Software Requirements Specification

for

SplitPay

Version 1.0 approved

Prepared by Tamyla Nguyen & Raine Johnson

The University of Washington Bothell

5/28/2024

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Tamyla Nguyen	2024-05-30	Initial Draft	1.0
Raine Johnson	2024-05-30	Initial Draft	1.0

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the functionalities of the SplitPay system. This document focuses on the security enhancements and design changes required to address identified risks. This document will cover the system's features, external interfaces, and other nonfunctional requirements.

1.2 Document Conventions

This document adheres to standard conventions to ensure clarity and consistency. Section titles and key terms are in **bold** to emphasize their importance. Each requirement is uniquely identified by a number (e.g., REQ-1, REQ-2) for easy reference.

1.3 Intended Audience and Reading Suggestions

This document is intended for multiple stakeholders involved in the development, management, and use of SplitPay:

- **Developers:** To implement security features and enhancements.
- **Project Managers:** To oversee the development process and ensure that security requirements are met.
- **Testers:** To design test cases and validate that security features function correctly.
- **Users:** To understand the security measures in place and how they affect system use.

It is recommended to start with the Introduction for an overview, then proceed to the Overall Description for context, and finally focus on the System Features and Nonfunctional Requirements for specific details.

1.4 Product Scope

SplitPay is a web and mobile application designed to manage shared expenses among users, such as roommates, friends, or colleagues. The primary goal is to facilitate the tracking and splitting of payments in a secure manner. This document specifically addresses the security enhancements necessary to protect user data from threats such as SQL injection, DoS attacks, and identity theft. These enhancements are aligned with the organization's goal of providing a secure and reliable service that meets industry standards.

1.5 References

Digital Payment Security Risks and Best Practices | Comerica Bank, www.comerica.com/insights/business-finance/digital-payment-security-risks-and-best-practices.html. Accessed 31 May 2024.

Nguyen, Tamyla Thi, and Raine Johnson. "Detailed Security Risk Assessment For Splitpay System." 24 Apr. 2024, Accessed 30 May 2024.

"Top 10 E-Commerce Security Threats & Their Detailed Solution." *Astra Security Blog*, 27 Feb. 2024, www.getastra.com/blog/knowledge-base/ecommerce-security-threats/.

Umair, Rafay. "Software Requirements Specification for Splitpay Prepared By." *Academia.Edu*, 4 Dec. 2017, www.academia.edu/35335490/Software_Requirements_Specification_for_SplitPay_Prepared_by.

2. Overall Description

2.1 Product Perspective

SplitPay is designed as an advanced, secure platform for managing shared expenses. As a self-contained application, SplitPay operates independently but can integrate with other financial management systems through secure APIs. A high-level architecture diagram illustrating the system components and their interactions with external systems is provided in Appendix B.

2.2 Product Functions

SplitPay must perform several key functions to meet user needs:

- **Expense Tracking:** Record and categorize shared expenses.
- **Payment Splitting:** Calculate individual shares of expenses.
- **Secure Authentication:** Protect user accounts with robust authentication methods.
- **Data Protection:** Safeguard sensitive user information.
- **Alerting and Notifications:** Inform users of important activities and potential security issues.

These functions ensure a secure way of managing shared financial responsibilities.

2.3 User Classes and Characteristics

SplitPay is designed for the following user classes:

- **Regular Users**: Typically, non-technical individuals using the app for personal finance management. They require an intuitive interface and robust security features to protect their data.
- **Administrators**: Users with advanced privileges to manage accounts and oversee system operations. They need access to administrative tools and detailed security logs.
- **Developers**: Responsible for implementing and maintaining the software, requiring comprehensive documentation and access to secure coding practices.
- **Testers:** Focused on verifying the system's functionality and security, needing detailed test cases and scenarios.

Each user class has specific needs and characteristics that influence the system's design and functionality.

2.4 Operating Environment

SplitPay operates across various environments to accommodate diverse user needs:

- **Hardware:** Compatible with standard PCs, smartphones, and tablets.
- **Operating Systems:** Supports major operating systems, including Windows, macOS, Linux, iOS, and Android.
- **Software:** Accessible through modern web browsers (Chrome, Firefox, Safari) and mobile application environments.

The system must function seamlessly in these environments, ensuring consistent performance and security.

2.5 Design and Implementation Constraints

The development of SplitPay must adhere to several constraints:

- **Regulatory Compliance:** Adherence to industry standards and regulations, such as GDPR and PCI-DSS.
- **Technology Stack:** Use of specific development frameworks (e.g., React, Node.js) and databases (e.g., PostgreSQL) for consistency and maintainability.
- **Security Considerations**: Implementation of encryption, secure coding practices, and regular security audits.
- **Interoperability:** Ability to integrate with third-party financial systems via secure APIs.

These constraints guide the development process and ensure the system meets all necessary requirements.

2.6 User Documentation

Comprehensive user documentation will be provided to support all user classes:

- **User Manuals:** Detailed guides on using the application features.
- **Online Help:** Accessible help articles and tutorials within the application.
- **Security Best Practices:** Documentation on maintaining account security and understanding the implemented security measures.

These documents ensure users can effectively use the system and understand its security features.

2.7 Assumptions and Dependencies

Several assumptions and dependencies are identified for the successful implementation of SplitPay:

- **Internet Access:** Users will have reliable internet connectivity.
- **User Compliance:** Users will follow recommended security practices, such as using strong passwords.
- **Third-Party Components:** Dependence on third-party libraries and tools for certain functionalities, which must be kept updated.
- **Regular Updates:** Continuous improvement and updating of the system to address new security threats.

These factors are critical for ensuring the system operates effectively and securely.

3. External Interface Requirements

3.1 User Interfaces

SplitPay provides a user-friendly interface for interacting with the system:

- **Login Screens:** Secure login screens with support for two-factor authentication (2FA).
- **Dashboard:** A centralized dashboard for managing expenses and viewing financial summaries.
- **Notifications:** Alert users of security events and important actions.

The interface design prioritizes usability while ensuring security.

3.2 Hardware Interfaces

SplitPay interacts with device hardware to enhance security:

- **2FA Devices:** Integration with mobile devices for receiving authentication codes via SMS or authenticator apps.
- **Biometric Sensors:** Support for biometric authentication (e.g., fingerprint, facial recognition) on compatible devices.

These interfaces ensure secure user authentication and access.

3.3 Software Interfaces

SplitPay interfaces with various software components:

- **Database:** Secure interactions with the database using parameterized queries to prevent SQL injection.
- **APIs:** Secure API endpoints for data exchange with third-party systems and services.
- **Encryption Libraries:** Use of encryption libraries for data protection at rest and in transit.

These interfaces ensure secure and reliable data handling.

3.4 Communications Interfaces

SplitPay requires secure communication channels:

- **HTTPS:** All data transmission is secured using HTTPS.
- **Email Notifications:** Secure email services for sending notifications to users.
- **Web Sockets:** Secure real-time communication for instant updates and alerts.

These communication interfaces protect data integrity and confidentiality.

4. System Features

4.1 System Feature 1- SQL Injection Mitigation

4.1.1 Description and Priority

High Priority: SQL injection is a critical vulnerability that can compromise the database and expose sensitive user information. Mitigating this risk is essential to protect the system's integrity and user data.

4.1.2 Stimulus/Response Sequences

- ***Stimulus****: User submits data through forms (e.g., login, registration).*
- **Response**: System validates and sanitizes inputs, using parameterized queries to interact with the database, ensuring no malicious SQL code is executed.

4.1.3 Functional Requirements

REQ-1: The system shall validate all user inputs to prevent SQL injection.
REQ-2: The system shall use parameterized queries for all database interactions, ensuring that user inputs are treated as data, not executable code.

4.2 System Feature 2 – DoS Attack Mitigation

High Priority: Denial of Service (DoS) attacks can render the system unavailable to users by overwhelming it with traffic. Implementing measures to detect and mitigate these attacks is crucial for maintaining service availability.

4.2.2 Stimulus/Response Sequences

- **Stimulus**: System detects excessive requests from a single source or pattern indicating a DoS attack.
- **Response**: The system throttles the request rate, temporarily blocks the offending IP, and alerts administrators to investigate further.

4.2.3 Functional Requirements

REQ-3: The system shall implement rate limiting to prevent DoS attacks.
REQ-4: The system shall monitor network traffic for anomalies and potential DoS attack patterns.

4.3 System Feature 3 – Identity Theft Prevention

4.3.1 Description and Priority

High Priority: Identity theft poses a significant risk to users, potentially leading to unauthorized access to sensitive information. Implementing robust authentication mechanisms is essential to prevent such occurrences.

4.3.2 Stimulus/Response Sequences

Stimulus: User attempts to log in.
Response: The system requires both a password and a second factor of authentication (e.g., a code sent to the user's mobile device). Upon successful authentication, the user gains access to the system.

4.3.3 Functional Requirements

REQ-5: The system shall enforce strong password policies, requiring a minimum length and complexity.
REQ-6: The system shall support two-factor authentication (2FA) for all user logins.
REQ-7: The system shall encrypt sensitive data both at rest and in transit to protect against unauthorized access.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must support up to 1000 concurrent users without significant performance degradation. Page load times should not exceed 2 seconds under normal operational conditions. Scalability is essential to handle peak loads efficiently.

5.2 Safety Requirements

Data integrity and availability should be maintained at all times. The system should include regular backups and data recovery procedures to prevent data loss. In case of system failure, the recovery time objective (RTO) should be within 1 hour to minimize disruption.

5.3 Security Requirements

SplitPay must comply with industry standards such as OWASP and NIST for security. All data must be encrypted using strong encryption protocols (e.g., AES-256). Regular security audits and penetration testing should be conducted to identify and mitigate vulnerabilities

5.4 Software Quality Attributes

The system must be reliable, with an uptime of 99.9% or higher. Maintainability is ensured through modular code design and comprehensive documentation. Testability is facilitated by detailed test cases and automated testing tools. Usability must be prioritized, ensuring that security measures do not compromise user experience.

5.5 Business Rules

Access to administrative functions must be restricted to authorized personnel only. Regular users should only access their own data and shared expense information. Business rules should enforce clear access controls and auditing to ensure compliance with organizational policies and regulations.

6. Other Requirements

All third-party components and libraries used in SplitPay must comply with the security standards outlined in this document. The system must be regularly updated with patches and security fixes to address emerging threats. Continuous monitoring and incident response plans should be in place to handle security incidents promptly.

Appendix A: Glossary

- ***SQL Injection****: A code injection technique that exploits vulnerabilities in an application's software by inserting malicious SQL code in input fields.*
- ***DoS Attack****: Denial of Service attack, an attempt to make a machine or network resource unavailable to its intended users.*
- ***2FA****: Two-Factor Authentication, a method of confirming users' identities using two different factors.*
- ***AES-256****: Advanced Encryption Standard with a 256-bit key length, used for encrypting data*.

Appendix B: Analysis Models

- ***Data Flow Diagrams****: Illustrate the flow of data within the system, highlighting points of input validation and data processing.*
- ***State-Transition Diagrams****: Show the different states of the system and transitions triggered by user actions or system events.*

Appendix C: To Be Determined List

- ***User Interface Design Specifications****: Detailed design documents for the user interfaces.*
- ***Comprehensive Testing Plan****: A detailed plan outlining the testing strategy, test cases, and scenarios for new security features.*