Assignment 1 - Mock Stack Overflow Requirements Document

Authors: John Jacoby, Yue Wen Peter Li
Course: CS500 - Foundations of Software Engineering
Term: Spring 2024
Professor: Joydeep Mitra

Requirements Specification

1. View Posts

- **Function**: Display Post
- **Description**: User clicks on a question title hyperlink. Database then retrieves corresponding post and display on the web page from the database.
- **Inputs**: URL and User ID.
- **Source**: User input through the interface, Post data stored in the site's database.
- **Outputs**: Contents of the Displayed Post
- **Destination**: User's web page.
- **Action**: When a user clicks the hyperlink, the system fetches post from the database. The posts are then formatted and displayed on the user's web page. The post may include questions, answers, and relevant metadata like author, date, vote count, and tags.
- **Requirements**: Database access to retrieve post data. Properly formatted post data including metadata. Question titles and hyperlinks need to be presented on the homepage or in search results.
- **Pre-condition**: Database contains the post that can be retrieved and displayed, and the web page hyperlink displayed on the homepage or in search results.
- **Post-condition**: User is presented with the post.
- **Side effects**: None.

2. Create New Posts

- **Function**: Post Submission
- **Description**: Allows users to submit new posts to the site. The function handles the collection of post content, tags, and submission to the database for storage.
- **Inputs**: Post content, selected tags, user ID, datetime.
- **Source**: User input through the site interface, datetime service.
- **Outputs**: NewPost – the newly created post that is submitted to the database.
- **Destination**: Site's database for storing posts and the webpages for the posts.
- **Action**: When a user submits a new post, the system associates it with the user’s ID, captures the datetime of the post, and stores it in the database. The post is then made available for viewing by other users.
- **Requirements**: Text editor for post composition and user authentication to associate posts with user accounts.
- **Pre-condition**: User must be authenticated and authorized to create a post. Database does not contain the post.

- **Post-condition**: A new post is added to the database and becomes visible on the site, i.e. will show up in response to the 'search' and 'view' requirement functions.
- **Side effects**: Too many posts at the same time may overwhelm the system

3. Search for Existing Posts

- **Function**: Post Search
- **Description**: Enables users to search for existing posts based on keywords, tags, author names, and/or datetime ranges. This function uses a search algorithm to retrieve and display a list of posts that match the search terms, as well as sort the returned list.
- **Inputs**: Search query (keywords, tags, author names, datetimes), optional sorting parameters (alphabetical by username, datetime, number of up-votes).
- **Source**: User input through the site's search interface.
- **Outputs**: SearchResults – a list of posts that match the search criteria.
- **Destination**: User's device screen.
- **Action**: The system processes the search query, queries the database, and retrieves posts that match the criteria. The results are then formatted and displayed to the user.
- **Requirements**: A search algorithm, access to the post database, filtering and sorting mechanisms.
- **Pre-condition**: The database contains indexed posts that can be searched.
- **Post-condition**: The user is presented with a list of posts that match the search criteria.
- **Side effects**: Too many requests at the same time may overwhelm the system. Web crawler may steal the information from the database using the method.

4. Commenting on Posts

- **Function**: Post Comment
- **Description**: Allows users to add comments to posts. This function handles the collection, validation, and submission of user comments, as well as their display under the corresponding post.
- **Inputs**: Comment text, user ID, post ID (the ID of the post being commented on).
- **Source**: User input through the site's interface.
- **Outputs**: NewComment – the user's comment, which is added to the site under the specified post.
- **Destination**: Database entry and the webpage associated with the specific post.
- **Action**: The system captures the user's comment, associates it with the user’s ID and the relevant post ID, then stores it in the database. The comment is then displayed under the post for other users to see.
- **Requirements**: Text input field for comments, user authentication to associate comments with user accounts. The question is not locked.
- **Pre-condition**: User must be authenticated. The post on which the comment is to be made must exist.
- **Post-condition**: The comment added to the post and is visible to other users.
- **Side effects**: Too many comments may make it difficult for users to read the entire comment thread.

5. Voting on Posts

- **Function**: Post Voting
- **Description**: Enables users to vote on posts to indicate their quality or relevance. This function allows for up-voting or down-voting of posts and handles the updating of vote counts associated with each post.
- **Inputs**: Vote (up-vote or down-vote), user ID, post ID (the ID of the post being voted on).
- **Source**: User input through the site's interface.

- **Outputs**: UpdatedVoteCount – the new vote count for the post after the user's vote has been applied.
- **Destination**: Database entry and webpage associated with the specific post.
- **Action**: The system registers the user's vote, updates the post's total vote count in the database, and reflects this change on the user interface. The vote count influences the visibility and ranking of posts in search and view functions.
- **Requirements**: Voting mechanism on the user interface, user authentication to ensure legitimate voting.
- **Pre-condition**: User must be authenticated. The post on which the vote is to be made must exist.
- **Post-condition**: The vote count of the post is updated, potentially affecting its visibility and ranking on the site.
- **Side effects**: Malicious use may lead to an unfair ranking system.

6. Tagging Posts

- **Function**: Post Tagging
- **Description**: Allows users to add tags to their posts to categorize and facilitate easier searching and sorting of posts. This function enables the assignment of multiple tags per post, and the management of these tags, including adding new tags or selecting from existing ones.
- **Inputs**: Selected tags, user ID, post ID (the ID of the post being tagged).
- **Source**: User input through the site's interface.
- **Outputs**: NewPost – the post with the newly assigned tags.
- **Destination**: Database entry and webpage associated with the specific post.
- **Action**: The system allows the user to select or create tags when creating a post. These tags are then associated with the post in the database, aiding in the organization and retrieval of posts via the search and view functions.
- **Requirements**: Interface for selecting or creating tags, any requirement necessary for the 'Create New Post' functionality, since this is built into that.
- **Pre-condition**: Currently in the post create workflow.
- **Post-condition**: The post is created with the selected tags, enhancing searchability and categorization.
- **Side effects**: None.

7. User Profiles

- **Function**: Profile Management
- **Description**: Allows users to create, view, edit, and delete their own profiles on the site. This function manages user information, including usernames, account creation date, and profile pictures. It also displays user activity such as their posted content and voting history.
- **Inputs**: User information (username, profile picture, etc.).
- **Source**: User input through the site's interface. Database.
- **Outputs**: UserProfile – the user's profile with their information and activity summary.
- **Destination**: User's profile webpage and database for storing profile data.
- **Action**: The system enables users to input and update their profile information, which is then stored in the database. The profile page also aggregates and displays the user's activities on the site, including their posts and voting history.
- **Requirements**: Interface for entering and editing profile information, database storage for user data, mechanisms to display user activity, authentication mechanisms for logging in.

- **Pre-condition**: User must be registered and authenticated to edit an account, or logged out to create a new account. Account must not already exist.
- **Post-condition**: User's profile is created or updated with the provided information, which is then visible to the user and others.
- **Side effects**: None.

8. Post Moderation

- **Function**: Moderation System
- **Description**: Provides a system for moderators (selected users) to review and manage posts to ensure they adhere to the site's guidelines. This includes editing, deleting, flagging, or locking/unlocking posts that are inappropriate, off-topic, or violate community standards.
- **Inputs**: Post ID, moderator actions (edit, delete, flag, lock/unlock), moderator ID.
- **Source**: Moderator input through the site's interface.
- **Outputs**: Post – the post after undergoing the moderation action.
- **Destination**: Database entry associated with the specific post and moderation logs.
- **Action**: The system allows moderators to review posts and perform necessary actions such as editing content, removing posts, flagging or locking them for further review. Locked posts can also be unlocked.
- **Requirements**: Moderation tools interface, user roles with different permissions (e.g., regular user, moderator).
- **Pre-condition**: Moderator must be authenticated and authorized to perform moderation tasks.
- **Post-condition**: The post is edited, deleted, or flagged as per the moderator's action, influencing its visibility and status on the site.
- **Side effects**: Some moderators may abuse their privileges and potentially harm the community.

9. Respond to question posts

- **Function**: Response post Submission
- **Description**: Allows users to submit a response to a question post. The function handles the collection of response content and submission to the database for storage.
- **Inputs**: Response content, original question post ID, user ID, datetime.
- **Source**: User input through the site interface, datetime service.
- **Outputs**: NewPost – the newly created post that is submitted to the database.
- **Destination**: Site's database for storing posts and and the webpages for the posts.
- **Action**: When a user submits a response post, the system associates it with the user’s ID, captures the original question post id and datetime of the response post, and stores it in the database. The response post is then made available for viewing by other users. This function uses a sorting algorithm to order the most up-voted response posts at the top.
- **Requirements**: Text editor for post composition and user authentication to associate posts with user accounts. Sorting algorithm.
- **Pre-condition**: User must be authenticated and authorized to respond to a post. Database does not contain the response post.
- **Post-condition**: A new response post is added to the database and becomes visible on the site.
- **Side effects**: Too many responses may make it difficult for users to read the entire post thread.

10. Viewing other users’ profiles

- **Function**: Visiting other users' Profile
- **Description**: Allows users to view other users' profiles on the site. This function is different from the "user profiles" function. The user visiting the other user is called "visiting user", and the user being visited is called "visited user".
- **Inputs**: Visiting user ID, visited user ID.
- **Source**: User input through the site's interface. User profile data from the database.
- **Outputs**: UserProfile – the visited user's profile with their information and activity summary.
- **Destination**: Visiting user’s screen.
- **Action**: The visiting function allows users to access other users' profiles in view-only mode by clicking on the visited user's username hyperlink. The profile page also aggregates and displays the user's activities on the site, including their posts and voting history.
- **Requirements**: Interface for accessing other users' profiles via hyperlink. Database storage for user data, mechanisms to display user activity, authentication mechanisms for logging in.
- **Pre-condition**: Visiting user must be registered. Visited user must exist in the database.
- **Post-condition**: Visited user's profile is displayed on the webpage.
- **Side effects**: None.

Requirements Validation

1. View Posts

- The user visits a question post by clicking the hyperlink of the post => The content of the post displays on the webpage.
- User enters a deleted post's URL=> Webpage prompts error message: ”This post does not exist.”

2. Create New Posts

- The user writes and submits a new post => The post is added to the database, where it is available for access i.e. by the Serach Posts and View Posts functions
- The user writes and submits a post with the same title as the previous post => Webpage prompts error message: ”Post title must be unique”
- The user submits a empty title or content post => Webpage prompts error message: ”Post title and content must be not empty”

3. Search for Existing Posts

- The user makes a search for which there are no matching results in the database => The user gets a message explaining that there are no results
- The user makes a search for which there are matching results in the database => The users sees those search results
- The user makes a search with empty string => Nothing happen on the webpage.

4. Commenting on Posts

- The user adds a comment => The comment is added to the database and associated with the post the comment was made on.
- The user adds a comment which exceeds the character limit(assuming 400 characters) => The comment is added to the database and associated with the post the comment was made on. but only display first

100 characters, and the rest with ellipsis.

5. Voting on Posts

- The user votes a post 'up' => The post's vote value is incremented in the database
- The user votes a post 'down' => The post's vote value is decremented in the database

6. Tagging Posts

- The user adds a tag to a post during the post creation process => If/when the post is actually submitted, it is added to the database with an association to the tag(s).
- The tags of the post exceed the character limitation (assuming 100 characters) => Webpage prompts error message: "Maximum 100 characters."

7. User Profiles

- The user creates a profile => The profile is added to the database.
- The user deletes their profile => The profile is removed from the database.
- The user edits their profile => The profile is updated in the database to reflect the edits.
- The user goes to their profile page => Profile data is pulled from the database and displayed on the screen.

8. Post Moderation

- A moderator deletes a post => The post is removed from the database and therefore not visible to anyone anymore.
- A moderator edits a post => The post content is changed in the database
- A moderator flags a post => The flag is represented with a value in the database, and users viewing the post see a notification of the flag and the moderator's explanation of the flag.
- A moderator locks a post => The lock is represented with a boolean false value in the database, and users viewing the post see a lock. The voting-up, voting-down, comment and respond functions are disabled.
- A moderator unlock a post => The lock is represented with a boolean true value in the database, and users viewing the post see it as usual with no special decorations. The voting-up, voting-down, comment and respond functions are enabled as usual.

9. Respond to question posts

- The user writes and submits a response post => The response post is added along with the origin question post to the database, where it is available for access i.e. by the Search Posts and View Posts functions
- The user submits a empty content response post => Webpage prompts error message: "Post content must be not empty"

10. Viewing other users profiles

- The visiting user goes to another user's profile page => Profile data is pulled from the database and displayed on the screen.

Simplifying Assumptions

1. We are only supporting English
2. Comments appear automatically in reverse chronological order under a post with no sorting or filtering options.
3. Post titles should be unique, even though in the real world we could have posts with the same titles be differentiated by datetime and username.
4. Only posts can be voted on - not comments