Distilling Software Requirement Specifications

Discussion w/ Claude

https://claude.ai/chat/905021f9-8e89-4f5a-aedf-81b9e4306429

I will be providing you document on the topic of writing software requirements. Please create 3 page distillation of their content.

Here is a 3-page distillation of the key points from the provided documents on writing software requirements:

1. Introduction
    - Software requirements specifications (SRS) are critical for clearly defining what functionality and characteristics a software system needs to have. A good SRS allows customers to accurately describe their needs and developers to understand exactly what to build.
    - The SRS serves as the basis for the entire software development lifecycle - for design, implementation, testing, delivery and maintenance. It is a key point of agreement between customers and suppliers.
2. Characteristics of a Good SRS An SRS should be:
    - Correct - every requirement has been validated with the customer
    - Unambiguous - each requirement has only one possible interpretation
    - Complete - all significant requirements are included and defined
    - Consistent - no conflicts between individual requirements
    - Ranked for importance/stability - helps manage scope and changes
    - Verifiable - it's possible to check whether each requirement is met
    - Modifiable - easy to make changes while maintaining consistency
    - Traceable - origin of each requirement is clear and it can be referenced in future artifacts

Other important quality attributes include saying what is needed without unnecessary design detail, using consistent terminology, and being understandable by all stakeholders.

3. SRS Development Process
    - Requirements elicitation - work with stakeholders to discover and clarify their needs through workshops, interviews, scenarios, prototypes, etc. Consider the operational concepts and environment.
    - Requirements analysis - develop a technical understanding of requirements, model the system, resolve conflicts, assess feasibility, and allocate to architecture elements.
    - Requirements specification - document the requirements in a well-organized SRS with natural language and diagrams. Specify functional requirements, interfaces, quality attributes, design constraints, etc.
    - Requirements validation - confirm the documented requirements match actual stakeholder intent. Review the SRS and get sign-off.
    - Requirements management - manage changes to requirements, maintain traceability, track status throughout the lifecycle.
4. SRS Outline A typical SRS document outline includes:
    - Introduction - purpose, scope, definitions, overview
    - Overall description - product perspective, functions, user classes, operating environment, constraints, assumptions, dependencies
    - Specific requirements - external interfaces, functional requirements, quality attributes, design constraints
    - Supporting information - tables, figures, appendices
5. Summary: Writing good software requirements is challenging but critical to the success of any software project. It requires clear communication between customers and developers and rigorous analysis to ensure the requirements are complete, consistent, feasible and verifiable. The SRS becomes the foundation for system design, implementation and testing. Managing requirements changes and traceability throughout the lifecycle is also essential. Organizations should establish standards and processes for high-quality requirements engineering practices.

Assistant: Here is a more concise 3-page distillation focused on the most essential points:

Page 1: Essential Characteristics of Software Requirements A software requirements specification (SRS) defines the required functionality, performance, interfaces, and quality attributes of a software system. To effectively fulfill this purpose, the requirements in an SRS must be:

- Unambiguous - open to only one interpretation
- Testable - can be verified through testing or other means
- Clear - easily understandable by all stakeholders
- Correct - accurately represents actual customer needs
- Consistent - does not conflict with other requirements
- Modifiable - can adapt to necessary changes
- Traceable - can be linked to design, code and tests
- Feasible - technically and economically possible
- Complete - does not omit any significant requirements

Involving stakeholders in reviewing the SRS is key to validating the requirements. The SRS should focus on what the system needs to do, not design details. Requirements should be uniquely identified for traceability and ranked for importance and stability.

Page 2: The SRS Development Process Developing an SRS involves:

- Elicitation - Gather requirements from stakeholders through interviews, workshops, document analysis, prototyping, use cases, and other interactive techniques. Focus on understanding user needs and the operational environment.
- Analysis - Examine requirements for necessity, consistency, completeness, feasibility and testability. Model the system from different perspectives. Resolve conflicts and prioritize requirements.
- Specification - Write the SRS in natural language and diagrams. Organize content appropriately, using an outline like:
  1. Introduction - purpose, scope, definitions, references, overview
  2. Overall description - product perspective, user needs, assumptions and constraints
  3. Specific requirements - functions, performance, interfaces, quality attributes
  4. Supporting info - tables, appendices, index
- Validation - Review the SRS with stakeholders to ensure it accurately captures their needs. Use techniques like prototyping and traceability analysis.
- Management - Control changes to baselined requirements. Maintain bidirectional traceability from source to implementation. Track metrics like volatility.

The SRS evolves iteratively throughout development based on feedback and approved changes. It requires collaboration between customers and the product team.

Page 3: Applying the SRS The SRS provides benefits throughout the software lifecycle:

- Contractual basis between client and supplier
- Input for project planning and cost/schedule estimation
- Baseline to assess proposed changes against
- Reference for designers to ensure all requirements are met
- Source of test cases for verification
- Aid for new team members to understand the system
- Guide for product maintenance and enhancement

To achieve these benefits, the SRS must be kept up-to-date as an authoritative source of requirements. This requires a robust change management process and tools to facilitate access to the latest baselined version.

The level of detail and formality of an SRS should be appropriate for the scale, complexity and risk of the project. Industry standards recommend using a well-defined template for a complete SRS. However, agile approaches may employ lightweight techniques to incrementally specify and refine requirements.

Ultimately, the true measure of an SRS is whether it enables developers to deliver a system that satisfies users. This relies upon establishing clear, comprehensive requirements and effectively managing them as development proceeds. Investing in disciplined requirements engineering practices pays dividends in improving the quality and success of software projects.

From these document please provide:

1. A template for an SRS document
2. A template for documenting a single requirement
3. An example based on the template for a single requirement

Sure, I can provide you with those templates and an example based on the information from the documents.

Template for an SRS document:

1 Introduction

1. Purpose
2. Scope
3. Definitions, Acronyms, and Abbreviations
4. References
5. Overview

2 Overall Description

1. Product Perspective
2. Product Functions
3. User Classes and Characteristics
4. Operating Environment
5. Design and Implementation Constraints
6. User Documentation
7. Assumptions and Dependencies

3 External Interface Requirements

1. User Interfaces
2. Hardware Interfaces
3. Software Interfaces
4. Communications Interfaces

4 System Features

1. System Feature 1
2. System Feature 2 (and so on for each feature)

5 Other Nonfunctional Requirements

1. Performance Requirements
2. Safety Requirements
3. Security Requirements
4. Software Quality Attributes
5. Business Rules

6 Other Requirements Appendix

A. Glossary Appendix

B. Analysis Models Appendix

C. Issues List

Template for documenting a single requirement:

- Requirement ID: A unique identifier for the requirement.
- Requirement Description: A clear, concise description of the requirement.
- Rationale: The reason behind the requirement, its business or technical justification.
- Source: The origin of the requirement (e.g., customer, business need, regulation).
- Priority: The importance of the requirement (e.g., essential, desirable, optional).
- Dependencies: Other requirements that have an impact on this one.
- Acceptance Criteria: Conditions that must be met for the requirement to be considered satisfied.

Example of a single requirement using the template:

- Requirement ID: REQ-LOGIN-001
- Requirement Description: The system shall allow users to log in using their email address and password.
- Rationale: Users need a secure way to access their personal data and system functionality.

- Source: Customer request
- Priority: Essential
- Dependencies: REQ-USER-MGMT-001 (User management system requirement)
- Acceptance Criteria:
    1. Users can enter their email and password on the login screen.
    2. The system verifies the email and password against the user database.
    3. If the credentials are valid, the user is logged in and directed to their dashboard.
    4. If the credentials are invalid, an error message is displayed, prompting the user to try again.

This example follows the template, providing a unique identifier, a clear description, the rationale behind the requirement, its source and priority, any dependencies on other requirements, and specific criteria that can be used to verify if the implemented feature satisfies the stated requirement.

Please create a complete and detailed example SRS for a twitter-like application/system.

Here's a complete and detailed example SRS for a Twitter-like application/system:

1. Introduction
    1. Purpose This document specifies the requirements for a Twitter-like social media application, named "Chatter." The application will allow users to share short messages, follow other users, and interact with their posts.
    2. Scope Chatter will be a web-based application accessible through modern web browsers. The application will also have a mobile-friendly version optimized for smartphones.
    3. Definitions, Acronyms, and Abbreviations
        - Post: A short message of up to 280 characters shared by a user.
        - Repost: Sharing another user's post on one's own timeline.
        - Like: Expressing approval of a post.
        - Follow: Subscribing to another user's posts.
    4. References
        - IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
    5. Overview: The remainder of this document provides a detailed description of the Chatter application. Section 2 gives an overall description, Section 3 outlines external interface requirements, Section 4 details the system features, and Section 5 describes other nonfunctional requirements.
2. Overall Description
    1. Product Perspective Chatter is a standalone web application that will interact with a backend database to store user data, posts, and interactions.
    2. Product Functions The main functions of Chatter will be:
        - User registration and authentication
        - Posting short messages
        - Following other users
        - Interacting with posts (liking, reposting)
        - User profile management
    3. User Classes and Characteristics Chatter will have two main user classes:
        - Regular Users: Can post messages, follow other users, and interact with posts.
        - Administrators: Have all the privileges of regular users, plus the ability to manage user accounts and moderate content.
    4. Operating Environment
        1. Chatter will operate on a web server and will be accessible through web browsers on PCs and mobile devices.
    5. Design and Implementation Constraints
        - The application will be developed using modern web technologies (e.g., HTML5, CSS3, JavaScript).
        - The backend will be developed using a suitable web framework and database system.
    6. User Documentation
        1. User documentation will include online help and a PDF user manual.
    7. Assumptions and Dependencies
        - Users will have access to a web browser and internet connection.
        - The application will depend on the availability of the web server and database system.
3. External Interface Requirements
    1. User Interfaces
        - The application will have a responsive web interface optimized for desktop and mobile screens.
        - The interface will be intuitive and easy to navigate.
    2. Hardware Interfaces

    - The application will not have any direct hardware interfaces.
  3. Software Interfaces
     - The application will interface with a backend database to store and retrieve data.
  4. Communications Interfaces
     - The application will use HTTP/HTTPS for communication between the client and server.
4. System Features
   1. User Registration and Authentication
      1. Description Users can create new accounts by providing a unique username, email address, and password. They can then log in using their credentials.
      2. Functional Requirements
         - REQ-AUTH-001: The system shall allow users to create new accounts by providing a unique username, email address, and password.
         - REQ-AUTH-002: The system shall authenticate users based on their username and password.
         - REQ-AUTH-003: The system shall allow users to log out of their accounts.
   2. Posting Messages
      1. Description Authenticated users can post short messages of up to 280 characters.
      2. Functional Requirements
         - REQ-POST-001: The system shall allow authenticated users to post messages of up to 280 characters.
         - REQ-POST-002: The system shall display posted messages on the user's timeline and the timelines of their followers.
   3. Following Users
      1. Description Users can follow other users to see their posts in their timeline.
      2. Functional Requirements
         - REQ-FOLLOW-001: The system shall allow users to follow other users.
         - REQ-FOLLOW-002: The system shall display posts from followed users in the user's timeline.
   4. Interacting with Posts
      1. Description Users can interact with posts by liking or reposting them.
      2. Functional Requirements
         - REQ-INTERACT-001: The system shall allow users to like posts.
         - REQ-INTERACT-002: The system shall allow users to repost messages.
   5. User Profile Management
      1. Description Users can manage their profile information, including their display name, bio, and profile picture.
      2. Functional Requirements
         - REQ-PROFILE-001: The system shall allow users to update their display name, bio, and profile picture.
         - REQ-PROFILE-002: The system shall display a user's profile information to other users.
5. Other Nonfunctional Requirements
   1. Performance Requirements
      - REQ-PERF-001: The system shall load pages within 2 seconds on average.
      - REQ-PERF-002: The system shall handle at least 1000 concurrent users.
   2. Security Requirements
      - REQ-SEC-001: The system shall encrypt passwords using a secure hashing algorithm.
      - REQ-SEC-002: The system shall protect against common web vulnerabilities (e.g., XSS, CSRF, SQL injection).
   3. Software Quality Attributes
      - REQ-QUAL-001: The system shall be available 99.9% of the time.
      - REQ-QUAL-002: The system shall be easily maintainable and extensible.

This example SRS covers the main aspects of a Twitter-like application, including user management, posting messages, following users, interacting with posts, and user profiles. It also includes nonfunctional requirements related to performance, security, and quality attributes. Each requirement is given a unique identifier and a clear description.

Using ALL the content from the given PDF files please create a complete and detailed example SRS for a twitter-like application/system as specified by the NASA and IEEE documents.

Here is a complete and detailed example SRS for a Twitter-like application/system, based on the content and recommendations from the provided NASA and IEEE documents:

Software Requirements Specification (SRS)

Chatter: A Twitter-like Social Media Application

1. Introduction
    1. Purpose
       This document specifies the software requirements for Chatter, a Twitter-like social media application that allows users to share short messages, follow other users, and interact with their posts. The purpose of this SRS is to provide a clear and detailed description of the functional and non-functional requirements for the Chatter application.
    2. Scope
       The scope of this SRS covers the software requirements for the Chatter application, including user registration and authentication, posting messages, following users, interacting with posts (liking, commenting, reposting), user profile management, and administration features. The Chatter application will be developed as a web-based application with a responsive user interface for desktop and mobile devices.
    3. Definitions, Acronyms, and Abbreviations
        - Post: A short message of up to 280 characters shared by a user.
        - Comment: A response to a post, also limited to 280 characters.
        - Repost: Sharing another user's post on one's own timeline.
        - Like: Expressing approval of a post or comment.
        - Follow: Subscribing to another user's posts.
        - SRS: Software Requirements Specification
        - UI: User Interface
        - API: Application Programming Interface
    4. References
        - IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
        - NASA-STD-8739.8: Software Assurance and Software Safety Standard
        - NASA-HDBK-2203: NASA Software Engineering Handbook
    5. Overview
       The remainder of this SRS is organized as follows: Section 2 provides an overall description of the Chatter application, including its perspective, functions, user characteristics, constraints, and dependencies. Section 3 specifies the detailed software requirements, covering functional, external interface, performance, and other quality attributes. Section 4 includes supporting information, such as a traceability matrix and sample user interface mockups.
2. Overall Description
    1. Product Perspective
       Chatter is a standalone web application that enables users to share short messages, follow other users, and interact with posts through likes, comments, and reposts. The application will be accessible through modern web browsers on desktop and mobile devices. Chatter will interact with a backend server and database to store and retrieve user data, posts, and interactions.
    2. Product Functions
       The main functions of the Chatter application include:
        - User registration and authentication
        - Posting messages (up to 280 characters)
        - Following other users
        - Displaying a timeline of posts from followed users
        - Interacting with posts through likes, comments, and reposts
        - User profile management (e.g., updating profile information, profile picture)
        - Search functionality for users and posts
        - Administration features for managing users and moderating content
    3. User Classes and Characteristics
       Chatter will support three main user classes:
        1. Regular Users: Can register, authenticate, post messages, follow other users, interact with posts, and manage their profile.
        2. Administrators: Have all the privileges of regular users, plus the ability to manage user accounts and moderate content.
        3. Anonymous Users: Can view public posts and profiles but cannot interact or post messages until they register and authenticate.

       The application will be designed to accommodate users with varying levels of technical expertise and social media experience.
    4. Operating Environment
       Chatter will be a web-based application, operating on a web server and accessible through modern web browsers (e.g., Chrome, Firefox, Safari) on desktop and mobile devices. The server-side components will run on a Linux-based operating system, utilizing a web application framework (e.g., Ruby on Rails, Django) and a relational database

    management system (e.g., PostgreSQL, MySQL).
5. Design and Implementation Constraints
    - The application will be developed using modern web technologies, such as HTML5, CSS3, and JavaScript.
    - The server-side components will be implemented using a suitable web application framework and follow RESTful API design principles.
    - The application will be designed with scalability, performance, and security in mind.
    - The user interface will be responsive and optimized for various screen sizes and devices.
    - The application will comply with relevant web accessibility guidelines (e.g., WCAG 2.1).
6. User Documentation
    User documentation for Chatter will include:
    - Online help and tutorials integrated into the application
    - A user manual in PDF format, covering all features and functionality
    - FAQ section on the application website
    - Video tutorials demonstrating key features and workflows
7. Assumptions and Dependencies
    - Users will have access to a modern web browser and a stable internet connection.
    - The application will depend on the availability and performance of the web server and database infrastructure.
    - Integration with third-party services (e.g., email providers for notifications) will rely on the availability and stability of those services.

3. Specific Requirements
    1. Functional Requirements
        1. User Registration and Authentication
            - FR-AUTH-1: The system shall allow users to register by providing a unique username, email address, and password.
            - FR-AUTH-2: The system shall validate the format and uniqueness of the provided username and email address during registration.
            - FR-AUTH-3: The system shall securely store user passwords using a strong hashing algorithm and salting mechanism.
            - FR-AUTH-4: The system shall authenticate users based on their provided username or email address and password.
            - FR-AUTH-5: The system shall allow users to log out of their accounts, terminating their session.
        2. Posting Messages
            - FR-POST-1: The system shall allow authenticated users to post messages containing up to 280 characters.
            - FR-POST-2: The system shall display posted messages on the user's timeline and the timelines of their followers, ordered chronologically.
            - FR-POST-3: The system shall allow users to include images, videos, or links in their posts.
        3. Following Users
            - FR-FOLLOW-1: The system shall allow authenticated users to follow other users.
            - FR-FOLLOW-2: The system shall display a user's follower count and a list of their followers.
            - FR-FOLLOW-3: The system shall allow users to unfollow previously followed users.
        4. Interacting with Posts
            - FR-INTERACT-1: The system shall allow authenticated users to like posts and comments.
            - FR-INTERACT-2: The system shall allow authenticated users to comment on posts, with comments limited to 280 characters.
            - FR-INTERACT-3: The system shall allow authenticated users to repost messages, sharing them on their own timeline.
            - FR-INTERACT-4: The system shall display the count of likes, comments, and reposts for each post.
        5. User Profile Management
            - FR-PROFILE-1: The system shall allow authenticated users to update their profile information, including display name, bio, location, and website.
            - FR-PROFILE-2: The system shall allow authenticated users to upload and update their profile picture.
            - FR-PROFILE-3: The system shall display a user's profile information, including their post count, follower count, and following count.
        6. Search Functionality
            - FR-SEARCH-1: The system shall provide a search feature that allows users to search for other users by username or display name.
            - FR-SEARCH-2: The system shall provide a search feature that allows users to search for posts containing specific keywords or hashtags.
        7. Administration Features
            - FR-ADMIN-1: The system shall allow administrators to view and manage user accounts, including the ability to suspend or delete accounts.

        - FR-ADMIN-2: The system shall allow administrators to moderate content, including the ability to remove posts and comments that violate community guidelines.
2. External Interface Requirements
    1. User Interfaces
        - EIR-UI-1: The system shall provide a responsive web-based user interface that adapts to different screen sizes and devices.
        - EIR-UI-2: The user interface shall be intuitive, visually appealing, and easy to navigate.
        - EIR-UI-3: The user interface shall provide clear feedback and confirmation messages for user actions.
        - EIR-UI-4: The user interface shall display appropriate error messages and guidance when user input is invalid or an error occurs.
    2. Hardware Interfaces
        - EIR-HW-1: The system shall be compatible with standard desktop and mobile devices with modern web browsers.
        - EIR-HW-2: The system shall not require any specialized hardware interfaces.
    3. Software Interfaces
        - EIR-SW-1: The system shall integrate with a backend server and database for data storage and retrieval.
        - EIR-SW-2: The system shall integrate with a secure authentication and authorization system.
        - EIR-SW-3: The system shall provide a RESTful API for potential integration with third-party applications or services.
    4. Communications Interfaces
        - EIR-COMM-1: The system shall use HTTPS for secure communication between the client and server.
        - EIR-COMM-2: The system shall use standard HTTP methods (GET, POST, PUT, DELETE) for API communication.
3. System Features
    1. Performance Requirements
        - PERF-1: The system shall support a minimum of 10,000 concurrent users without significant performance degradation.
        - PERF-2: The system shall respond to user actions within 2 seconds under normal load conditions.
        - PERF-3: The system shall handle a minimum of 100 new user registrations per minute.
        - PERF-4: The system shall handle a minimum of 500 new posts per minute.
    2. Safety and Security Requirements
        - SAFE-1: The system shall protect sensitive user data, such as passwords and personal information, from unauthorized access.
        - SAFE-2: The system shall implement secure authentication and authorization mechanisms to prevent unauthorized access to user accounts.
        - SAFE-3: The system shall validate and sanitize user input to prevent common web vulnerabilities, such as cross-site scripting (XSS) and SQL injection attacks.
        - SAFE-4: The system shall implement rate limiting and captcha mechanisms to prevent automated attacks and abuse.
    3. Software Quality Attributes
        - QUAL-1: The system shall be designed for maintainability, with a modular architecture and clear separation of concerns.
        - QUAL-2: The system shall be designed for scalability, allowing for horizontal scaling of server infrastructure to accommodate growth in user base and traffic.
        - QUAL-3: The system shall be designed for reliability, with appropriate error handling, logging, and monitoring mechanisms.
        - QUAL-4: The system shall be designed for usability, with a focus on providing a simple, intuitive, and efficient user experience.
4. Logical Database Requirements
    - DB-1: The system shall utilize a relational database management system for persistent data storage.
    - DB-2: The database schema shall be designed to efficiently store and retrieve user data, posts, comments, likes, and relationships between entities.
    - DB-3: The database shall support transactions to ensure data integrity and consistency.
    - DB-4: The database shall be designed with appropriate indexing and query optimization to support high-performance read and write operations.
5. Design Constraints
    - DC-1: The system shall be designed using a responsive web design approach to ensure compatibility with various screen sizes and devices.
    - DC-2: The system shall follow RESTful API design principles for server-side components.
    - DC-3: The system shall use a modern web application framework and follow industry best practices for security, performance, and maintainability.

6. Software System Attributes
    1. Reliability
        - RELY-1: The system shall have an uptime of at least 99.9%, excluding scheduled maintenance.
        - RELY-2: The system shall automatically recover from common failures, such as server crashes or network interruptions.
    2. Availability
        - AVAIL-1: The system shall be available 24 hours a day, 7 days a week, with the exception of scheduled maintenance windows.
        - AVAIL-2: Scheduled maintenance windows shall not exceed 2 hours per month and shall be communicated to users in advance.
    3. Security
        - SECR-1: The system shall employ encryption techniques to protect sensitive data in transit and at rest.
        - SECR-2: The system shall implement secure user authentication and authorization mechanisms, such as multi-factor authentication and role-based access control.
        - SECR-3: The system shall regularly undergo security audits and penetration testing to identify and address potential vulnerabilities.
    4. Maintainability
        - MAIN-1: The system shall be designed with a modular architecture, promoting code reuse and facilitating future enhancements and maintenance.
        - MAIN-2: The system shall follow coding standards and best practices to ensure code readability, consistency, and maintainability.
        - MAIN-3: The system shall be accompanied by comprehensive documentation, including architecture diagrams, API specifications, and code comments.
    5. 5 Portability
        - PORT-1: The system shall be designed to be portable across different web server platforms and database management systems.
        - PORT-2: The system shall use platform-independent technologies and standards, such as HTML, CSS, and JavaScript, to ensure compatibility with various web browsers and devices.

4. Supporting Information
    1. Traceability Matrix
       A traceability matrix will be maintained to map software requirements to design components, test cases, and other relevant artifacts throughout the development lifecycle.
    2. Sample User Interface Mockups
       Sample user interface mockups will be created to provide a visual representation of the Chatter application's key screens and workflows, aiding in the communication and validation of requirements.

This example SRS document incorporates the key elements and recommendations from the NASA and IEEE guidelines, including a clear structure, detailed functional and non-functional requirements, and considerations for software quality attributes. The document also includes supporting information, such as a traceability matrix and sample user interface mockups, to aid in the development and validation of the Chatter application.

Please create complete and detailed requirements following the template you previously provided for the following 6 requirements:

- FR-AUTH-2
- FR-POST-3
- FR-INTERACT-4
- EIR-SW-3
- PERF-4
- QUAL-1

Certainly! Here are the complete and detailed requirements for the specified items, following the template I provided earlier:

FR-AUTH-2: User Registration Input Validation

- Requirement ID: FR-AUTH-2
- Requirement Description: The system shall validate the format and uniqueness of the provided username and email address during registration.
- Rationale: To ensure that user input is consistent, valid, and unique, preventing duplicate or invalid accounts.
- Source: Functional Requirements Document
- Priority: High

- Dependencies: FR-AUTH-1 (User Registration)
- Acceptance Criteria:
  1. The system shall validate that the provided username is between 3 and 20 characters long and consists of alphanumeric characters, underscores, or hyphens.
  2. The system shall validate that the provided email address is in a valid format (e.g., user@domain.com).
  3. The system shall check that the provided username and email address are unique and not already associated with an existing account.
  4. If the provided username or email address is invalid or already in use, the system shall display an appropriate error message and prompt the user to enter a different value.

FR-POST-3: Multimedia Content in Posts

- Requirement ID: FR-POST-3
- Requirement Description: The system shall allow users to include images, videos, or links in their posts.
- Rationale: To enable users to share rich multimedia content and enhance the user experience.
- Source: Functional Requirements Document
- Priority: Medium
- Dependencies: FR-POST-1 (Posting Messages)
- Acceptance Criteria:
  1. The system shall provide an option for users to attach images, videos, or links when creating a new post.
  2. The system shall support common image formats, such as JPEG, PNG, and GIF.
  3. The system shall support common video formats, such as MP4 and WebM.
  4. The system shall validate that the attached files do not exceed a predefined size limit (e.g., 5MB for images, 50MB for videos).
  5. The system shall display the attached images, videos, or links as part of the post on the user's timeline and the timelines of their followers.

FR-INTERACT-4: Display Interaction Counts

- Requirement ID: FR-INTERACT-4
- Requirement Description: The system shall display the count of likes, comments, and reposts for each post.
- Rationale: To provide users with a sense of engagement and popularity of posts.
- Source: Functional Requirements Document
- Priority: Medium
- Dependencies: FR-INTERACT-1 (Liking Posts and Comments), FR-INTERACT-2 (Commenting on Posts), FR-INTERACT-3 (Reposting Messages)
- Acceptance Criteria:
  1. The system shall display the count of likes, comments, and reposts for each post below the post content.
  2. The like, comment, and repost counts shall be updated in real-time as users interact with the post.
  3. If a post has no likes, comments, or reposts, the respective count shall be displayed as "0" or not shown at all.

EIR-SW-3: RESTful API

- Requirement ID: EIR-SW-3
- Requirement Description: The system shall provide a RESTful API for potential integration with third-party applications or services.
- Rationale: To enable interoperability and extend the functionality of the Chatter application through integration with external systems.
- Source: External Interface Requirements Document
- Priority: Low
- Dependencies: None
- Acceptance Criteria:
  1. The system shall expose a well-documented RESTful API that allows third-party applications to interact with the Chatter application.
  2. The API shall support standard HTTP methods (GET, POST, PUT, DELETE) for retrieving and manipulating data.
  3. The API shall use JSON as the primary data format for request and response payloads.
  4. The API shall implement proper authentication and authorization mechanisms to ensure secure access to data and functionality.
  5. The API shall provide comprehensive documentation, including endpoint descriptions, request/response examples, and error codes.

PERF-4: New Post Handling Capacity

- Requirement ID: PERF-4
- Requirement Description: The system shall handle a minimum of 500 new posts per minute.
- Rationale: To ensure that the system can accommodate a high volume of user activity and maintain responsiveness.
- Source: Performance Requirements Document
- Priority: High
- Dependencies: None
- Acceptance Criteria:
  1. The system shall be able to process and store at least 500 new posts per minute without experiencing significant performance degradation.
  2. The system shall maintain an average response time of less than 2 seconds for post creation requests under the specified load.
  3. The system shall be tested using load testing tools to simulate the required post creation volume and validate its performance.

QUAL-1: Maintainable System Architecture

- Requirement ID: QUAL-1
- Requirement Description: The system shall be designed for maintainability, with a modular architecture and clear separation of concerns.
- Rationale: To facilitate easy maintenance, debugging, and future enhancements of the Chatter application.
- Source: Software Quality Attributes Document
- Priority: High
- Dependencies: None
- Acceptance Criteria:
  1. The system architecture shall be divided into logical modules or components, each responsible for a specific functionality or feature.
  2. The modules shall be loosely coupled, with well-defined interfaces and minimal dependencies between them.
  3. The system shall follow a layered architecture, separating the presentation, business logic, and data access layers.
  4. The codebase shall be organized into meaningful directories and files, reflecting the system's modular structure.
  5. The system shall use appropriate design patterns and principles, such as dependency injection and separation of concerns, to achieve a maintainable architecture.

These detailed requirements provide a clear and comprehensive description of the specific functionalities, performance expectations, and quality attributes of the Chatter application. They follow the template, including the requirement ID, description, rationale, source, priority, dependencies, and acceptance criteria.