Software Requirement Specification

Project Title: Simple Finance Management System
Technology: Java, Spring Boot, MySQL
Duration: 4 Days

1. Introduction

1.1 Purpose

The system helps individuals or small businesses manage finances by tracking accounts, expenses, liabilities, and loans in one place. It provides insights into financial health through categorized records and summary reports.

1.2 Scope

The application will:

- Support multiple financial **accounts** (cash, bank, wallet, etc.).
- Track **spends & expenses** by category.
- Manage **liabilities & loans**.
- Generate basic **reports** on cash flow, expenses, and liabilities.

2. Functional Requirements

2.1 Account Management

- Create, edit, delete accounts.
- Fields: account name, type (cash/bank/wallet), opening balance, current balance.

2.2 Expense & Spend Tracking

- Add spends/expenses linked to an account.
- Categorize expenses (food, rent, travel, etc.).
- Record amount, date, and notes.

2.3 Liabilities

- Add liabilities (e.g., credit card dues, utility bills).
- Track due date and amount.
- Mark as **paid/unpaid**.

2.4 Loans

- Add loans (borrowed or given).
- Record principal, interest rate, EMI, duration.
- Track repayment schedule and outstanding balance.

2.5 Reports

- **Expense Report:** Total expenses by category/date range.
- **Cash Flow Report:** Inflows vs outflows per account.
- **Liability Report:** Pending liabilities.
- **Loan Report:** Outstanding loan balances.

3. Non-Functional Requirements

- **Usability:** Simple dashboard for quick access.
- **Security:** Basic authentication (username/password).
- **Performance:** Should handle 1,000+ transactions efficiently.
- **Reliability:** Ensure balance never mismatches actual transactions.
- **Scalability:** Support adding more modules in the future (e.g., investments).

4. System Design (High-Level)

4.1 User Roles

- **User (single role)** → manages all accounts and transactions.

4.2 Modules

1. **Account Module** → CRUD operations on accounts.
2. **Expense Module** → Record and categorize spends.
3. **Liability Module** → Track dues and payments.

4. **Loan Module** → Manage borrowings and repayments.
5. **Reports Module** → Summaries and analytics.

4.3 Workflow

1. User adds accounts (e.g., Cash, SBI Bank).
2. User records expenses → account balance decreases.
3. User records liabilities & due dates.
4. User records loans with repayment details.
5. User views reports to analyze financial health.

5. Constraints

- Basic UI (can be Angular/React or just Thymeleaf for simplicity).
- No multi-user support (single user only for MVP).
- Charts/analytics optional (can be simple tables for 3–4 days).

6. Deliverables

1. **Backend (Spring Boot):** REST APIs for accounts, expenses, liabilities, and loans.
2. **Frontend:**
   - Dashboard (overview of balances).
   - Expense input form + listing.
   - Liability/Loan management.
   - Reports page.
3. **Database:** Tables/collections for accounts, expenses, liabilities, and loans.

7. Evaluation Criteria

1. **Functional Accuracy** – Accounts update balances correctly, expenses/liabilities/loans work as expected.
2. **Usability** – User can add records with 2–3 clicks.
3. **Performance** – Reports generated under 2s for 1,000 records.
4. **Data Integrity** – Balance never mismatches transaction history.
5. **Completeness of Reports** – Expense, Cash Flow, Liability, Loan reports accurate.
6. **Timeline Compliance** – Completed within 3–4 days.
7. **Maintainability** – APIs and DB schema modular for future enhancements.

8. **Overall Acceptance** – User can manage complete finance cycle (account → expense → liability → loan → report) without errors.