PES UNIVERSITY

UE22CS341A: Software Engineering
Case Study

Unit 1 Deliverable

Samay SR
Sandesh Kumatgi

Software Requirements Specification (SRS) for Banking System Web Application

1. Introduction

1.1 Purpose

This document specifies the requirements for the Banking System Web Application, developed using React for the frontend and Node.js with MySQL for the backend. The system facilitates online banking services such as account management, transactions, and customer data handling.

1.2 Scope

The Banking System Web Application is designed for use by bank customers and employees, providing a secure, user-friendly interface for managing accounts, processing transactions, and accessing banking services online. The system will interact with the MySQL database to store and retrieve customer and transaction data.

.

1.3 Definitions, Acronyms, and Abbreviations

- DBMS: Database Management System
- API: Application Programming Interface
- UI: User Interface
- React: A JavaScript library for building user interfaces
- Node.js: A JavaScript runtime built on Chrome's V8 engine
- MySQL: A relational database management system

1.4 References

- IEEE Standard for Software Requirements Specifications (IEEE Std 830 1998)

1.5 Overview

This document is structured into sections detailing the overall description, external interface requirements, system features, and non-functional requirements of the Banking System Web Application.

2. Overall Description

2.1 Product Perspective

The Banking System Web Application is an independent application that interacts with the bank's central database through a RESTful API built using Node.js. The frontend is developed using React to ensure a dynamic and responsive user experience.

2.2 Product Functions

- User Authentication: Login and registration functionality for users.
- Account Management: View and manage bank accounts.
- Transaction Processing: Facilitate deposits, withdrawals, transfers, and payments.
- Customer Data Management: Securely store and retrieve customer information.
- Report Generation: Generate account statements and transaction histories.
- Security Features: Implement user authentication and data encryption.

2.3 User Classes and Characteristics

- Customers: Individuals with bank accounts who use the web application for online banking.
- Bank Employees: Staff responsible for managing customer accounts and transactions.
- Administrators: Personnel responsible for system maintenance, security, and data integrity.

2.4 Operating Environment

- Frontend: Runs on modern web browsers, developed using React.
- Backend: Hosted on a server running Node.js, connected to a MySQL database.
- Database: MySQL server for data storage and management.

2.5 Design and Implementation Constraints

- Compliance: Must comply with financial regulations and data protection laws.
- Security: Must ensure secure data transmission and storage using HTTPS and encryption.
- Performance: Must handle concurrent transactions and provide a responsive user experience.

2.6 Assumptions and Dependencies

- The application assumes a stable internet connection for user access.
- Regular updates and maintenance of the system are assumed for optimal performance.
- The system depends on third-party services for payment processing and data backups.

3. External Interface Requirements

3.1 User Interfaces

- Web Interface: Accessible via modern web browsers, designed with React for a responsive UI.
- Admin Interface: For bank employees and administrators to manage accounts and the system.

3.2 Hardware Interfaces

- Servers: Hosting Node.js backend and MySQL database.
- Client Devices: Computers, tablets, and smartphones used by customers to access the web application.

3.3 Software Interfaces

- RESTful API: Node.js backend provides API endpoints for frontend interaction.
- MySQL Database: Manages customer data, transactions, and account details.
- Third-Party Services: Payment gateway API integration for online payments.

3.4 Communication Interfaces

- HTTPS: Secure communication protocol for data transmission between the frontend and backend.
- WebSocket (optional): For real-time updates and notifications.

4. System Features

4.1 User Authentication

Users authenticate using their email and password.

4.2 Account Management

Users can view and manage their bank accounts online.

4.3 Transaction Processing

Enables secure financial transactions like deposits, withdrawals, and transfers.

4.4 Security Features

Ensures the security of the application and user data through robust authentication and encryption mechanisms.

4.5 Error Handling

Manages application errors and provides feedback to users

5. Non-Functional Requirements

5.1 Performance Requirements

- The system will load the main dashboard.

5.2 Security Requirements

- The system shall use HTTPS for all communications between the frontend and backend.
- The system shall ensure that all sensitive data, such as passwords and transaction details, are encrypted in the database.

5.3 Usability Requirements

- The web application shall be responsive, providing a consistent user experience across different devices.
- The system shall provide an intuitive and user-friendly interface with clear navigation.

5.4 Reliability Requirements

- The system shall maintain an uptime of 99.9%.
- The system shall have failover mechanisms in place to handle server outages(optional).

6. Other Requirements

6.1 Regulatory Requirements

The system shall comply with GDPR (General Data Protection Regulation) and other relevant data protection laws

6.2 Environmental Requirements

The system shall be hosted in a data center with controlled environmental conditions, including temperature and humidity.(Optional)