# SW Engineering CSC648-848-05 Spring 2023

DebugMe

Team 02

Emilee Padilla - Team Lead
emann@mail.sfsu.edu
Jijeong Lee - Frontend Lead
Cristobal Padilla - GitHub Lead
Matthew Bush - Backend Lead
Khayotbek Azimov - Cloud Lead
Jeffrey Ma - Database Lead
Hector Magallanes - Document Editor Lead

Milestone 4
05/18/2023

History Versions Table

| Revision | Date |
|---|---|
| m4v1 | 05/18/23 |
| m3v2 | 05/18/23 |
| m3V1 | 04/27/23 |
| m2V2 | 04/27/23 |
| m2V1 | 03/31/30 |
| m1V2 | 04/18/23 |
| m1V1 | 02/23/23 |

# Table of Contents

# 1. Product Summary

**Name of the product:**
DebugMe

**Explicit itemized list of all major committed functions**

1. Unregistered users shall be able to create an account
2. Registered users shall be able to delete their accounts
3. Registered users shall be able to log in
4. Registered users shall be able to log out
5. Registered users shall be able to see their profiles
6. Registered users shall be able to update their personal information
    - 6.1. Registered users shall be able to upload a profile picture
    - 6.2. Registered users shall be able to change their username
    - 6.3. Registered users shall be able to add a bio to their personal information
    - 6.4. Registered users shall be able to edit their bio
7. Registered users shall be able to change from Free to Premium subscription plan
8. Registered users shall be able to change from Premium to Free subscription plan
9. Registered users shall be able to see other registered users' username
10. Registered users shall be able to see other registered users' profile picture
11. Registered users shall be able to see other registered users' bio
12. Registered users shall be able to see all posts in the Forum
13. Registered users shall be able to search Forum posts by title
14. Registered users shall be able to create new Forum posts
    - 14.1. Registered users shall be able to add a title to their post
    - 14.2. Registered users shall be able to add text to their post body
    - 14.3. Registered users shall be able to add images to their posts
15. Registered users shall be able to delete their posts
16. Registered users shall be able to see any post's comments
17. Registered users shall be able to leave a comment on any post
18. Registered users shall be able to delete their post comments
19. Registered users shall be able to like any post
20. Registered users shall be able to send a direct message to other registered users

21. Registered users shall be able to receive direct messages from other registered users
22. Premium users shall be able to see a list of their upcoming Mentoring Sessions
23. Premium users shall be able to see all Premium Guides
24. Premium users shall be able to access any Premium Guide
25. Premium users shall be able to rate any Premium Guide
26. Premium users shall be able to request a Mentoring Session from any Mentor
27. Premium users shall be able to cancel their Mentoring Sessions
28. Premium users shall be able to save any Premium Guide to their list of saved guides
29. Premium users shall be able to remove any Premium Guide to their list of saved guides
30. Registered users shall be able to create Premium Guides
    - 30.1. Registered users shall be able to add a title to their Premium Guide
    - 30.2. Registered users shall be able to add text to their Premium Guide body
    - 30.3. Registered users shall be able to add an image to their Premium Guide
31. Mentors shall be able to delete their Premium Guides
32. Mentors shall be able to see their Mentoring Session requests
33. Mentors shall be able to approve Mentoring Session requests
34. Mentors shall be able to refuse Mentoring Session requests

**Say what is unique in your product:**
Our product is unique in the fact that it serves the niche community of people looking for help specifically with getting internships in Tech. There are lots of services for jobs but not for internships which we need to help get said jobs. Most of all we utilize our premium feature which is our premium guides. Through this feature,

**URL of your product accessible to instructor, on deployment server:**
[http://18.237.101.95/](http://18.237.101.95/)

## 2. Usability Test Plan 2.5 pages max.

**Purpose:**

- The purpose of this usability test is to evaluate the effectiveness, efficiency, and user satisfaction of the five key functions on our platform: creating a premium guide, saving the guide as a favorite, rating the guide, requesting a mentoring session through premium guide, and canceling the mentoring session request.

**Problem Statement and Objectives:**

- The objective is to identify these issues, measure their impact on user experience, and gather data to improve usability.

**Test Objectives per Function**

- **Creating a Premium Guide:** The objective is to test the simplicity and intuitiveness of the process. It is important that users can create a guide without unnecessary complications or confusions, as this is one of the primary features of the platform.
- **Saving a Premium Guide as a Favorite:** The goal is to assess the process's ease and user-friendliness. Users should be able to quickly save their favorite guides and access them later without difficulty.
- **Rating a Premium Guide:** The purpose is to evaluate the visibility and accessibility of the rating system, and how smoothly users can leave a rating. Users should be able to rate a guide easily, promoting engagement and interaction on the platform.
- **Requesting a Mentoring Session:** The objective is to examine the ease of use and reliability of the appointment booking system. This function is key for users to connect with mentors and enhance their learning experience.
- **Canceling a Mentoring Session:** The goal is to test the efficiency and clarity of this process. Users should be able to cancel a mentoring session easily, providing flexibility and control over their interactions on the platform.

**Test Description per Function**

- **Use the following user for performing your usability test:**
  - Email: joseo@mail.com
  - PW: Password1!

- **Saving a Premium Guide as a Favorite:**
  **System Setup:** Desktop browsers(Chrome, Safari, Firefox), Operating Systems (Windows and MacOS).
  **Starting Point:** Premium Guide Page.
  **Intended Users:** Registered premium users who would like to save a guide as favorite.

**URL:** http://18.237.101.95/premiumguides
**Measurement:** Time to save a guide as favorite, user satisfaction, any errors encountered.

- **Rating a Premium Guide:**
  **System Setup:** Desktop browsers(Chrome, Safari, Firefox), Operating Systems (Windows and MacOS).
  **Starting Point:** Premium Guide.
  **Intended Users:** Registered premium users after accessing the premium guide.
  **URL:** http://18.237.101.95/premiumguides
  **Measurement:** Time taken to rate, visibility of rating system, user satisfaction, any errors encountered.

- **Requesting a Mentoring Session Through Premium Guide:**
  **System Setup:** Desktop browsers(Chrome, Safari, Firefox), Operating Systems (Windows and MacOS).
  **Starting Point:** Premium Guide page.
  **Intended Users:** Registered premium users who would like to request a mentoring session with a Mentor.
  **URL**: http://18.237.101.95/premiumguides
  **Measurement**: Time taken to request a session, number of steps, user satisfaction, any errors encountered.

- **Canceling a Mentoring Session:**
  **System Setup**: Desktop browsers(Chrome, Safari, Firefox), Operating Systems (Windows and MacOS).
  **Starting Point**: Premium user on their "my page".
  **Intended Users**: Registered premium users who have requested a mentoring session.
  **URL**: http://18.237.101.95/mypage
  **Measurement**: Time taken to cancel a session, clarity of cancellation process, user satisfaction, any errors encountered.

- **Creating a Premium Guide:**
  **System Setup:** Desktop browsers(Chrome, Safari, Firefox), Operating Systems (Windows and MacOS).
  **Starting Point:** Premium Guides page.
  **Intended Users:** Registered users who would like to become Mentors.
  **URL:** http://18.237.101.95/premiumguides
  **Measurement:** Time taken to create a guide, number of steps, any errors encountered, user satisfaction.

**User Profile:**

- Our typical users are tech-savvy people who want to help each other find success in their tech internships

**Method:**

- The chosen method is the task-based usability testing, where each function will be tested separately by the user performing a task associated with the function. The tasks will be realistic and represent typical use of the system.

**Test Environment:**

- The test will be conducted in a controlled environment resembling a typical user setting. This includes a standard desktop computer with an internet connection. The test will also be observed and recorded for further analysis.

**Test Monitor Role:**

- The test monitor will guide the participants through the process, answer any queries, and ensure that the test stays on track. The monitor will not intervene unless the participant asks for help.

**Usability Test Table: Effectiveness**

| Test/use case | % Completed | Errors | Comments |
|---|---|---|---|
| Save Guide as favorite | 75% | | Button is off center. |
| Rate Premium Guide | 100% | | |
| Request Mentoring Session through premium guide | 75% | Button was not doing anything, had to refresh page.<br>Was trying to request mentor session with herself, we didn't account for that edge case and no error msg was displayed | Button is off center. |

| | | | |
|---|---|---|---|
| Cancel Mentoring Session | 62% | There was no session to cancel. | |
| Create a Premium Guide | 75% | | - Creation page looks kinda bland, not intuitive.<br>- Difficult time figuring out where to start to get to creating a guide. |

**Usability Test Table: Efficiency**

| Test/use case | Time to Complete Task (avg) | Steps Taken (in clicks) |
|---|---|---|
| Save Guide as favorite | 10s | 3 |
| Rate Premium Guide | 10s | 3 |
| Request Mentoring Session through premium guide | 10s | 5 |
| Cancel Mentoring Session | 25s | 8 |
| Create a Premium Guide | 60s | 10 |

## User Satisfaction: Likert Questionnaire

| | Strongly Disagree (1) | Disagree (2) | Neither Agree nor Disagree (3) | Agree (4) | Strongly Agree (5) |
|---|---|---|---|---|---|
| I found it easy to create a guide | 1 | 1 | 2 | 3 | 1 |
| The process of creating a guide was intuitive | 1 | 2 | 1 | 3 | 1 |
| I am satisfied with the tools provided for guide creation | 1 | 1 | 2 | 2 | 2 |
| I can easily save a guide as my favorite | 1 | 0 | 1 | 3 | 3 |
| The option to save a guide as a favorite is clearly visible | 1 | 0 | 0 | 4 | 3 |
| I am satisfied with the 'save as favorite' function | 1 | 0 | 1 | 4 | 2 |
| It is simple for me to rate a guide | 1 | 0 | 0 | 2 | 5 |
| The rating system is clear and understandable | 1 | 1 | 0 | 2 | 4 |
| I am satisfied with the guide rating process | 1 | 1 | 3 | 1 | 2 |
| Requesting a mentoring session through a premium guide is straightforward | 1 | 1 | 3 | 2 | 1 |
| I understand how to request a mentoring session | 1 | 0 | 1 | 4 | 2 |
| I am satisfied with the process of requesting a mentoring session | 1 | 1 | 3 | 2 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| **Canceling a mentoring session is easy** | **1** | **1** | **2** | **4** | **0** |
| **The option to cancel a mentoring session is clearly visible** | **1** | **1** | **2** | **3** | **1** |
| **I am satisfied with the cancellation process of a mentoring session** | **1** | **0** | **2** | **5** | **0** |

## 3. QA Test Plan

1. The system shall respond visually within 7 seconds.
    a. System Setup:
        i. Hardware: Computer with internet connection
        ii. Software: User logged in to DebugMe on a supported browser
        iii. URL: http://18.237.101.95/posts
    b. Feature to be tested: Page load time
    c. QA Test Plan: Open any DebugMe page and check that it loads within the specified time

2. Users shall see their posts within 1 min after successfully posting.
    a. System Setup:
        i. Hardware: Computer with internet connection
        ii. Software: User logged in to DebugMe on a supported browser
        iii. URL: http://18.237.101.95/posts
    b. Feature to be tested: Forum Post creation
    c. QA Test Plan: Create a new post and verify that it is accessible within the specified time

3. Registered user passwords shall contain at least one uppercase letter, one digit, one special character, and a minimum of 8 characters
    a. System Setup:
        i. Hardware: Computer with internet connection
        ii. Software: Supported browser showing DebugMe Sign Up page
        iii. URL: http://18.237.101.95/signup
    b. Feature to be tested: New user password validation
    c. QA Test Plan: Register a new user with a password that fulfills all of the necessary requirements

4. Registered users shall stay in logged in with the same browser
    a. System Setup:
        i. Hardware: Computer with internet connection
        ii. Software: User logged in to DebugMe on a supported browser
        iii. URL: http://18.237.101.95/premiumguides
    b. Feature to be tested: Browser session persistence
    c. QA Test Plan: Visit Premium Guides page while logged in, close the tab, open the page again in a new tab and check that user is still logged in

5. Passwords shall be encrypted
    a. System Setup:
        i. Hardware: Computer with internet connection

    ii. Software: Supported browser showing Sign In page, MySQL Workbench with an open connection to the DebugMe database
    iii. URL: http://18.237.101.95/signin

b. Feature to be tested: Password encryption
c. QA Test Plan: Register a new user and verify that their password is encrypted in the database

| Test # | Test Title | Description | Input | Output | Results |
|---|---|---|---|---|---|
| 1 | Page load time | Verify that DebugMe pages load in 7 seconds or less | Open Forum page | Forum page shows all content within 7 seconds | PASS |
| 2 | Post availability after creation | Verify that posts are accessible within 1 minute after creation | Create a new post | Post is accessible within 1 minute | PASS |
| 3 | Password validation | Verify that users can only create an account if their password meets all the requirements | Create user with a valid password | Password is accepted and user is created | PASS |
| 4 | Browser session persistence | Verify that users stay logged in even after closing DebugMe in their browser | Close DebugMe while logged in and reopen in the same browser | User is still logged in after reopening DebugMe | PASS |
| 5 | Password encryption | Verify that registered users have their passwords encrypted | Register a new user | User is created and their password is encrypted in the database | PASS |

## 4. Code Review

- Coding Style:
  - Frontend: Airbnb React/JSX Style
  - Backend: Google Python Style
- Code to be reviewed:

```
import React, { useState } from "react"; // Needed for AWS since it's using node 16
import PropTypes from 'prop-types';
import { toast } from "react-toastify";
import { customAxios } from "../../utils/customAxios";
import Button from "../../Components/Button";
import Modal from "../../Components/Modal";
import Textarea from "../../Components/Textarea";
import useAuth from "../../Hooks/useAuth";
import "./mypage.css";

const propTypes = {
  menteeSessions: PropTypes.array,
  mentorSessions: PropTypes.array,
  mentoringRequests: PropTypes.array
};

function MentoringSessions({ menteeSessions, mentorSessions, mentoringRequests }) {
  const [showMessageModal, setShowMessageModal] = useState(false);
  const [messageInfo, setMessageInfo] = useState({});
  const [content, setContent] = useState("");
  const { auth } = useAuth();

  /**
 * openMessageModal() sets message information and open message Modal
 * based on the passed-in sender, receiver information
 */
  const openMessageModal = (sender, receiver) => {
    setMessageInfo({
      sender_id: sender.id,
      sender_email: sender.email,
      receiver_id: receiver.id,
      receiver_email: receiver.email,
      receiver_name: receiver.name,
    })
    setShowMessageModal(true);
  }

  const postMessage = async () => {
    try {
      const data = {
        sender_id: messageInfo.sender_id,
        sender_email: messageInfo.sender_email,
        receiver_id: messageInfo.receiver_id,
        receiver_email: messageInfo.receiver_email,
        content: content
      };
      const res = await customAxios({
        method: "post",
        url: "/api/messages",
        data,
        headers: {
          "Content-Type": "application/x-www-form-urlencoded",
        },
      });
      if (res.status === 200 || res.status === 201) {
        toast.success("Your message has been sent.", {
          position: "bottom-left",
          autoClose: 3000,
          hideProgressBar: true,
        });
        setShowMessageModal(false);
      }
    } catch (e) {
      console.log(e);
    }
  };

  /**
* handleMentoring() requests mentoring session status changes to server
* based on apiUrl and mentoringId
*/
  const handleMentoring = async (apiUrl, mentoringId) => {
    try {
      const res = await customAxios({
        method: "post",
        url: apiUrl,
        data: { id: mentoringId },
        headers: {
          "Content-Type": "application/x-www-form-urlencoded",
        },
      });
      if (res.status === 200 || res.staus == 201) {
        window.location.reload();
      }
    } catch (e) {
      console.log(e);
    }
  };
```

- Why this code was chosen: Mentoring Sessions are one of the main functions related to Premium Guides, which are our superior feature. This code was chosen since it's related to how Premium Users and Mentors manage their Mentoring Sessions.

- Feedback from a team member: Code looks well formatted and is fairly self explanatory to read. However, some in-line comments within the functions would help to identify what each code block is doing.

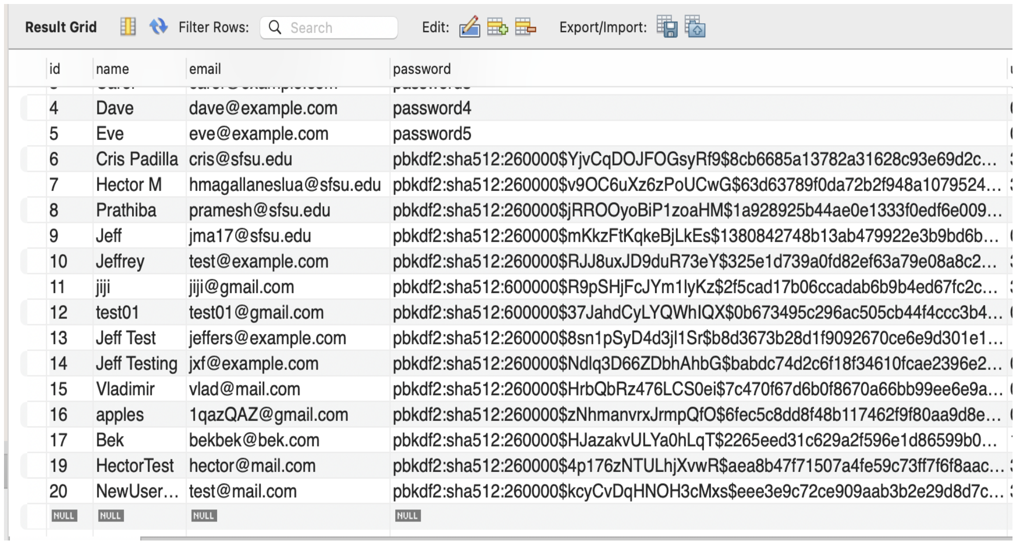
- Feedback from another team:  The code appears to be well organized and nicely formatted. . They have used Imports properly for the components. They have used useStates such as showMessageModal, messageInfo, content, and auth and all the variables are appropriately initialized with their initial values. For the handleMentoring function, it seems this function handles mentoring session status change by making POST request to the apiUrl, they used  the try-catch blocks to catch errors during API requests, it lacks detailed error handling and error messages to  handle potential errors or provide a better error handling with user feedback.

## 5. Self-Check On Best Practices For Security

- List major assets you are protecting

    - User passwords

- Confirm that you encrypt PW in the DB (describe the process and show real examples (i.e screenshots……)
    - Passwords are hashed using Flask's werkzeug.security module.
    - **PBKDF2** is the key derivation function used by werkzeug.security module.
    - **SHA 512** is the hashing method.
    - **60,000** is the number of iterations.
    - **Salt** is 16 characters long.
    - The encrypted password is then stored in

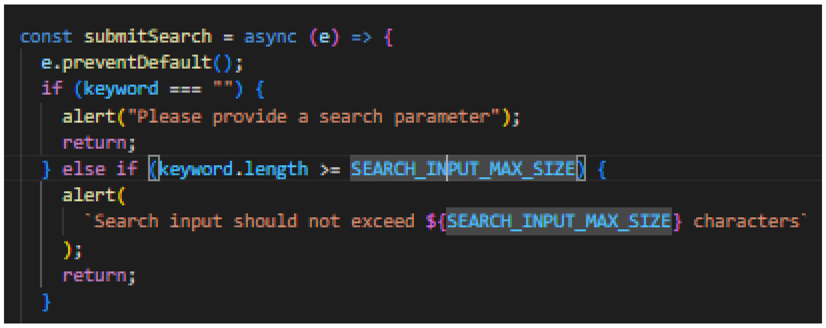


Result Grid | Filter Rows: Search | Edit: | Export/Import:

| id | name | email | password |
|---|---|---|---|
| 4 | Dave | dave@example.com | password4 |
| 5 | Eve | eve@example.com | password5 |
| 6 | Cris Padilla | cris@sfsu.edu | pbkdf2:sha512:260000$YjvCqDOJFOGsyRf9$8cb6685a13782a31628c93e69d2c… |
| 7 | Hector M | hmagallaneslua@sfsu.edu | pbkdf2:sha512:260000$v9OC6uXz6zPoUCwG$63d63789f0da72b2f948a1079524… |
| 8 | Prathiba | pramesh@sfsu.edu | pbkdf2:sha512:260000$jRROOyoBiP1zoaHM$1a928925b44ae0e1333f0edf6e009… |
| 9 | Jeff | jma17@sfsu.edu | pbkdf2:sha512:260000$mKkzFtKqkeBjLkEs$1380842748b13ab479922e3b9bd6b… |
| 10 | Jeffrey | test@example.com | pbkdf2:sha512:260000$RJJ8uxJD9duR73eY$325e1d739a0fd82ef63a79e08a8c2… |
| 11 | jiji | jiji@gmail.com | pbkdf2:sha512:600000$R9pSHjFcJYm1lyKz$2f5cad17b06ccadab6b9b4ed67fc2c… |
| 12 | test01 | test01@gmail.com | pbkdf2:sha512:600000$37JahdCyLYQWhIQX$0b673495c296ac505cb44f4ccc3b4… |
| 13 | Jeff Test | jeffers@example.com | pbkdf2:sha512:260000$8sn1pSyD4d3jl1Sr$b8d3673b28d1f9092670ce6e9d301e1… |
| 14 | Jeff Testing | jxf@example.com | pbkdf2:sha512:260000$Ndlq3D66ZDbhAhbG$babdc74d2c6f18f34610fcae2396e2… |
| 15 | Vladimir | vlad@mail.com | pbkdf2:sha512:260000$HrbQbRz476LCS0ei$7c470f67d6b0f8670a66bb99ee6e9a… |
| 16 | apples | 1qazQAZ@gmail.com | pbkdf2:sha512:260000$zNhmanvrxJrmpQfO$6fec5c8dd8f48b117462f9f80aa9d8e… |
| 17 | Bek | bekbek@bek.com | pbkdf2:sha512:260000$HJazakvULYa0hLqT$2265eed31c629a2f596e1d86599b0… |
| 19 | HectorTest | hector@mail.com | pbkdf2:sha512:260000$4p176zNTULhjXvwR$aea8b47f71507a4fe59c73ff7f6f8aac… |
| 20 | NewUser… | test@mail.com | pbkdf2:sha512:260000$kcyCvDqHNOH3cMxs$eee3e9c72ce909aab3b2e29d8d7c… |
| NULL | NULL | NULL | NULL |

the password field of the User's table in the database.

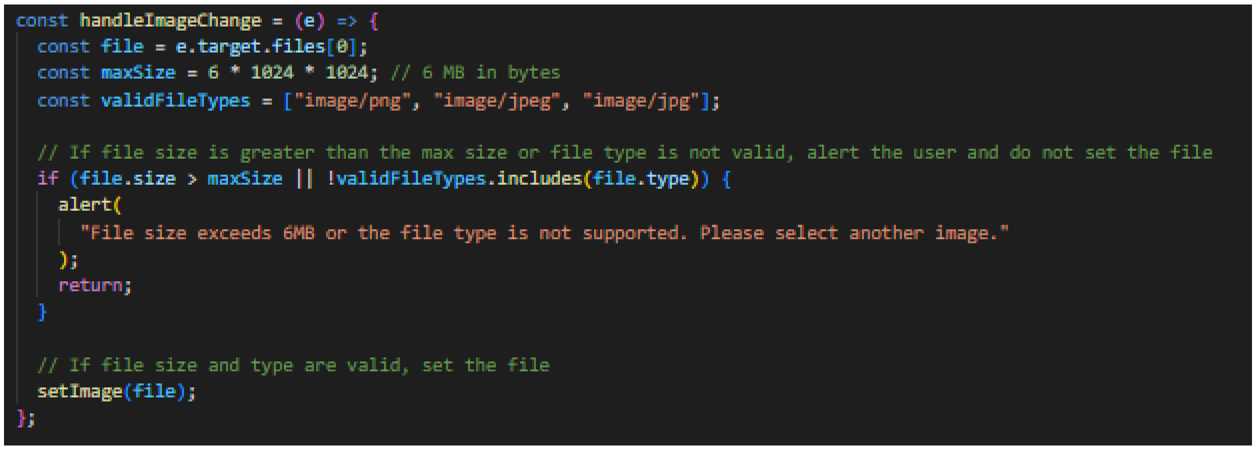
- Confirm Input data validation (list what is being validated and what code you used) – we request you validate search bar input
- Input size: max is 250 characters



```
const submitSearch = async (e) => {
  e.preventDefault();
  if (keyword === "") {
    alert("Please provide a search parameter");
    return;
  } else if (keyword.length >= SEARCH_INPUT_MAX_SIZE) {
    alert(
      `Search input should not exceed ${SEARCH_INPUT_MAX_SIZE} characters`
    );
    return;
  }
```

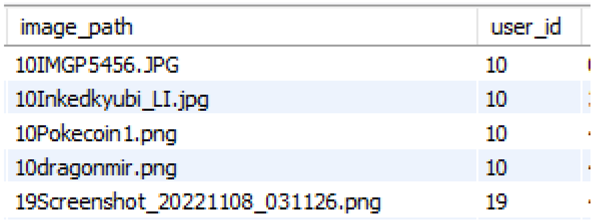
Search size is limited by the SEARCH_INPUT_MAX_SIZE value in the code. We ensure that the character limit is under or equal to 250 characters.

- Image Size, Image file type (jpg / jpeg / png)

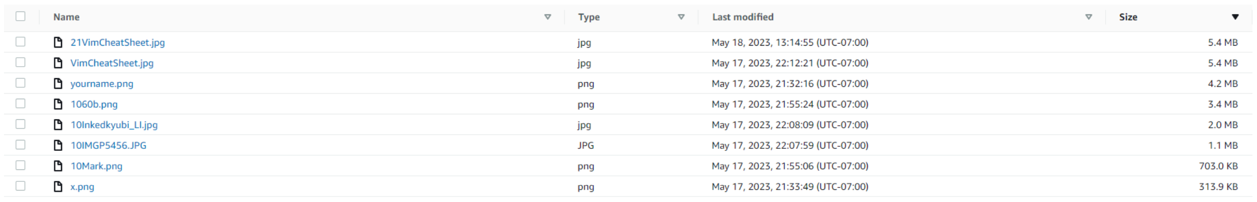


```
const handleImageChange = (e) => {
  const file = e.target.files[0];
  const maxSize = 6 * 1024 * 1024; // 6 MB in bytes
  const validFileTypes = ["image/png", "image/jpeg", "image/jpg"];

  // If file size is greater than the max size or file type is not valid, alert the user and do not set the file
  if (file.size > maxSize || !validFileTypes.includes(file.type)) {
    alert(
      "File size exceeds 6MB or the file type is not supported. Please select another image."
    );
    return;
  }

  // If file size and type are valid, set the file
  setImage(file);
};
```

We validate input data such as File size and file types through the code above. It checks the file size and or file type and if it is over the approved limit it will send an alert to try to select another image.



| image_path | user_id |
|---|---|
| 10IMGP5456.JPG | 10 |
| 10Inkedkyubi_LI.jpg | 10 |
| 10Pokecoin1.png | 10 |
| 10dragonmir.png | 10 |
| 19Screenshot_20221108_031126.png | 19 |



| Name | Type | Last modified | Size |
|---|---|---|---|
| 21VimCheatSheet.jpg | jpg | May 18, 2023, 13:14:55 (UTC-07:00) | 5.4 MB |
| VimCheatSheet.jpg | jpg | May 17, 2023, 22:12:21 (UTC-07:00) | 5.4 MB |
| yourname.png | png | May 17, 2023, 21:32:16 (UTC-07:00) | 4.2 MB |
| 1060b.png | png | May 17, 2023, 21:55:24 (UTC-07:00) | 3.4 MB |
| 10Inkedkyubi_LI.jpg | jpg | May 17, 2023, 22:08:09 (UTC-07:00) | 2.0 MB |
| 10IMGP5456.JPG | JPG | May 17, 2023, 22:07:59 (UTC-07:00) | 1.1 MB |
| 10Mark.png | png | May 17, 2023, 21:55:06 (UTC-07:00) | 703.0 KB |
| x.png | png | May 17, 2023, 21:33:49 (UTC-07:00) | 313.9 KB |

Once the image is stored in the database it is renamed to {userid}+filename format. And as you can see in the above image no image surpasses the approved file size.

## 6. Self-Check: Adherence to original Non-Functional Specs

1. **System**
   1.1. The application shall be accessible on Google Chrome version 110.0+ **DONE**
   1.2. The application shall be accessible on Firefox version 109.0+ **DONE**
   1.3. The application shall be accessible on Safari version 15.0+**DONE**

2. **Performance**
   2.1. The system shall respond visually within 7 seconds **DONE**
   2.2. Users shall see their posts within 1 min after successfully posting. **DONE**

3. **Storage, Security, Environmental**
   3.1. File size for uploaded images in no time shall exceed 6 megabytes **DONE**
   3.2. Accepted image file types are PNG, JPG, and JPEG **DONE**
   3.3. Registered users shall have their emails verified **ISSUE:** We will not be implementing this requirement
   3.4. An approved user shall have their email stored **DONE**
   3.5. Registered users must make a password that contains a capital letter and a number **DONE**
   3.6. Registered users shall have unique usernames **DONE**
   3.7. Registered users shall stay in logged in with the same browser **DONE**

4. **Marketing, Legal**
   4.1. The logo shall be displayed on the footer of every page **ON TRACK**
   4.2. All registered users must accept our terms and conditions **DONE**
   4.3. A link to our terms and conditions shall be shown on the footer of every page. **DONE**
   4.4. The logo shall be displayed on the header of every page. **DONE**
   4.5. A link to contact information shall be shown on the footer of every page. **DONE**

5. **Content**
   5.1. Each page shall have both a header and a footer **DONE**
   5.2. Each page shall contain proper semantic tags **ON TRACK**

6. **Privacy**
   6.1. Unregistered users shall not have their information stored **DONE**
   6.2. User information shall not be shared with third parties unless compelled by law **DONE**
   6.3. Passwords shall not be viewable at any time **DONE**
   6.4. Passwords shall be encrypted **DONE**

7. **Coding Standards**
   7.1. JavaScript variables and functions shall be in camelCase **DONE**
   7.2. JavaScript functions shall state parameters and return value types **ON TRACK**

8. **Usability**
   8.1. The date/time format shall follow month/day/year **DONE**
   8.2. Text shall be a font size of 9 or larger **DONE**
   8.3. The application shall be available in the English language **DONE**
   8.4. The application shall be available in the US **DONE**

8.5. The application shall provide error message for wrong input **DONE**
8.6. Components shall follow the same design guideline **DONE**
8.7. Submit Buttons shall not be invoked within 300 ms after click **DONE**

9. **Accessibility**
9.1. All uploaded images must have an alternative text **DONE**
9.2. All content shall be accessible with screen readers **ON TRACK**
9.3. All content shall be navigable by keyboard **ON TRACK**
9.4. All heading level shall be in order **ON TRACK**
9.5. Data tables shall not have empty cells **DONE**
9.6. All content shall be readable with zoom to 200% **DONE**
9.7. All Colorset shall pass color contrast test **ON TRACK**

## 7. List of Contributions

1. **Jiji/Jijeong Lee (Front End Leader) - Score: 9**

    - message page both frontend and backend
    - work on my page functionality and updates
    - Worked on code used for peer review
    - Went to all meetings and participated

    - Worked on functional requirements
    - Worked on My Page frontend
    - Worked on message features(get message, post message) - both on frontend and backend
    - Worked on unique constraint for user email field
    - Worked on Saved Premium Guide list
    - Worked on Mentoring Session List
    - Worked on Premium guide page

2. **Mat/Matthew Bush (Back End Leader) - Score: 8**
    - Went to all meetings and participated
    - Worked on posts
    - Worked on functional requirements
    - Worked on Forum and Post pages backend
3. **Cris/Cristobal Padilla (Github Leader) - Score: 10**
    - Utilized Trello
    - Helped work on M3 corrections
    - Very communicative in between meetings
    - Went to all meetings and participated
    - Worked on local storage/cookies
    - Worked on functional requirements
    - Worked on registered users shall be able to change from Free to Premium subscription plan and vice versa
    - Worked on Add input validation to Posts search bar

- Display “no comments” on posts with no comments
- Add functionality to send message from User Profile popup

4. **Hayat/Khayotbek Azimov (Cloud Leader) - Score: 9**
    - Went to all meetings and participated
    - premium guide and rendering dynamically on the database
    - Created the usability table
    - Worked on functional requirements
    - Worked on registered users shall be able to change their username
    - Worked on registered users shall be able to add a bio to their personal information
    - Worked on registered users shall be able to edit their bio
    - Lead most of the work for the Usability plan
    - Worked on surveys for Usability with people
5. **Jef/Jeffrey Ma (Database Leader) - Score: 9**
    - Went to all meetings and participated
    - Worked on work on messages
    - Worked on uploading pictures and how to store them
    - Worked on Section #5
    - Worked on doing code review with classmates outside of team
    - Worked on functional requirements
    - Worked on the DB connecting with everything properly
    - Changing subscription plans (implement frontend)
    - Worked on surveys with people for usability
    - Updated all the tables

6. **Hector Magallanes (Document Leader) - Score: 10**
    - front end for posts with form that submits what we need without user info
    - Document editing and formatting
    - Worked on code for review
    - Went to all meetings and participated
    - Very communicative in between meetings
    - Adjusted forum language
    - Worked on functional requirements
    - Worked on History Versions Table
    - Worked on Table of Contents
    - Lead most of the work with Hayat for Usability
    - Worked on restrict Premium Guide access for non-premium users
    - Check how to store Posts/Guides body as markdown instead of plain text
7. **Emilee Padilla (Team Lead) - Score: 8**
    - Created a structure to follow on Trello for working efficiently
    - Updated our discord channels
    - Created and shared our documents
    - Created office hour for questions
    - Worked on Section #5
    - Made contribution list
    - Checked in with team mates for progress updates
    - Went to all meetings and participated
    - Created the logo

- Worked on editing the document
- Worked on usability testing with people
- Worked on functional requirements
- Created survey for User Satisfaction: Likert Questionnaire