TEST PLAN FOR
BRAINSTORM

ChangeLog

Version	Change Date	By	Description
1.0	Oct. 16, 2023	Roland Fehr	Adding to basic template (headings, lists, etc.)
1.1	Oct. 20, 2023	Akira	Added the testing list

1 Introduction

1.1 Scope

The Brainstorm web app will consist of seven features with their corresponding functional requirements:

1) User Account
    a. Register a new account via form completion and button press
    b. Login with an existing account via form completion and button press
    c. Verify login attempt and administer token to user's browser on successful login
    d. Sign out button that allows users to sign out of their account

2) Chat Room
    a. Create a chat room using a button on the home page (does not require that users be logged in)
    b. Create a chat room using a button when a user is logged in to their account

c. When a user selects the “create chat room” button, a new window appears that allows them to specify a title and description
d. Created chat rooms contain a numeric code that can be shared with other users so that they can join
e. Created chat rooms contain a list of all the users currently in the chat room

3) Real-Time Communication
    a. Users can communicate with others in a chat room in real-time
    b. When a user types and submits a message, all members of the chat room (including the user) will be able to see the message appear instantly

4) Brainstorming Sessions
    a. All users in a chat room have the option of hosting a brainstorming session by clicking a button
    b. When creating a brainstorming session, a new window appears that allow users to specify a title and description
    c. The host user of a brainstorming session can set custom timers for each round in a session
    d. Once a brainstorming session has been started, all users can type and submit their ideas for the session in a text box
    e. Once a session round is complete, all the ideas submitted by every user will appear together as text in random order

5) Idea Filtration
    a. At the end of a brainstorming session round, the host user can decide whether to conduct voting and/or generate a pros and cons list from each user
    b. When voting is selected, all users can vote for ideas that they like (one vote per user per idea)
    c. When a pros and cons list is selected, all users can generate pros and cons for each idea presented
    d. Once voting has finished and the host user is satisfied, they can decide to move to the next round, where ideas that were not voted on will be discarded
    e. The host user can run as many rounds as they like, and can end the session whenever they want

6) Management of Chat Rooms and Brainstorming Sessions
    a. Users can view all the chat rooms they are currently a part of, and can leave the chat room whenever they want with a single button press
    b. Users can view the results for past brainstorming sessions for each chat room by selecting an option icon
    c. Users can see all the members that are currently in a chat room as a vertical list

7) Respond to 100 users making 1000 requests per minute concurrently (non-functional)
    a. The system shall support 100 users logged in concurrently without crashing or corrupting data
    b. The system shall support 1000 user requests per minute concurrently without crashing or corrupting data

1.2 Roles and Responsibilities

Name	Net ID	GitHub Username	Role
Roland Fehr	fehrr1	rfehr-creator	Test Manager
Yee Tsung Kao	kaoj	Jacksonkao97	Frontend developer
Akira Cooper	coopera	nagaokakid	Database Manager
Ravdeep Singh	singhr27	singhr27	Frontend developer

2 Test Methodology

2.1 Test Levels

Unit Testing

1. User Accounts
    I. Test the constructor for the User.cs class to ensure that it has all the appropriate properties associated with a user when an instance is successfully made
    II. Test the UserService.cs class to ensure that it creates a new User object if the given username for registration doesn't already exist
    III. Test the UserService.cs class to ensure that it throws a UsernameExists exception if the given username already exists
    IV. Test the UserService.cs class to ensure that it throws an InvalidInput exception if the given username is null
    V. Test the MongoRepository.cs class to ensure that it can connect to the MongoDB collection for User documents
    VI. Test the AuthService.cs class to ensure that it creates a valid security token for a user when the username and password are valid for a login attempt
    VII. Test the AuthService.cs class to ensure that it removes the security token for a user when they log out of the app
    VIII. Test the AuthService.cs class to ensure it throws an Unauthorized exception if a login attempt contains an invalid username

IX. Test the AuthService.cs class to ensure it throws an Unauthorized exception if a login attempt contains an invalid password for a valid user
X. Test the AuthService.cs class to ensure it throws an Unauthorized exception if a login attempt contains an invalid password and username

2. Chat Room
   I. Test the constructor for the ChatRoom.cs class to ensure that it has all the appropriate properties associated with a chat room when an instance is successfully made
   II. Test the MongoRepository.cs class to ensure it can connect to the MongoDB collection for ChatRoom documents
   III. Test the ChatRoomService.cs class to ensure that it throws a UserNotFound exception when a non-registered user tries to create a chat room
   IV. Test the ChatRoomService.cs class to ensure that it throws a ChatRoomNotFound exception when it attempts to look for a chat room after being given an invalid chat room ID
   V. Test the ChatRoomService.cs class to ensure that it returns a list of ChatRoom objects when all the given chat room IDs are valid
   VI. Test the ChatRoomService.cs class to ensure that it throws a ChatRoomNotFound exception when it attempts to look for a chat room after being given an invalid join code
   VII. Test the ChatRoomService.cs class to ensure that it returns a valid ChatRoom object when a valid chat room join code is given
   VIII. Test the ChatRoomService.cs class to ensure it creates a valid ChatRoom object when given a valid CreateChatRoom request (the user ID is valid)
   IX. Test the ChatRoomService.cs class to ensure it throws an InvalidChatRoomRequest exception when given an invalid request (the user ID is invalid)
   X. Test the ChatRoomService.cs class to ensure it properly adds a user's ID to the ChatRoom object after the user creates a chat room successfully

3. Real-Time Communication
   I. Test the constructor for the ChatRoomMessage.cs class to ensure that it has all the appropriate properties associated with a chat room message when an instance is successfully made
   II. Test the constructor for the DirectMessage.cs class to ensure that it has all the appropriate properties associated with a direct message when an instance is successfully made
   III. Test the constructor for the DirectMessageHistory.cs class to ensure that it has all the appropriate properties associated with the history of direct messages between two users when an instance is successfully made

IV. Test the MongoRepository.cs class to ensure that it can connect to the MongoDB collection for DirectMessage and ChatRoomMessage documents
V. Test the DirectMessageService.cs class to ensure that it throws a Null exception when given a null value for the sender ID of a direct message
VI. Test the DirectMessageService.cs class to ensure that it throws a Null exception when given a null value for the receiver ID of a direct message
VII. Test the DirectMessageService.cs class to ensure that it returns a new list when no direct messages exist between two users (the sender just started a direct message with the receiver)
VIII. Test the DirectMessageService.cs class to ensure that it returns a list of all the direct message histories given a user's ID
IX. Test the OnlineUserService.cs class to ensure that when a new user ID is added with the corresponding connection ID, that the class' dictionary contains the connection ID value for that user ID
X. Test the OnlineUserService.cs class to ensure that removes the correct connection ID in its dictionary when given a connection ID in the "remove" method

4. Brainstorming Session
   I. Test the constructor for the BrainstormSession class by validating it's input (i.e.: all chat members are present in the brainstorming session object)
   II. Test the function that creates a new brainstorming session to ensure that it adds the brainstorming session object to all users in the session
   III. Test the function that sends brainstorm ideas to the corresponding brainstorming session to ensure the input is valid
   IV. Test the function that sends brainstorm ideas to the corresponding brainstorming session to ensure the string is stored correctly in the brainstorm session object
   V. Test the function that sends brainstorm ideas to the corresponding brainstorming session to ensure that the user is permitted to send the string (the user belongs to the chat room)
   VI. Test the function for creating a timer for a brainstorming session and make sure it throws an exception if the input is not valid (not a number)
   VII. Test the function for creating a timer for a brainstorming session and make sure it adds the timer to the brainstorm session object correctly
   VIII. Test the function for closing a brainstorm session to ensure that it saves the session results in the object and updates all users (removes the session ID from each user)
   IX. Test the function for sending brainstorm ideas to the session to ensure that all messages remain anonymous
   X. Test the function for aggregating brainstorm ideas to ensure that it appears as one list in random order for all users in the session

5. Idea Filtration
    I. Test the function for initiating voting among users to ensure that each user is sent a VotingPrompt object
    II. Test the function for initiating a pros and cons list from each user to ensure that a ProsAndCons object is created containing all the user IDs for the session
    III. Test the function for running a voting poll to ensure that when a vote for an idea is given, that it increments the number of votes for that idea in the session by one
    IV. Test the function for running a voting poll to ensure that users can only vote once per idea
    V. Test the function for taking responses for a pros and cons list by ensuring that each response is added to the ProsAndCons object for a given idea
    VI. Test the function for advancing to the next round in a session to ensure that all ideas with no votes are discarded in the Brainstorm Session object
    VII. Test the function for initiating a voting poll to ensure that the input is valid
    VIII. Test the function for initiating a voting poll to ensure that only the brainstorming session host user can activate them
    IX. Test the function for initating a pros and cons list by ensuring that the input is valid
    X. Test the function for initiating a pros and cons list by ensuring that only the brainstorming session host user can activate them

6. Management of Chat Rooms and Brainstorming Sessions
    I. Test the function for removing chat rooms from a user's list to ensure that it removes the correct chat room when given a chat room ID
    II. Test the function for removing a brainstorming session result from a user's list to ensure that it removes the correct session when given a session ID
    III. Test the function that provides a list of chat rooms for a user to ensure that all chat rooms for the user are returned as a list when given a user ID
    IV. Test the function that provides a list of brainstorming session results for a user to ensure that all session results are returned as a list when given a user ID
    V. Test the constructor for the ChatRoomListManager class to ensure that it contains all the properties for managing chat rooms for a single user when an instance is successfully made
    VI. Test the constructor for the BrainstormSessionResultListManager class to ensure that it contains all the properties for managing brainstorm session results for a single user when an instance is successfully made
    VII. Test the constructor for the VerifyAction class to ensure that it contains all the properties for asking a user to click "OK" or "Cancel" to confirm their actions (when removing a chat room or session result)
    VIII. Test the function in the VerifyAction class that sends a verification window to a user when they remove a chat room to ensure that it sends the correct object to the correct user

IX. Test the function in the VerifyAction class that receives user input for a verification window to ensure that it does not execute an action if the input is "Cancel"
X. Test the function in the VerifyAction class that receives user input for a verification window to ensure that it executes an action if the input is "OK"

Integration Testing

I. Test the user controller class and make sure that the login function works
II. Test the user controller class and make sure that the register function works
III. Test the user controller class and make sure that the login function return is valid
IV. Test the user controller class and make sure that the register function returns valid information
V. Test the chatroom controller and make that only a valid user can create a chatroom
VI. Test the chatroom controller and make sure it creates a room
VII. Test the chatroom controller and make sure it returns a valid chatroom
VIII. Test SignalR chatroom hub to make sure that any user including anonymous user can join a chat room
IX. Test SignalR chatroom hub to make sure it sends a message to all group members when it's instructed to
X. Test SignalR direct message hub to make sure it sends a direct message to the specified user
XI. Test all standard CRUD operations in the MongoRepository.cs class to ensure they work properly for each collection in the MongoDB database
XII. Test the UI buttons and ensure that it sends the correct JSON object to the back-end (i.e: create a chat room, join a chat room, etc.)

Acceptance Testing

1. User Accounts
   - Given that I'm a user and I go to the main page for the web app, there should be a section where I can fill in my login information. If I don't have an account, I can click the register button to create a new account. Once I'm done, I can enter my login credentials and sign into the app with my account.
   - Given that I'm a user and I'm already logged in to my account, I can click the "Sign Out" button to log out of my account and go back to the main page

2. Chat Room
   - Given that I'm a user, I can create a chat room from the main page or when I'm already logged in. All I have to do is click "Create Chat Room" and fill out a title and description
   - Give that I'm a user, I can join a chat room by entering the code in the "Join Chat Room" field either on the main page or when I'm already logged in

3. Real-Time Communication
   - Given that I'm a user, I can use the chat room text box to type in messages that I want to send to all chat members

- Given that I'm a user, I can see messages that I've sent and messages that other users have sent instantly in the chat room window
- Given that I'm a user, I can see messages that I've sent and the other user has sent instantly in the direct message chat room window

4. Brainstorming Session
   - Given that I'm a user, I can click the "Create Brainstorm Session" button when on the main page or home page to create a new brainstorming session
   - Given that I'm a user, once I've selected the option to create a brainstorm session, a window appears where I can specify a title and description for the session
   - Given that I'm a user, when I or someone else in a chat room starts a brainstorming session, me and all other members are taken to a new window where the brainstorming session begins

5. Idea Filtration
   - Given that I'm a user, I can type in ideas in a session and submit as many as I want
   - Given that I'm a user, I can see all the ideas submitted by every user (labeled anonymously) at the end of each session round
   - Given that I'm a user, I can vote on the ideas that I like at the end of each round
   - Given that I'm a user, I can create a pros and cons list for each idea at the end of each round
   - Given that I'm a host user for a session, I can set timers for each round
   - Given that I'm a host user for a session, I can end the session at any time
   - Given that I'm a host user, I can run additional rounds for a session, where the ideas with no votes don't appear in subsequent rounds

6. Management of Chat Rooms and Brainstorming Sessions
   - Given that I'm a user, I can see a list of all the chat rooms I'm a member of when I'm logged in
   - Given that I'm a user, I can see a list of all past brainstorming session results on a separate tab when I'm logged in
   - Given that I'm a user, I can leave chat rooms by clicking on the "X" for chat rooms in my list
   - Given that I'm a user, I can delete session results by clicking on the "X" for session results in my list

Regression Testing
We setup the CI pipeline with GitHub actions that builds the project and runs all tests when a pull request is made (this includes unit testing and integration testing).

Load Balance Testing
Since our project is mainly a chat application, the majority of our load testing will go towards ensuring we can send many real-time chats at the same time. We will create at least 100 users

and have them all send 10 messages per second. On top of that we'll create 1 user per second. The system should be able to handle this level of volume in terms of concurrent requests.

2.2 Test Completeness

The following criteria must be met to verify that testing is complete for the app:

- 100% back-end code coverage where all of the back-end source code is covered by test cases
- All the API endpoints return valid information
- Requests sent from UI to the API do not cause errors in the logic layer
- Responses sent from API to the UI do not cause errors in the browser
- CRUD operations sent from the API to the database do not cause errors in the API or lead to inconsistent data in the database
- All UI features (buttons, windows, chat list, etc) perform their necessary functions and do not cause the app to crash

3 Resource & Environment Needs

3.1 Testing Tools

The following tools will be used for testing:

- GitHub
- GitHub actions
- Visual Studio
- VS Code
- Docker Desktop
- MongoDB Atlas

3.2 Test Environment

The minimum hardware requirements for testing are:

- 2 CPU cores
- 1 GB RAM
- 1 GB HD

The minimum software requirements for testing are:

- Windows 7 and above
- Any Linux distribution
- Any Mac that can run Visual Studio
- GitHub Actions

- Web Browser

4 Terms/Acronyms

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
URL	Uniform Resource Locator
UI	User Interface
DTO	Data Transfer Object
CPU	Central Processing Unit
HD	Hard Disk
RAM	Random Access Memory
GB	Gigabyte
DB	Database
VS	Visual Studio