# Software Requirements Specification (SRS)

**Project Name**: El Downtown

**Prepared By**: Ahmed Abdelsalam – Mohamed Taha – Riham Fouad – Ali Nazeer – Abdelaleem Baher

**Date**: 11/9/2024

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to outline the functional and non-functional requirements for **El****Downtown**, a social networking platform designed to enable users to share updates, connect with others, and receive real-time notifications. This platform aims to enhance user interaction with a clean interface and intuitive user experience. This document will serve as a guide for developers, testers, and stakeholders involved in the project.

### 1.2 Scope

The **El Downtown** system is a social networking platform designed to enable users to share updates, connect with others, and receive real-time notifications. The system aims to enhance user interaction with an intuitive user interface and a minimalistic design that prioritizes meaningful engagement The system will provide core functionality for:

* **User Registration and Authentication**: Allows users to create accounts using email or third-party login (Google, Facebook) and manage secure login credentials.
* **Content Sharing and Interaction**: Enables users to create, edit, delete, and interact (Upvote/Downvote, comment) with posts containing text, images, or videos.
* **User Profile Management and Social Features**: Allows users to customize profiles, connect with others, and manage posts and interactions on their feed.

### 1.3 Definitions, Acronyms, and Abbreviations

* **API**: Application Programming Interface
* **UAT**: User Acceptance Testing
* **CI/CD**: Continuous Integration and Continuous Deployment, practices in software development and deployment
* **SQL**: Structured Query Language, used for managing relational databases
* **OAuth**: Open Authorization, a protocol used for secure third-party authentication
* **WCAG**: Web Content Accessibility Guidelines, standards for web accessibility

### 1.4 References

* <https://github.com/formercornet/Eldowntown>
* <https://el-downtown-hmgjq1b.gamma.site/>

## 2. System Overview

### 2.1 Product Perspective

**El Downtown** is a standalone system designed as a social networking platform. It interacts with external services for user authentication and notifications, and directly interfaces with users through a web-based application and mobile app application.

* **Frontend**: A responsive user interface built with React Native for cross platform, allowing users to interact with the platform across devices.
* **Backend**: Server-side logic developed with Flask to manage user requests, process business logic, and handle API interactions.
* **Database**: MySQL is used for persistent data storage, including user profiles, posts, comments, and friend connections and other options.

### 2.2 Product Functions

The key functions of the system include:

1. **User Account Management:** This function allows users to register and log in through email and password or third-party accounts. It also provides password recovery options for users who forget. User roles (admin and user) are assigned at the time of registration or by an admin.
2. **Social Interaction and Content Sharing:** Users can create, edit, delete, and share posts containing text, images, or videos. They can interact with others’ content by Upvoting/ Downvoting posts, commenting on them, and viewing other users’ profiles. Admins can monitor and manage user-generated content to ensure compliance with community standards.
3. **Profile Customization**: Users can manage their profiles by updating their name, bio, and profile picture. This function allows users to personalize their presence on the platform, enabling them to share more about themselves and engage with others in a more meaningful way.

### 2.3 User Classes and Characteristics

Different types of users who will interact with the system include:

* **Admins**: Have full access to all system functions, including managing users and content and configuring settings.
* **Users**: Regular users can perform standard tasks, such as creating, editing, and deleting posts, Upvoting/Downvoting posts, commenting on posts, and managing their profile information. They can also connect with other users and interact with the platform’s social features.

### 2.4 Operating Environment

The system will operate in the following environments:

* **Client Side**: Runs on web browsers like Chrome, Firefox, and Safari and as a mobile application for IOS and ANDROID.

**Server Side:** The backend will be hosted on a cloud server (e.g., AWS, Google Cloud) running a windows operating system. It will use Flask for the server-side logic, with a supporting stack that includes Python and node js for development, and GitHub for continuous integration and deployment (CI/CD).

* **Database**: A MySQL database will be used to store persistent data, including user profiles, posts, comments, and social interactions. This database will be optimized for handling relational data and ensuring efficient data retrieval.

## 3. Functional Requirements

### 3.1 Use Case Diagrams / User Stories

* **Use Case 1**: User Registration
  + **Description**: New users can register using email and password or through third-party accounts (e.g., Google, Facebook)
  + **Actors**: User
  + **Preconditions**: User is on the registration page.
  + **Postconditions**: User’s account is created, and they are redirected to their profile page.
  + **Steps**:
    1. User selects “Sign Up” .
    2. User enters email and password or selects a third-partyauthentication
    3. System validates information and creates the account.
    4. Users are redirected to their profile page.
* **Use Case 2**: Create post
  + **Description**: User can create new posts with text, images, or vidoes.
  + **Actors**: User
  + **Preconditions**: User is logged in and on their home page or profile page.
  + **Postconditions:** The post is saved and displayed on the user’s feed and, on friends' feeds.
  + **Steps**:

1. User selects the “Create Post” button.
2. User enters post content (text, image, or video).
3. System saves the post and displays it on the user’s feed.

### 3.2 Feature Requirements

#### Feature 1: User Registration

* **Description**: Allows new users to register with email and password or through third-party accounts.
* **Inputs**: Email, password, Username, third-party account information (if applicable).
* **Outputs**: Confirmation of successful registration and account creation.
* **Error Handling**: If the email is already registered, an error message is displayed. If third-party authentication fails, an error message is shown.

**Feature 2: User Login**

* **Description**: Enables users to log in with email or Username and password or third-party accounts.
* **Inputs**: Email or Username, password, or third-party credentials.
* **Outputs**: User is logged into the application and redirected to their home page.
* **Error Handling**: Invalid credentials error message.

**Feature 3: Password Recovery**

* **Description**: Allows users to reset their passwords through an email based “Forgot Password” feature.
* **Inputs**: Registered email address.
* **Outputs**: Password reset link sent to the user’s email.
* **Error Handling**: If the email is not registered, an error message is shown.

**Feature 4: User Roles**

* **Description**: The system supports two user roles with different permissions: regular users and admins.
* **Inputs**: User login credentials.
* **Outputs**: Access control based on user role.
* **Error Handling**: Unauthorized actions show an error message.

**Feature 5: Profile Management**

* **Description**: Users can update their profile information, such as name, bio, and profile picture.
* **Inputs**: Profile details (name, bio, profile picture).
* **Outputs**: Updated profile information displayed on the user’s profile page.
* **Error Handling**: If the update fails, an error message is shown.

**Feature 6: Create Post**

* **Description**: Allows users to create posts containing text, images, or videos.
* **Inputs**: Text, image, or video for the post.
* **Outputs**: Post appears in the user’s feed and is shared with friends.
* **Error Handling**: If post creation fails, an error message is displayed.

**Feature 7: Edit Post**

* **Description**: Users can edit their posts.
* **Inputs**: Edited content (text, image, or video).
* **Outputs**: Updated post content displayed on the user’s and friends' feeds.
* **Error Handling**: If the edit fails, an error message is shown.

**Feature 8: Delete Post**

* **Description**: Allows users to delete posts they have created.
* **Inputs**: User selects a post to delete.
* **Outputs**: Post is removed from the user’s and friends' feeds.
* **Error Handling**: If deletion fails, an error message is displayed.

**Feature 9: Upvote/Downvote Post**

* **Description**: Users can Upvote or Downvoteposts from others.
* **Inputs**: User action to Upvote or Downvotea post.
* **Outputs**: Updated Upvote/Downvotecount on the post.
* **Error Handling**: If the action fails, an error message is displayed.

**Feature 10: Comment on Post**

* **Description**: Users can comment on others’ posts.
* **Inputs**: Comment text.
* **Outputs**: Comment is displayed under the post.
* **Error Handling**: If the comment fails to post, an error message is shown.

## 4. Non-Functional Requirements

### Performance Requirements

* The system should respond to user requests within 2 seconds under normal conditions.
* It must support up to 10,000 concurrent users without significant degradation in performance.

### 4.2 Security Requirements

* User data must be encrypted in transit and at rest.
* Only authorized users should have access to specific functionalities, with role-based access control

### 4.3 Usability Requirements

* The user interface should be intuitive, with clear navigation and user guidance.
* Accessibility features should be included to support users with disabilities, following WCAG 2.1 standard and postman to test functionalities.

### 4.4 Reliability and Availability Requirements

* The system must have 99.9% uptime to ensure continuous availability.
* In case of failure, the system must recover within 5 minutes .

### 4.5 Scalability

* The system should be able to scale to support 1 million users or 1 TB data.

### 4.6 Compatibility

* The software should work on modern web browsers, including Chrome, Firefox, Edge, and Safari.
* The system should be compatible with mobile devices for IOS and ANDROID.

## 5. System Models

### 5.1 Use Case Diagrams

Create post use case diagram:

### 

User Registration use case diagram:

A diagram of a diagram

Description automatically generated

## 6. External Interface Requirements

### 6.1 User Interfaces

* The system should have an intuitive UI with the following major components:
  1. **Settings Page**: Allows users to manage account-specific settings, such as:
     + 1. Privacy preferences and blocked users.
       2. Notification settings and email preferences.
       3. Account security, including password management.
  2. **Explore Page**: Enables users to discover new content, trending posts, and suggested users.
  3. **Notifications Center**: Dedicated section for viewing all notifications, including likes, comments, friend requests, and updates.
  4. **Home Screen**: Displays a summary of user activity.
  5. **Chat**: Provides a messaging interface for communication.

### 6.2 API Interfaces

The system will provide APIs for:

* **User Management**: APIs for user registration, login, profile updates, and password recovery.
* **Post Management**: APIs for creating, editing, deleting, and retrieving posts (including text, images, and videos).
* **Social Interactions**: APIs for Upvoting, Downvoting, and commenting on posts, as well as managing friend requests and connections.
* **Notifications**: APIs to manage and deliver real-time notifications for interactions such as votes, comments, and new connections.
* **Messaging**: APIs for one-on-one and group messaging, including sending text, images, and multimedia.

### 6.3 Hardware Interfaces

* The system is designed primarily for web and mobile platforms and does not require interaction with external hardware.

## 7. Other Requirements

### 7.1 Legal and Regulatory Requirements

* The system must comply with data privacy and security regulations such as the General Data Protection Regulation (GDPR).

### 7.2 Documentation Requirements

### API documentation must be provided for developers, detailing endpoints and authentication processes.

### 7.3 Data Backup Requirements

* The system should perform data backups daily.
* Backup data should be stored for a period of 30 days to ensure data availability and support restoration in case of system failure.

## 8. Conclusion

This SRS document outlines the necessary functional and non-functional requirements for **El Downtown**, a social networking platform designed to facilitate user interaction, content sharing, and community building. By adhering to these specifications, the development team will create a system that meets the needs of all stakeholders, providing a secure, scalable, and user-friendly and healthy environment. This document will serve as a foundational guide, ensuring that development aligns with project goals.