Design 2: IEEE 830 Standard

- **Introduction**
  - 1.1 Purpose: This document specifies the requirements for the Secure Messaging System.
  - 1.2 Scope: This SRS covers all aspects of the system, including user authentication, message encryption, storage, and the web interface.
  - 1.3 Intended Audience and Reading Suggestions: This document is intended for the developer, testers, and stakeholders.
  - 1.4 Project Scope: The project aims to create a secure messaging platform with end-to-end encryption.
  - 1.5 References: IEEE 830-1998, Standard for Software Requirements Specifications
- **Overall Description**
  - 2.1 Product Perspective: The Secure Messaging System is a standalone web application.
  - 2.2 Product Features:
    - User Authentication
    - Message Encryption/Decryption
    - Secure Message Storage
    - Inbox/Outbox System
  - 2.3 User Classes and Characteristics:
    - Security-conscious journalist: Tech-savvy, needs secure communication.
    - Privacy-focused business owner: Needs to protect confidential data.
  - 2.4 Operating Environment: Web browser (Chrome, Firefox,), Python 3.x, Flask/Django, SQLite/MySQL.
  - 2.5 Design and Implementation Constraints:
    - Time constraints
    - Limited budget
    - Security best practices
  - 2.6 User Documentation: A user guide will be provided.
  - 2.7 Assumptions and Dependencies:
    - Users have internet access.
    - Cryptography libraries are reliable.
- **System Features**
  - 3.1 User Authentication:
    - Description: Allows users to register and log in securely.
    - Requirements: Hashed and salted passwords using bcrypt/Argon2.
  - 3.2 Message Encryption/Decryption:
    - Description: Encrypts messages before sending and decrypts them upon receipt.
    - Requirements: End-to-end encryption with AES or RSA.
  - 3.3 Secure Message Storage:
    - Description: Stores messages securely in the database.

            - Requirements: Messages are encrypted at rest.
- **External Interface Requirements**
    - 4.1 User Interfaces: Web-based interface with inbox/outbox views.
    - 4.2 Hardware Interfaces: Standard computer hardware.
    - 4.3 Software Interfaces: Flask/Django framework, SQLite/MySQL database.
    - 4.4 Communications Interfaces: HTTPS for secure communication.
- **Other Nonfunctional Requirements**
    - 5.1 Performance Requirements: Message delivery latency < 0.5 seconds.
    - 5.2 Security Requirements: Protection against XSS, SQL injection, and other vulnerabilities.
    - 5.3 Software Quality Attributes: Usability, reliability, maintainability.
- **Other Requirements**
    - Compliance with security best practices.
- **Appendix A: Glossary**
- **Appendix B: Analysis Models**
- **Appendix C: Issues List**
- **Revision History**
- **Design 3: Agile-Focused SRS**
- **Introduction**
    - 1.1 Purpose: This document outlines the requirements for the Secure Messaging System, using an agile approach.
    - 1.2 Project Goals and Objectives: To create a secure and user-friendly messaging application.
    - 1.3 Target Audience: Users who need secure communication.
- **User Stories**
    - As a registered user, I want to send encrypted messages so that my communication remains private.
        - Acceptance Criteria:
            - The message is encrypted before sending.
            - The recipient can decrypt the message.
    - As a user, I want to log in securely so that my messages are protected.
        - Acceptance Criteria:
            - The system uses hashed and salted passwords.
            - Login attempts are protected against brute-force attacks.
    - As a user, I want to receive messages in a clear and organized inbox so that I can easily manage my communication.
        - Acceptance Criteria:
            - Messages are displayed in chronological order.
            - Users can easily delete messages.
- **System Overview**
    - Technology Stack: Python (Flask/Django), SQLite/MySQL, cryptography library.
- **Non-Functional Requirements**
    - Performance: Message delivery latency < 0.5 seconds.
    - Security: Protection against common web vulnerabilities.

    - Usability: Clean and intuitive user interface.
    - Scalability: The system should be able to handle a growing number of users.
- **Open Issues and Risks**
    - Potential security vulnerabilities in cryptography libraries.
    - Time constraints for development.