Software Requirements Specification (SRS) for Social Media Database Management System

1. Introduction

1.1 Purpose

The purpose of this Social Media Database Management System (SMDMS) is to provide a robust, scalable, and efficient backend infrastructure for a social media platform. The system will handle user information, posts, comments, likes, media uploads, and relationships between users. This document outlines the requirements and features necessary to support the platform's operations and ensure seamless functionality and performance.

1.2 Intended Audience

This document is intended for:

Developers working on implementing the backend database and API for the social media platform.

Database Administrators responsible for managing, optimizing, and securing the database.

Project Managers overseeing the project to ensure timely and accurate delivery.

Quality Assurance Testers who will validate the functionality and performance of the system.

Stakeholders interested in understanding the capabilities and scope of the database system.

1.3 Scope

The scope of the SMDMS includes:

Storing, retrieving, and updating user data, posts, comments, and likes.

Managing relationships between users (such as follows).

Handling multimedia content uploads (photos, videos).

Ensuring data consistency, security, and efficient performance.

Supporting query operations for the social media application's front-end.

This SRS covers the requirements for a database system that will interact with a social media platform front-end and is intended to handle millions of users and posts efficiently.

1.4 Definition - Users

In this system, the term User refers to any individual who registers on the social media platform. Users can perform various actions, including posting content, liking and commenting on posts, following other users, and using hashtags to categorize posts.

2. Overall Description

2.1 User Interfaces

The database will interact with user-facing components through API endpoints. These interfaces are not directly visible to end-users but are accessed by the application layer to perform CRUD (Create, Read, Update, Delete) operations on data like user profiles, posts, comments, and likes.

2.2 System Interfaces

The database will:

Interact with the front-end server via an API layer.

Communicate with storage servers for media uploads (e.g., Amazon S3).

Integrate with authentication systems for user verification.

2.3 Constraints, Assumptions, and Dependencies

Constraints:

The system must be GDPR-compliant, ensuring data privacy for users.

It must support high scalability for future growth.

Availability should be 24/7 with minimal downtime.

Assumptions:

Users will access the platform via various devices, and the system should be responsive.

High traffic loads during peak hours are expected and must be managed efficiently.

Dependencies:

The database is dependent on cloud storage solutions for media storage.

It relies on the authentication service for secure user access.

2.4 User Characteristics: Types of Users

Registered Users: Can create posts, comment, like, follow other users, and use hashtags.

Moderators: Can monitor posts and manage inappropriate content.

Administrators: Have access to all data, can manage the user base, and handle any back-end database tasks.

3. System Features and Requirements

3.1 Functional Key

The following functionalities must be implemented in the database system:

1. User Management:

CRUD operations for user profiles.

Login and authentication data management.

2. Content Management:

CRUD operations for posts, photos, and videos.

Tracking and categorizing posts with hashtags.

3. Engagement Management:

Managing likes and comments on posts.

Enabling user follow/unfollow functionality.

4. Notification System:

Storing notifications triggered by user actions like new followers, likes, and comments.

3.2 Use Cases

1. UC1 - User Registration: Allows new users to register, adding a new record to the USER table.

2. UC2 - User Login: Authenticates users, storing login details in the LOGIN table.

3. UC3 - Creating a Post: Users can create posts with captions, photos, or videos.

4. UC4 - Following/Unfollowing: Users can follow or unfollow other users, updating the FOLLOW table.

5. UC5 - Liking and Commenting: Users can like and comment on posts, adding records to the COMMENT_LIKES and COMMENTS tables.

6. UC6 - Hashtagging: Posts can include hashtags, creating relationships in the POST_TAGS and HASHTAGS tables.

7. UC7 - Bookmarking: Users can bookmark posts, which are saved in the BOOKMARKS table.

3.3 External Interface Key

The database will interact with:

API Layer: For handling requests from the front end.

Authentication Service: For managing secure access to user accounts.

Storage Service: For saving and retrieving photos and videos.

3.4 Logical Database Requirements - Storage

The database will require tables for:

User Information: USER, LOGIN, FOLLOW, BOOKMARKS.

Content: POST, PHOTOS, VIDEOS.

Interactions: COMMENTS, COMMENT_LIKES, POST_TAGS, HASHTAGS.

Relationships: FOLLOW, POST_TAGS.

The database should be optimized for quick read/write operations due to the expected high frequency of user interactions.

3.5 Non-Functional Requirements

3.5.1 Safety

Data should be backed up daily to prevent data loss.

A disaster recovery plan must be in place to handle unforeseen failures.

3.5.2 Security

User Data Protection: Implement role-based access control to restrict access to sensitive data.

Encryption: All personal data, including login credentials, should be encrypted both in transit and at rest.

Authorization: Use OAuth or similar standards to control user access.

3.5.3 Availability

The system must be available 99.9% of the time, with minimal downtime.

Load balancing techniques should be implemented to handle high traffic during peak usage times.

This SRS aims to guide the design and development of a highly available, secure, and efficient database system to support a social media platform.