CMPT 353 Final Project – Test Report

Name: Adam Baker NSID: adb888 Student Number: 11252243

Server Testing:

To test my backend server that connects the frontend application to the database, I performed several loadtests to confirm that I was getting responses at an acceptable rate for the following operations:

- Login system administrator:
  - Result: got responses in 1-47ms
- Create a user
  - Result: got responses in 1-62ms
- Login
  - Result: got responses in 2-26ms
- Attempt to login with incorrect username/password
  - Result: got error responses in all requests in 1-31ms
- Attempt to create a user that has already been created
  - Result: got error responses in all requests in 1-23ms
- Get all users
  - Result: got responses in 1-24ms
- Get all channels
  - Result: got responses in 1-33ms
- Add a channel
  - Result: got responses in 50-101ms
- Get all messages from a channel
  - Result: got responses in 1-27ms
- Add a message to a channel
  - Result: got responses in 57-116ms
- Add a reply to a message
  - Result: got responses in 58-116ms
- Add a reply to a reply:
  - Result: got responses in 50-108ms
- Get rating of an unrated message:
  - Result: got responses in 1-25ms
- Upvote a message:
  - Result: got responses in 58ms-124ms
- Downvote a message:
  - Result: got responses in 58-258ms
- Get rating of a message:
  - Result: got responses in 1-26ms
- Search for messages with a search string:
  - Result: got responses in 2-38ms

- Search for messages from a specific user:
    - Result: got responses in 2-38ms
- Search for user with most posts:
    - Result: got responses in 2-28ms
- Search for user with least posts:
    - Result: got responses in 2-26ms
- Search for user with highest rating:
    - Result: got responses in 63-151ms
- Search for user with lowest rating:
    - Result: got responses in 61-112ms

// Admin operations

- Delete a user
    - Result: got responses in 1-77ms
- Attempt to delete system admin
    - Result: got error responses in 1-22ms
- Delete a channel
    - Result: got responses in 1-82ms
- Delete a message with nested replies
    - Result: got responses in 2-12240ms

Conclusion: After testing my backend server, I can see some areas that could use some improvement. Namely, any operations that required nested database queries were longer. Removing messages with nested replies took especially long. A solution would be to refactor the server to use less nested database queries (perhaps by utilizing the JOIN MySQL command more).

Application Testing:

To test my frontend application, I performed the following operations again to see if expected behaviour occurred:

- Login system administrator:
    - Result: login successful
- Create a user
    - Result: user creation and login successful
- Login
    - Result: login successful
- Attempt to login for nonexistent user
    - Result: hit with error message as expected
- Attempt to create a user that has already been created
    - Result: hit with error message as expected
- Get all users
    - Result: users page displays all users properly (current user's name is highlighted with blue as intended)

- Get all channels
    - Result: channels page displays all channels properly
- Add a channel
    - Result: channel created and displayed successfully
- Get all messages from a channel
    - Result: channel page displays all messages nested properly
- Add a message to a channel
    - Result: message posted and displayed successfully
- Add a reply to a message
    - Result: reply posted and displayed successfully
- Add a reply to a reply:
    - Result: reply to a reply posted and displayed successfully
- Get rating of a message:
    - Result: ratings on all messages in a channel displaying properly
- Upvote a message:
    - Result: message upvoted successfully
- Downvote a message:
    - Result: message downvoted successfully
- Search for messages with a search string:
    - Result: all messages containing the given search string displaying properly
- Search for messages from a specific user:
    - Result: all messages by the given username displaying properly
- Search for user with most posts:
    - Result: user with the most posts displaying properly
- Search for user with least posts:
    - Result: user with the least posts displaying properly
- Search for user with highest rating:
    - Result: user with the highest rating displaying properly
- Search for user with lowest rating:
    - Result: user with the lowest rating displaying properly

// Admin operations

- Delete a user
    - Result: user successfully deleted (option not available to non-admin)
- Attempt to delete system admin
    - Result: hit with error alert as expected (option not available to non-admin)
- Delete a channel
    - Result: channel successfully deleted (option not available to non-admin)
- Delete a message with nested replies
    - Result: message and child messages all successfully deleted (option not available to non-admin)

I also performed the following operations to test login persistence:

- Login:
    - Result: user JSON appears in local storage with correct user id, username, and admin value
- Logout:
    - Result: user JSON removed from local storage properly
- Refresh when logged in:
    - Result: user remained logged in when refreshing from any page
- Close and reopen application when logged in:
    - Result: user remained logged in when application was reopened

Conclusion: After testing my frontend application, I see nothing but success. All implemented features are working as intended with no unexpected errors, and at a quick speed as well! The only thing that I would have maybe done differently is handling redirects for certain pages. For example, if a logged in user goes to the login page, they are met with a white screen. Ideally, we'd want to redirect to the channels pages. But that is an issue for another day!