# TESTING PLAN FOR GIFHUB - CachingFeelings

---

## 0 ChangeLog

| Version | Change Date | By | Description |
|---|---|---|---|
| Version Number | Date of Change | Name of Person Who Made Changes | Description of Changes Made |
| 1 | 2024-02-19 | Dingyuan Zhang | Created |
| 2 | 2024-02-27 | Heejeong Kim | Edit |
| 3 | 2024-02-27 | Breanna Brown | Edit |

## 1 Introduction

### 1.1 Scope

**This test plan will test the following core features:**
**Functional Requirements:**

1. Account Creation
2. Student (Soul) Matching
3. Interact with Matches
4. Community Discovery
5. Icebreakers

**Non-functional Requirements:**

1. Non-Functional Secure Authentication

### 1.2 Roles and Responsibilities

| Name | UofM ID | GitHub Username | Role |
|---|---|---|---|
| Breanna Brown | millettb | bre9425 | Developer |
| Rahul Kumar | kumarr7 | bochacho | Developer |
| Oluwademilade Akinsola | akinsol2 | Demi-AK | Developer |
| Heejeong Kim | kimh4 | Heejoy | Developer |
| Dingyuan Zhang | zhangd7 | AllNEWAREAS | Developer |

# 2 Test Methodology

## 2.1 Test Levels

**Unit Tests:**

Core Feature 1: Account Creation

- Enables submit button when passwords match: Tests the functionality for users to change passwords, a direct aspect of account security during account creation or update.
- Renders inputs and validates username correctly: Directly related to account creation, ensuring that the username meets specific criteria.
- Validates password requirements correctly: Directly tests password strength requirements during account creation or update.
- Renders LoginForm component: Validates the login form's presence, directly supporting the user's ability to access their account.
- Allows the user to fill out the LoginForm: Tests user interaction with the login form, a critical part of accessing created accounts.
- Renders SignupForm component: Directly tests the presence of a signup form, essential for account creation.
- Renders Container component: While more general, this test can be included under account creation as the container might host sign up and login links.

Core Feature 2: Student (Soul) Matching

- Renders bio textarea with correct placeholder: Related to users sharing interests/preferences, which is foundational for matching algorithms.
- Renders select options correctly: Tests the presence of selectable options for user preferences, directly influencing the matching process.
- Renders user details: Essential for viewing the details of potential matches, making it a core part of the matching feature.

- Renders multiple User components: Supports displaying multiple matches to the user, a direct requirement for browsing through potential soul matches.
- Renders PopupWin component when open: Could be attributed to this feature if it's used for interacting with or viewing more details about matches.
- Calls onClose prop when close button is clicked: If the PopupWin is used within the matching context, closing it would be part of managing match interactions.
- Clicking on an OS option updates state: If we stretch its relevance, selecting an operating system might indirectly relate to user preferences, but it's less directly tied to the core functionalities of either feature based on the provided user stories.
- Ensure multiple programming language selections are possible: Similar to the OS option, this could relate to sharing technical interests for matching but does not directly support a primary function of the outlined features.
- Navigates to different routes: Essential for navigating the app to access different functionalities, including viewing matches, but it's more of a general utility than specific to matching alone.

Core Feature 3: Interact with Matches

- Renders MessageInput component: Ensures the input field for typing messages is present, which is fundamental for users to interact with their matches.
- Submits messages correctly: Tests if messages typed by the user are sent correctly upon submission, crucial for real-time communication.
- Renders ConversationHistory component: Verifies the display of past messages between users, essential for maintaining context in ongoing conversations.
- Notifies user of new messages: Checks if users receive notifications for new messages, a key aspect of user engagement and interaction.
- Enables message deletion: Tests the functionality that allows users to delete messages, ensuring user control over their conversation history.
- Filters conversations by user: Confirms the ability to view conversations with specific matches, streamlining the user's interaction experience.
- Renders MatchList component: Validates the display of a user's matches, foundational for choosing whom to interact with.
- Updates read status of messages: Tests if the read status of messages updates correctly, important for tracking conversation engagement.
- Blocks a user: Ensures the functionality to block a user works as intended, critical for user safety and comfort.
- Unblocks a user: Verifies the unblocking functionality, allowing users to manage their block list dynamically.

Core Feature 4: Community Discovery

- Renders CommunityFeed component: Tests the presence of the community feed, where users discover posts and discussions.
- Posts a new thought: Checks the functionality for users to post new thoughts or content, a primary interaction within the community.
- Likes a post: Ensures users can like a post, facilitating engagement within the community.

- Dislikes a post: Confirms the functionality for users to dislike a post, offering feedback on community content.
- Comments on a post: Tests the ability for users to comment on posts, encouraging discussions and interactions.
- Renders TopicSelection component: Verifies the display of options for selecting topics of interest, directing the user experience within the community.
- Filters posts by topic: Ensures posts can be filtered by selected topics, tailoring the community feed to user interests.
- Deletes a user's own post: Confirms that users can delete their posts, providing control over their content.
- Updates a post: Tests the ability to edit and update a post, essential for maintaining relevance and accuracy of shared content.
- Renders UserProfiles from posts: Validates that user profiles are accessible from posts, fostering community connections.

Core Feature 5: Icebreakers

- Renders IcebreakerQuestions component: Ensures the display of icebreaker questions, crucial for initiating conversations.
- Selects an icebreaker question: Tests the functionality for users to select an icebreaker question, facilitating the start of a conversation.
- Sends an icebreaker message: Confirms that selected icebreaker messages are sent correctly to a match, key for engaging with new matches.
- Tracks usage of icebreakers: Verifies the system tracks which icebreakers have been used, optimizing the selection for future interactions.
- Renders IcebreakerResponse component: Tests the display of responses to icebreaker messages, essential for continuing conversations.
- Customizes icebreaker questions: Ensures users can create custom icebreaker questions, adding personalization to the feature.
- Retrieves popular icebreakers: Checks the system's ability to suggest popular icebreaker questions based on user feedback and usage.
- Feedback on icebreaker effectiveness: Tests the functionality for users to provide feedback on the effectiveness of an icebreaker.
- Categorizes icebreakers by theme: Confirms that icebreakers are categorized by themes, aiding users in choosing relevant questions.
- Updates icebreaker suggestions dynamically: Verifies that the list of suggested icebreakers updates based on user interactions and preferences.

**Integration Test:**

Core Feature 1: Account Creation

- Test sign up form information whether stored in database or not when successfully signed up for the new user
- Test whether backend sent appropriate information back to frontend, such as token
- Test login page when no valid user, backend sent error message
- Test when try to create an account with an already registered
- Test the sign up process enforces password strength requirements

- Test that all mandatory fields are validated on the backend
- Test the successful account creation
- Test that user creation is successful upon login, and are appropriately terminated upon logout.

Core Feature 2: Student (Soul) Matching

- Test the matching algorithm to ensure that users are matched based on shared interests
- Test that cached users can see each other in the match lists
- Test the system response when there are no available matches
- Test that users can update their matching preferences.
- Test reported or blocked users are no longer visible
- Test that the match list updates in real-time
- Test that user profiles are accessible and accurate within the matching interface
- Test the system's ability to handle a high volume of match requests
- Test that private user data not being disclosed without permission
- Test that matched users can initiate conversations without issues

Core Feature 3: Interact with Matches

- Test the messaging functionality to ensure that users can send and receive messages with their matches
- Test that the conversation history is correctly saved and displayed for each match
- Test the user attempts to message a blocked user
- Test the ability to close or end a conversation.
- Test for deleting a match and confirm that deleted matches are removed from the match list
- Test the images are correctly sent and received with proper display
- Test the responsiveness of the messaging interface across different devices and screen sizes

Core Feature 4: Community Discovery

- Test the functionality for users to select and explore topics of interest within the "Community Discovery" page
- Test that thoughts related to the selected topic are correctly displayed to the user
- Test that user can like or dislike posts and these are correctly recorded
- Test that share thoughts to the topic are visible to others
- Test the date of posting sorted correctly
- Test user profiles are accessible from the community posts
- Test the system's handling of inappropriate content

Core Feature 5: Icebreakers

- Test the display and functionality of suggested icebreaker questions on the "Finally" page
- Test that users can select an icebreaker question and successfully initiate a conversation based on that question

**Acceptance Test:**

Core Feature 1: Account Creation

- Click on the "Sign Up" button to begin the registration process

On the registration page, fill in the required fields:
    - Enter username
    - Create a password and enter it in the “Password” field
    - Confirm the password by entering it again in the “Confirm Password” field
    - Provide a postal code
    - Enter my birthday

  After filling in these details, click on the “Next” button to proceed

  In the next section, select my prederences:
    - Choose who I am looking to connect with
    - Specify who I want to date
    - Indicate the gender I identify as

  Click on the “Next” button

  Now, I will be prompted to personalize my profile:
    - Enter a bio
    - Choose my interests from the provided list

  After completing my profile, click on the “Next” button

  In the final step, I will upload pictures:
    - Choose the file I want to upload
    - Click on the “Submit” button to complete the registration process

  After submission, I will be navigated to the “Try” page
- Enter my username and password then it will direct to the “Try” page once I click the “Login” button.

Core Feature 2: Student (Soul) Matching
- Can drag the sphere and the sphere will rotate.
- Can click on the node, then it will prompt the user profile window
    - Click on close button to close the window
    - Click on “Like” button to like the person

Core Feature 3: Interact with Matches
- Click a user from the “Catch” page, and it will direct me to the “Finally” page to where messaging with the chosen user is enabled
- I can send messages or receive messages on the “Finally” page.
    - I can close the window

Core Feature 4: Community Discovery
- In the “Community Discovery” page, I get to choose one topic that I am interested in.
    - I get to click the “Like” button or “Dislike” button to show my interest.

Core Feature 5: Icebreakers
- In the “Finally” page, there are suggested questions that act as icebreakers for the user to start a conversation.
- I can choose one of the sentences provided and start a conversation.

**Regression Test:**

We configured the CI pipeline using GitHub Actions to automatically build the project and execute all tests whenever a pull request is submitted.

**Load Balance Test:**
The system is expected to efficiently manage the specified volume of concurrent requests without compromising performance. Initiate the creation of a new user profile at a consistent rate of 1 user per second, engage 100 active users within the app and direct these users to send 10 messages per second.

### 2.2 Test Completeness

Criterias for Test Completeness:

- Automated testings should be executed
- All the bugs and failed test should be checked and fixed
- All tests should cover 100% of the codebase
- Bugs should be fixed and reported

# 3 Resource & Environment Needs

### 3.1 Testing Tools

Following tools are used in our project for testing:

- Automation: GitHub Actions
- Backend Test: Jest
- Frontend Test: Jest

### 3.2 Test Environment

Following environments  are used in our project for testing:

- MacOS Sonoma
- Docker

# 4 Terms/Acronyms

| Term/Acronym | Definition |
|---|---|
| API | Application Program Interface |
| AUT | Application Under Test |
| CI | Continuous Integration |