/Functionality	Module (System Component)
Register/Login	User authentication	User Management
Search for Books	Searching book records	Book Catalog Management
Borrow Book	Book borrowing process	Lending & Transactions
Return Book	Book return process	Lending & Transactions

Book Management Module

Use Case	Process/Functionality	Module (System Component)
Add Books	Adding new books to the system	Book Management Module
Update Books	Modifying book details (title, author, availability, etc.)	Book Management Module
Delete Books	Removing books from the system	Book Management Module

Sample tools for visual use case diagram.

1. **Lucidchart:** A web-based diagramming tool that's easy to use and offers a wide range of templates and shapes.
2. **Draw.io:** A free online diagramming tool that integrates well with Google Drive and other cloud storage services.
3. **Microsoft Visio:** A powerful diagramming tool that's part of the Microsoft Office suite, ideal for creating detailed and professional diagrams.
4. **Creately:** Another web-based diagramming tool that offers collaboration features and a variety of templates.

Sample tools for visual use case diagram.

5. **Gliffy:** A cloud-based diagramming tool that offers an intuitive interface and integration with Atlassian products like Confluence and Jira.

6. **Cacoo:** An online tool that allows for real-time collaboration and has a wide variety of templates and shapes.

7. **SmartDraw:** A diagramming tool that offers intelligent formatting and a large collection of templates.

Suggested Activities

Activity 11) Title: Creating User Stories for Your Proposed System

Objective: Develop structured user stories to define system functionalities and user interactions.

Instructions:

1. Identify key users (e.g., students, customer, manager etc.).
2. Write user stories using the format:
   - As a [user], I want to [goal], so that [reason].
3. Define acceptance criteria for each story.
4. Validate and refine user stories based on feedback.

Suggested Activities

System Integration and Architecture (Activities)

Activity 12) *Title:* Creating a Survey Questionnaire for the Proposed Information System

Objective: To be able to apply the Importance of Closed-Ended/Open-Ende Questions and Key Criteria to Enhance Respondent Participation

Activity 13) *Title:* Designing an Entity-Relationship Diagram for a Database of your Proposed System

Objective: To understand and apply the principles of Entity-Relationship Diagrams (ERD) by designing a structured database model for a proposed system.

Reference books

- Sage A.P. and Rouse, W.B. Handbook of Systems Engineering and management, John Wiley & Sons, 1999.
- Kendall And Kendall, System Analysis and Design