IT-314: Software Engineering

USE CASES

Group -14

Social Media Platform

GROUP MEMBERS:

202201190	PATEL BANSI VINODBHAI
202201233	BARAIYA HARSH MUKESHBHAI
202201206	PATOLIYA VATSAL SHANTILAL
202201222	PANDAR HIMANSHU LAXMANBHAI
202201236	AMIN FENIL SURESHKUMAR
202201200	PAREKH PARTH NIRAVBHAI
202201223	KANSODARIYA KAVAN CHIMANBHAI
202201244	SUDANI MITUL RASIKBHAI
202201194	SARANG KENIL ARUNBHAI
202201243	BOSAMIYA HARSH RUPALBEN
202201224	PITHADIYA AADITYA PRAVINBHAI

Under the guidance of:

Prof.: Dr. Saurabh Tiwari

Mentor: Sarthak

Use Case: User Registration

❖ **Actors:** Unregistered user.

❖ **Preconditions:**

1. The user has access to the registration page of the platform.
2. The system is online and functional.

❖ **Main flow:**

1. The user navigates to the registration page on the system.
2. The user enters a desired username, email address, and password in the provided form.
3. The user clicks the "Register" button.
4. The system checks that all required fields (username, email, password) are filled.
5. The system validates the entered username:
    a. Ensures the username is less than 15 characters.
    b. Ensures the username does not contain special characters.
    c. Checks that the username is unique and not already registered.
6. The system validates the entered email address:
    a. Ensures the email is in a valid format.
    b. Checks that the email is not already registered.
7. The system validates the entered password:
    a. Ensures the password contains at least one uppercase letter, one lowercase letter, one special character, and one number.

8. If all validations pass, the system sends an OTP to the entered email.

9. The user enters the OTP received via email in the dialogue box.

10. If the user enters the correct OTP then the system displays a confirmation message indicating successful registration.

11. The system redirects the user to the login page, allowing them to access their account.

❖ Alternate Flows:

4.1. If any of the required fields (username, email, password) are empty, the system will display an error message:

- "Please fill in all required fields."

5.1. If the username exceeds 15 characters, the system will display an error message:

- "Username must be less than 15 characters."

5.2. If the username contains special characters, the system will display an error message:

- "Username must not contain special characters."

5.3. If the username is already registered, the system will display an error message:

- "Username is already taken. Please choose a different one."

6.1. If the email is not in a valid format, the system will display an error message:

- "Please enter a valid email address."

6.2. If the email is already registered, the system will display an error message:

- "This email is already registered. Please use a different email or log in using this email."

7.1. If the password does not meet the complexity requirements, the system will display an error message:

- "The Password must contain at least one uppercase letter, one lowercase letter, one special character, and one number.".

8.1. If the system encounters an issue while saving the user data to the database, the system will display an error message:

- "Internal server error."

9.1 If the user enters the wrong OTP then the system displays an error message of “invalid OTP”.

❖ Postconditions:

❖ Success Scenario:
- ➢ A new user account is successfully created and stored in the system's database.
- ➢ The user is notified with a confirmation message of successful registration.
- ➢ The user is redirected to the login page and can use their credentials to log in to the system.

❖ Failure Scenarios:
- ➢ If validation fails (e.g., invalid username, email, or password), the system does not create the account and provides appropriate error messages to the user.
- ➢ If a system error occurs during registration (e.g., database failure), no user data is saved, and the system notifies the user to try again later.

Use Case: User login

❖ **Actors:** Registered user.

❖ **Preconditions:**

1. The user has access to the login page of the platform.
2. The user should have an account registered.
3. The system is online and functional

❖ **Main flow:**

1. The user navigates to the login page on the system.
2. The user enters their registered email address and password in the login form.
3. The user clicks the "Login" button.
4. The system validates the entered email address:
   a. Checks if the email exists in the database.
5. The system verifies the entered password:
   a. Ensures that the password matches the one stored for the provided email.
6. If the credentials are valid, the system logs the user into their account.
7. The system redirects the user to their account dashboard, allowing access to personalized features.

❖ Alternate Flows:

4.1. If any of the required fields (email or password) are empty, the system will display an error message:

- "Enter email and password."

5.1. If the email is not found in the database, the system will display an error message:

- "Invalid email or password."

6.1. If the entered password does not match the stored password for the provided email, the system will display an error message:

- "Incorrect password."

6.2. If the user forgets their password, they can click on the "Forgot Password" button to initiate the password recovery process.

7.1. If the system encounters an issue during the login process (e.g., server error or database unavailability), it will display an error message:

- "Internal server error"

❖ Postconditions:

❖ Success Scenario:
  ➢ The user is successfully logged in and redirected to their dashboard.
  ➢ The system marks the user session as active.

❖ Failure Scenarios:
  ➢ The user is not logged in due to incorrect credentials, missing fields, or an unregistered email.
  ➢ The system provides an appropriate error message and does not log in the user.

Use Case: Edit profile

❖ **Actors:** Registered user, system.

❖ **Preconditions:**

1. The user should have an account registered.
2. The user should be able to access the edit profile page.
3. The system is online and functional.

❖ **Main flow:**

1. The user navigates to the profile editing page.
2. The user selects a new profile photo to upload.
3. The user updates their bio by entering or modifying text in the bio field.
4. The user selects their gender from the available options (e.g., Male, Female).
5. The user clicks the "Save" or "Update" button.
6. The system validates the inputs:
    a. Ensures the profile photo is in a supported file format (e.g., JPEG, PNG).
    b. Ensures the gender selection is from the allowed options.
7. If all inputs are valid, the system updates the user's profile information in the database.

8. The system displays a confirmation message indicating that the profile has been updated successfully.

9. The system redirects the user to their updated profile page.

❖ Alternate Flows:

2.1. If the selected profile photo is in an unsupported format or exceeds the file size limit, the system will display an error message:

- "Unsupported file format. Please upload a JPEG or PNG image."

6.1. If the system encounters a validation error during input checks, it will display an appropriate error message and prevent saving changes until all inputs are valid.

7.1. If the system encounters an issue while saving the updated profile information to the database, it will display an error message: “Internal Server error.”

❖ Postconditions:

❖ Success Scenario:
  - ➢ The user's updated profile photo, bio, and/or gender information is saved in the system.
  - ➢ The changes are reflected on the user's profile and visible where applicable.

❖ Failure Scenarios:
  - ➢ If the user-provided data is invalid (e.g., unsupported file format for profile photo), the system rejects the update and provides an appropriate error message.
  - ➢ If a system error occurs, the profile remains unchanged, and the user is notified to try again later.

Use Case: Add post

❖ **Actors:** Registered user, system.

❖ **Preconditions:**

1. The user should have an account registered.
2. The user should be able to access the Add post interface.
3. The system is online and functional.

❖ **Main flow:**

1. The user navigates to the "Add Post" page or section.
2. The user uploads a photo or video from their device.
3. The user enters a caption in the text field.
4. The user clicks the "Post" button.
5. The system validates the uploaded file:
   a. Ensures the photo/video is in a supported format (e.g., JPEG, PNG, MP4).If all inputs are valid, the system saves the post to the database.
6. The system displays a confirmation message indicating that the post was created successfully.
7. The system updates the user's feed and profile to include the new post.

❖ **Alternate Flows:**

2.1. If the selected profile photo is in an unsupported format or exceeds the file size limit, the system will display an error message:

- "Unsupported file format. Please upload a JPEG or PNG image or mp4 video."

3.1. If the caption exceeds the character limit (e.g., 300 characters), the system will display an error message:

- "Caption must be within 300 characters."

5.1. If the user attempts to post without uploading a photo or video, the system will display an error message:

- "Please upload a photo or video to create a post."

❖ Postconditions:

❖ Success Scenario:
- ➢ The new post with the photo/video and caption is successfully created and stored in the system.
- ➢ The post is visible on the user's profile and/or feed.

❖ Failure Scenarios:
- ➢ If the provided inputs are invalid (e.g., unsupported file type, caption too long), the system does not create the post and provides an appropriate error message.
- ➢ If a system error occurs, the post is not saved, and the user is notified to try again later.

Use Case: Message other users

❖ **Actors:** Registered users, system.

❖ **Preconditions:**

1. The users should have an account registered.
2. The user should be able to access the conversation page.
3. The users must be following each other.
4. The system is online and functional.

❖ **Main flow:**

1. The sender navigates to the messaging section on the platform.
2. The sender selects a recipient from their list of followers who are also following them back.
3. The sender types a message in the input field.
4. The sender clicks the "Send" button.
5. The system checks whether both users follow each other:
   a. Confirms that a mutual following relationship exists.
6. The system validates the message content:
   a. Ensures the message is not empty.
7. If all validations pass, the system stores the message in the database and updates the conversation history.
8. The message appears in the recipient's inbox.

❖ Alternate Flows:

2.1. If the recipient is not following the sender back, the system will not display the sender.

6.1. If the message is empty, the system will not allow the user to send it.

6.2. If the message exceeds the character limit (e.g., 1000 characters), the system will display an error message:

- "The Message cannot exceed 1000 characters."

7.1. If the system encounters an issue while storing the message, it will display an error message:

- "Internal server error."

❖ Postconditions:

❖ Success Scenario:
  - ➢ The message is successfully sent by the sender and delivered to the recipient.
  - ➢ The conversation is updated and stored in the database.

❖ Failure Scenarios:
  - ➢ The system does not send the message due to invalid inputs, blocked relationships, or a system error.
  - ➢ The user is notified of the issue, and the message is not saved.

Use Case: Search user

❖ **Actors:** Registered user, system.

❖ **Preconditions:**

1. The users should have an account registered.
2. The search bar functionality must be available on the platform.

❖ **Main flow:**

1. The user navigates to the search bar on the platform.

2. The user types a username or part of a username into the search bar.

3. The system processes the query and searches for matching usernames in the database.

4. The system displays a list of usernames that match the search query below the search bar.

5. The user clicks on a specific username from the list.

6. The system navigates the user to the selected user's profile page.

❖ **Alternate Flows:**

3.1. If no matching usernames are found, the system will not display any users.

5.1. If the user clicks on a username but the system fails to navigate to the profile, the system will display an error message of unavailable profile.

❖ Postconditions:

- ❖ Success Scenario:
  - ➢ A list of matching usernames is displayed below the search bar based on the search query.
  - ➢ The user successfully navigates to the selected profile.
- ❖ Failure Scenarios:
  - ➢ No matching usernames are found for the search query.
  - ➢ The system encounters an issue retrieving or displaying results, and the user is notified.

Use Case: Like post

- **Actors:** Registered user, system.

- **Preconditions:**

1. The liker must be logged into their account.
2. The post to be liked must exist and be visible to the liker.

- **Main flow:**

1. The liker navigates to a post visible in their feed or the post owner's profile.

2. The liker clicks the "Like" button below the post.

3. The system registers the like:
    a. Increases the like count for the post.
    b. Associates the like with the liker in the database.

4. The system generates a notification for the post owner:
    a. The notification includes the name of the liker and the post that was liked.

5. The notification appears in the post owner's notification section.

6. The like count and status (e.g., highlighted "Like" button) are updated for the liker in real-time.

❖ Alternate Flows:

2.1. If the liker tries to like a post they have already liked, the system will instead remove the like:

- Decreases the like count for the post.
- Removes the like association from the database.

3.1. If the post has been deleted after being displayed to the liker, the system will display an error message:

- "This post is no longer available."

4.1. If the system fails to generate the notification for the post owner due to a server or database error, the liker is not notified, and the post owner's notifications remain unchanged.

5.1. If the notification system is delayed or unavailable, the post owner's notifications will not update in real-time but may be delivered later.

❖ Postconditions:

❖ Success Scenario:
  ➢ The like is successfully added to the post.
  ➢ The post owner's notifications are updated to inform them about the like.

❖ Failure Scenarios:
  ➢ The like is not registered due to a system error, and the user is notified.
  ➢ The post owner does not receive the notification if the notification system fails.

Use Case: Comment on a post

- **Actors:** Registered user, system.

- **Preconditions:**

1. The commenter must be logged into their account.
2. The post to be commented on must exist and be visible to the commenter.

- **Main Flow:**

1. The commenter navigates to a post visible in their feed or the post owner's profile.

2. The commenter types a message into the comment input field below the post.

3. The commenter clicks the "Post Comment" button.

4. The system validates the comment:
    a. Ensures the comment is not empty and does not exceed the maximum character limit.

5. The system registers the comment:
    a. Associates the comment with the commenter and the post in the database.
    b. Increments the comment count for the post.

6. The comment is displayed under the post in real-time, visible to the commenter and other users.

❖ Alternate flows :

2.1. If the comment field is left empty or contains only whitespace, the system will not let the sender send the message.

2.2. If the comment exceeds the maximum character limit, the system will not let the sender comment unless they reduce the comment size.

❖ Postconditions:

❖ Success Scenario:
- ➢ The comment is successfully added to the post.
- ➢ The post owner's notifications are updated to inform them about the comment.

❖ Failure Scenarios:
- ➢ The comment is not registered due to a system error, and the commenter is notified.

Use Case: Follow/unfollow a user

❖ **Actors:** Registered users(follower,followed), system.

❖ **Preconditions :**

1. The follower must be logged into their account.
2. The user to be followed/unfollowed must have an active account.

❖ **Main Flow: Follow a User**

1. The follower navigates to the profile of the user they wish to follow.
2. The follower clicks the "Follow" button on the profile page.
3. The system checks if the user is already being followed by the follower:
    a. If not, the system proceeds with the follow action.
4. The system updates the database:
    a. Adds the follower to the followed user’s followers list.
    b. Updates the follower’s following list.
5. The system generates a notification for the followed user:
    a. The notification includes the follower's name and profile link.
6. The system updates the button on the follower’s view to display "Following."

❖ **Main Flow: Unfollow a User**

1. The follower navigates to the profile of the user they wish to unfollow.
2. The follower clicks the "Unfollow" button on the profile page.
3. The system checks if the user is currently being followed by the follower:
    a. If yes, the system proceeds with the unfollow action.
4. The system updates the database:
    a. Removes the follower from the followed user’s followers list.

b. Updates the follower's following list.

5. The system updates the button on the follower's view to display "Follow."

❖ Alternate flows :

4.2. If the system encounters a delay or error in updating the database, it will display an error message:

"Unable to process your request. Please try again later."

❖ Postconditions:

- ❖ Success Scenario:
  - ➢ If following, the follower is added to the followed user's followers list,
  - ➢ If unfollowing, the follower is removed from the followed user's followers list, and no notification is sent.
- ❖ Failure Scenarios:
  - ➢ The action is not completed due to a system error, and the follower is notified.

Use Case: Save a post

❖ **Actors:** Registered user, system.

❖ **Preconditions :**

1. The saver must be logged into their account.
2. The post to be saved must exist and be visible to the saver.

❖ **Main Flow:**

1. The saver navigates to a post visible in their feed or another user's profile.
2. The saver clicks the "Save" button or icon associated with the post.
3. The system updates the database:
    a. Associates the post with the saver's saved posts collection.
4. The system confirms the action:
    a. Displays a message: "Post has been saved successfully."
5. The post is now accessible in the saver's saved posts collection, accessible from their profile or dashboard.

❖ **Alternate flows :**

3.2. If the post has been deleted or made unavailable before the saver clicks the "Save" button, the system will display an error message:

- "This post is no longer available."

5.1. If the system fails to update the database due to a server or connectivity issue, the system will display an error message:

- "Unable to save this post. Please try again later."

❖ Postconditions:

- ❖ Success Scenario:
  - ➢ If following, the follower is added to the followed user's followers list,
  - ➢ If unfollowing, the follower is removed from the followed user's followers list, and no notification is sent.
- ❖ Failure Scenarios:
  - ➢ The action is not completed due to a system error, and the follower is notified.