# Mock StackOverflow

Marko Krstulovic and Thean Cheat Lim

# Final List of requirements

## Definitions of Done

- Sanitize all user inputs whenever an API call is made, removing the opportunity for XSS attacks and SQL injections.
- All API changes are validated to check if the users sending these requests meet the correct reputation requirements for the API call. Reputation can't be manually edited by a user.

## NEW REQUIREMENT: User Validation/Authentication

1. Function Description: Validate user identity throughout application.
2. Inputs and Origins of the Inputs: Request to view page that requires logged in user (Post Creation, Profile viewing for own profile).
3. Outputs and Destinations: Session token created and maintained by the server, sent in cookies to a validated user, returns page attempting to be viewed or login page.
4. Requires:
    - Database CRUD (Create, Read, Update, Delete) operations for Users.
    - Session management library.
    - CSRF Token management library.
5. Actions:
    - Validate user-provided inputs for username and password upon first login.
    - Employ 2FA to validate CSRF Tokens.
    - Return a new session token if this is the initial log in.
    - If a user is already logged in, check that they possess a valid session token. If they do, accept the request.
    - If a user is already logged in and they lack a valid session token, reject the request.
6. Pre-conditions:
    - Email address and password must be in valid format, and must be populated.
    - Session token must be present if already logged in.
7. Post-conditions:
    - Session token is stored in cookie on user browser.
    - Session token is marked as http-only and same-site restricted.

    - User session updated with new session token if first login. Otherwise, session token maintained and not lost.
8. Side effects:
    - Storage may run out due to saving information within the database.
    - Session being lost or failing to send a session token could become a problem for login and user experience.

## View Posts

1. Function Description: Allow users to view existing posts, including metadata (i.e., vote count) on the platform.
2. Inputs and Origins of the Inputs:
    - Post IDs listed from Posts database. Use case: Display all the posts for the requested page.
    - Post ID coming from user interaction (e.g., user clicking on a link to view a post)
3. Outputs and Destinations: Rendered posts on the user interfaces.
4. Requires:
    - Database CRUD (Create, Read, Update, Delete) operations for Posts.
5. Actions:
    - Retrieves the requested thread of posts from the database.
    - If it was for viewing a specific question post, retrieve all the relevant answer posts. For answer posts, sort them descendingly by votes count.
    - Displays the retrieved posts from the database on the user interface.
6. Pre-condition: At least one post is available in the database.
7. Post-condition:
    - Posts are successfully displayed.
    - Posts view count increased (for the case of viewing a specific question post)
8. Side effects: None

## Create New Posts

1. Function Description: Allow users to compose and publish new posts
2. Inputs and Origins of the Inputs:
    - Post content and metadata (i.e, timestamp, optional post tags). These are provided by users or inferred through an interface.
    - A flag stating if a Post is a Question or an Answer post. This can be inferred from the user interaction/ user interface.
    - User ID from current session manager.
3. Outputs and Destinations: Published posts which are visible to the platform. The post content, optional tags, and metadata are also stored in a database.
4. Requires:
    - User authentication to ensure only authorized users can create posts.
    - Database CRUD (Create, Read, Update, Delete) operations for Posts.
    - Validation algorithm for post content.
5. Actions:

- Validate post content to ensure it meets specified criteria (i.e., A Question post must include a title and a description with more than 20 characters; No limitation for an Answer post). Don't allow post creation otherwise.
- Generate a post ID.
- If it was an answer post, link it back to the question post.
- Stores post content and its metadata (such as tags, author's user ID, timestamp) in a database

6. Pre-condition:
    - Users are logged in to be authorized to create posts.
    - User input for title, description, or answer is not empty.
7. Post-condition: The new post is stored in the database and is accessible through the platform.
8. Side effects:
    - Post database running out of storage.
    - Privacy/moderation issues from allowing text inputs (e.g. SQL injection, inappropriate language, etc.)..

## Search for Existing Posts

1. Function Description: Allow users to search for specific posts or topics
2. Inputs and Origins of the Inputs: Search query from users input via the platform interface
3. Outputs and Destinations: List of posts matching search criteria displayed on the User Interface.
4. Requires:
    - Database CRUD (Create, Read, Update, Delete) operations for Posts.
    - Search library/algorithm to support searching for posts.
5. Actions:
    - Query posts from the database that match the search criteria (Title and post tags).
    - Display the list of matching posts to users. It is possible to have no matching posts.
6. Pre-condition: At least one post is available in the database.
7. Post-condition: Search results matching the query is returned
8. Side effects:
    - Exceptions from search library/functionality used.

## Commenting on Posts

1. Function Description: Allow users to add comments to existing posts.
2. Inputs and Origins of the Inputs:
    - Post ID from current post being commented.
    - User ID from current session manager.
    - Input comment text from users via the platform interface
3. Outputs and Destinations: New comment on the post is displayed on the User Interface.
4. Requires:

- Database CRUD (Create, Read, Update, Delete) operations for Comments.
- User authentication so only authorized users can comment on posts.
- Reputation Score System in place to determine eligibility to comment.

5. Actions:
    - Associate the comment with the respective post.
    - Store the new comment in the database with relevant metadata (author, timestamp).
6. Pre-condition:
    - At least one post is available in the database to be commented on.
    - Users are logged in.
    - Users have enough reputation points (i.e, 50 reputation) to comment.
    - Comment text is not empty.
7. Post-condition: The new comment is stored in the database and is added/associated to the post.
8. Side effects: Comment database running out of storage.

## Voting on Posts

1. Function Description: Increases or decreases the score of a question or answer depending on user voting.
2. Inputs and Origins of the Inputs:
    - User ID from current session manager.
    - User ID of post owner.
    - Post ID from current webpage.
    - Upvote/Downvote event from user mouse click.
3. Outputs and Destinations:
    - New score and chosen vote displayed on user browser.
4. Requires:
    - User authentication system and session manager.
    - Reputation Score System in place to determine eligibility to vote.
5. Actions:
    - For Questions and Answers:
        i. If the user has not voted and upvotes, then the post score increases by 1 and the post's owner reputation increases by 10.
        ii. If the user has not voted and downvotes, the post score decreases by 1, the post's owner reputation decreases by 2, and the user's reputation decreases by 1.
        iii. If the user has voted and clicks the same vote option as previously selected, reverse that option (score and reputation changes).
        iv. If the user has voted and clicks a different vote option, reverse the option they originally selected, and then perform the new option.
    - For Comments, score functions the same as above, but doesn't change any reputation and you cannot downvote.
    - If the user lacks the required reputation, inform them in a message.
    - Saves this information in a database.

6. Pre-condition:
    - There must exist a post such that one can vote on it (with an existing user associated).
    - Current user must have at least 15 reputation to vote up.
    - Current user must have at least 125 reputation to vote down.
    - Current user is logged in.
    - Post is not closed or locked, and must exist.
7. Post-condition:
    - User reputation is updated and post owner reputation is updated.
    - User reputation is at least 1.
    - Score of the post has been updated, and the vote choice has been displayed.
8. Side effects:
    - Storage may run out due to saving information within the database.
    - If the user is not logged in, it prompts user log in.
    - Users can abuse voting mechanics.

## Tagging Posts

1. Function Description: Association of terms or keywords when creating questions, and suggestion of tags.
2. Inputs and Origins of the Inputs:
    - Post ID from current webpage (question creation).
    - User ID from current session manager.
    - User Input (tag name) from input field (key press).
3. Outputs and Destinations:
    - Tag appears under user question when question page is displayed.
4. Requires:
    - User authentication system and session manager.
    - Database CRUD (Create, Read, Update, Delete) operations
    - Reputation Score System
5. Actions:
    - For every tag entered by the user, associate existing tags to the Question.
    - If the tag entered didn't exist, create a new tag if the current user has over 1500 reputation.
    - If a user enters more than 5 tags or tries to create a tag without 1500 reputation, display an error.
6. Pre-condition:
    - Current user is logged in.
    - New post must be in progress.
    - At least one tag must exist.
7. Post-condition:
    - Post is associated with the selected tag.
    - If a new tag is selected, the tag is created if the user meets the 1500 reputation requirement.
    - At most 5 tags have been associated with the post.

- All of the tags associated exist.
- Information has been recorded in the database.

8. Side effects:
    - Storage may run out due to saving information within the database.
    - Tags will not be created if over 35 characters or reputation requirements are not met.
    - Users unable to create tags if they don't exist lack reputation..

## Viewing User Profiles

1. Function Description: Show user information, summary statistics, and views of a user's posts.
2. Inputs and Origins of the Inputs:
    - User ID from current session manager.
    - User ID of profile currently being viewed from webpage.
    - User input (link or tab click) from mouse press events.
3. Outputs and Destinations:
    - Summary page of personal information, summary statistics, and recent posts on user browser.
4. Requires:
    - User authentication system and session manager.
    - Reputation Score system.
    - Database CRUD (Create, Read, Update, Delete) operations
5. Actions
    - Shows summary statistics of user: reputation, number of questions, and number of answered questions.
    - Shows personal information such as display name, "About" section, and recent posts, date joined, and time last seen.
    - Shows the questions that the user has posted and the questions that the user has answered recently.
    - If the user ID matches the viewed profile ID, display the option to edit, the user's full name, and their email address. Hitting the edit profile button takes the user to the Edit User Profiles function.
    - If the user clicks on a specific post, they are taken to the post.
6. Pre-condition:
    - Current user is logged in.
    - The profile being viewed exists.
    - To edit, the current user must be viewing their own profile.
7. Post-condition:
    - User is redirected to the correct location. They will be redirected to view a specific post, or moved to the Edit Profiles function.
8. Side effects: None

## Editing User Profiles

1. Function Description: Show options that a user can edit and update them accordingly.
2. Inputs and Origins of the Inputs:
    - User ID from current session manager.
    - User clicks from mouse press events.
    - User text inputs (display name, email, about text) from key press events.
3. Outputs and Destinations:
    - Edit Profile page with real-time changes in fields on user browser.
    - View User Profile function displayed on user browser if saved/exited.
4. Requires:
    - User authentication system and session manager.
    - Database CRUD (Create, Read, Update, Delete) operations
5. Actions:
    - If the user enters or changes information in the text fields, display these changes.
    - Can edit display name, email, real name, or about section.
    - If the user navigates off of the page, discard the changes.
    - If the user saves their profile, update the profile information and take the user back to view their own profile.
6. Pre-condition:
    - Current user is logged in.
    - The profile being edited exists.
    - Current user is currently viewing/editing their own profile.
    - Text input fields for display name and email are not empty.
7. Post-condition:
    - Profile is updated with all of the changes made by the user if saved.
    - Changes do not persist if not saved.
8. Side effects:
    - Storage may run out due to saving information within the database.

## Post Moderation

1. Function Description: User moderation to open or close question threads and flag for inappropriate posts.
2. Inputs and Origins of the Inputs:
    - User ID from current session manager.
    - Post ID from current question page being viewed.
    - User click (flag or open/close) from mouse press event.
3. Outputs and Destinations:
    - Flags and open/close votes appear on the current post in the user browser.
4. Requires:
    - User authentication system and session manager.
    - Reputation Score system.
    - Database CRUD (Create, Read, Update, Delete) operations
5. Actions:

- If the user presses the flag button, increase the number of flags by 1 if not pressed previously. If previously flagged, decrease the number by 1.
- If the number of flags reach 6, delete the post and reduce its owner's reputation by 100.
- If the user votes to close the question, increase the number of close votes by 1.
- If the user votes to reopen the question, increase the number of reopen votes by 1.
- If the user selects a vote they previously made (close or reopen), remove that vote and decrement the count of the vote by 1.
- If the number of close votes reaches 3, change the question status to closed and allow reopen votes, then delete all close votes.
- When reopen votes reach 3, update the status of the question to be open and show the option to cast close votes again, then delete all reopen votes.
- Information is saved in the database.

6. Pre-condition:
    - Current user is logged in.
    - The post being flagged or voted on exists.
    - Current user must have at least 15 reputation to flag posts.
    - Current user must have at least 3000 reputation to close/open questions.
7. Post-condition:
    - Current post's number of flags is correctly updated if reputation requirements are met.
    - Post deleted if flags reached 6, and post owner's reputation decreased by 100, down to a minimum of 1.
    - Current post's close and open votes updated if reputation requirements are met.
    - Post status is updated based on the number of votes.
    - Close/Reopen votes deleted if question status changes.
8. Side effects:
    - Storage may run out due to saving information within the database.
    - Massive moderation abuse could be a problem if functionality is misused.

## Account Creation/Sign In

1. Function Description: Creates a new user account or updates session to use an existing one.
2. Inputs and Origins of the Inputs:
    - User click (button or image upload click) events for mouse presses.
    - User input (email, password, display name) from key presses.
3. Outputs and Destinations:
    - Signed in home page displayed on user browser.
4. Requires:
    - User authentication system and session manager.
5. Actions:

    - If the user selects the sign-up option, they are prompted to enter display name, email, full name, and a password. If they select to create an account, then save the profile, update the current user session, and redirect them to the home page.
    - If the user selects the sign-in option, they are prompted to enter email and password. If they select to then sign in:
        - If the email and password authentication passes, then update the session and redirect the user to the home page.
        - If the authentication fails, alert the user of this and prompt them to re enter their credentials.
    - Information is saved in the database.
6. Pre-condition:
    - Current user must not be logged in.
    - Email address, password, full name, and display name inputs must be populated.
    - Email address and password must be in valid format.
7. Post-condition:
    - New user account created if using new account, preset to 1 reputation.
    - User session updated with correct User ID.
    - User is sent to the home page if a new account is created or if they successfully log in.
    - User is alerted that their email or password is incorrect if sign-in fails.
8. Side effects:
    - Storage may run out due to saving information within the database.
    - Privacy problems could arise if credentials are leaked or accessible.

## Mark Post as Answer

1. Function Description: Allow users to mark an answer post as the solution to a specific question
2. Inputs and Origins of the Inputs:
    - Question Post ID coming from the user interface
    - Answer Post ID coming from the user interface
    - User ID (Question Post Owner ID) from the session
3. Outputs and Destinations:
    - Updated post status in the database, indicating a solution has been chosen
    - Visual indication on the user interface that an answer has been marked as the solution (e.g., a green checkmark).
4. Requires:
    - Database CRUD (Create, Read, Update, Delete) operations for Posts.
    - User authentication system and session manager.
5. Actions:
    - Update database: Set the "solution_id" field of the corresponding question post to the provided answer post ID.
    - Display the chosen answer with a visual marker indicating it's the solution.
6. Pre-condition:
    - A question with at least one answer exists.

- User must be logged in.
- The user must be authorized (ex, question post owner) to mark the solution.

7. Post-condition:
    - The chosen answer is marked as the solution in the database.
    - The user interface reflects the solution status.
8. Side effects:
    - Users might mark inappropriate posts as solutions.