

# SW Engineering CSC648-848-01 Summer 2023

## *Zenful*

Team 2 (Spider Computer Scientists)

Team Lead: Suzanne Heikal ([sheikal@mail.sfsu.edu](mailto:sheikal@mail.sfsu.edu))

Front-End Lead: Kimon Monokandilos

Back-End Lead: Braden Dalit

Github Master(s): Suzanne Heikal/Braden Dalit

Jada Campbell

Lucas Soto

Edward Chen

Milestone 4

07/27/23

History Table:

Milestone's Versions	Date
M4V1	07/27/23
M3V2	08/01/23
M3V1	07/24/23
M2V2	07/24/23
M2V1	06/28/23
M1V2	06/26/23
M1V1	06/12/23

## Table of Contents

<b>Product Summary.....</b>	<b>2</b>
<b>Usability Test Plan.....</b>	<b>4</b>
<b>QA Test Plan.....</b>	<b>7</b>
<b>Code Review.....</b>	<b>12</b>
<b>Best Practices for Security.....</b>	<b>16</b>
<b>Self-Check: Non-Functional Specs.....</b>	<b>17</b>
<b>List of Contributions.....</b>	<b>24</b>

## **Product Summary**

Our product is Zenful. It is a journaling app with added features, such as music, for a more personal and positive experience. The best way to market this product would be for us to utilize social media. Social media is a great tool for advertising products that have to do with productivity and an app like ours where you can make a journal into anything you need it to be is best promoted on social media. It would start with ads on Tik Tok. This is where you would find most of the users who are looking for something productivity related. The ad would include a sample of the templates we off as well as all the possibilities of the app. Then we would also do small skits that would get people interested. Some skits include someone wanting to organize their thoughts in one place that they could protect. Another way to market this product would be to market it on Instagram where people are influencers and would need to make journal content ideas, etc. Our main focus will be people who need to be able to write out what they need to get done but we should also focus on people who are going to do content because this is an app with the purpose of letting people decide what they need to write out in an electronic way that is secure.

### **Home Page**

1. The home page shall display step-by-step instructions to use Zenful.

### **General User**

1. General users shall be able to view the home page.
2. General users shall be able to register.

### **Registered User**

1. Registered users shall be able to login.
2. Registered users shall have access to Zenful once they are logged in.

### **Registration**

1. Users shall need to provide their email address during registration.
2. Users shall need to provide their username during registration.
3. Users shall need to provide their password during registration.

### **Login**

1. Registered users shall be able to login.
2. Registered users shall need to provide their username and password to login.
3. Registered users shall have the option to logout.

### **Admin**

1. The admin shall be able to access and modify the database.
2. The admin shall be provided with full data access.
3. The admin shall need to login before accessing admin permissions.

### **Journal**

1. Registered users shall be able to create a new private (accessible only to the registered user and no-one else) journal entry.
2. Registered users shall be able to see all of their existing private journal entries.

**Features**

1. Registered users shall be able to play public music through Zenful.
2. Registered users shall be able to pause currently playing music.

URL: <http://34.219.66.26/>

## Usability Test Plan

1. Journals are the main feature of Zenful, and the main purpose of usage, this along with other features, are to provide a better experience for writing than other sites.
  - a. Journal Creation: Users must be registered and logged in, access to tasks is on the Create Journal tab, journal must be created and saved.
  - b. Journal Viewing: Users must be registered and logged in, access to task is on the Saved Journals page, and journals must be loaded on screen.
  - c. Journal Management: Users must be registered and logged in, access to task is from a selected journal on the Saved Journals page, journal must be modified and saved.
  - d. Journal Deletion: Users must be registered and logged in, access to task is on the Saved Journals page, journal must be deleted and removed from record.
2. Music is being played in the background of the user's choosing, to keep them focused and motivated in the writing process, enhancing the experience.
  - a. Music Toggle: Users must be registered and logged in, task is in the user's account page, music must be either playing or disabled based on user setting.
  - b. Music Selection: Users must be registered and logged in, task is in the user's account page, music must be either playing the selected track based on user setting.
3. Animation serves as a simulant when compared to a static background, keeping users interested in the journaling process and in turn stay on the site.
  - a. Animation Toggle: Users must be registered and logged in, task is in the user's account page, animation must be either playing or not visible based on user setting.
  - b. Animation Selection: Users must be registered and logged in, task is in the user's account page, animation must be either playing the selected video based on user setting.
4. Users can upload music and animations to better fit their interests, and the customizability gives individual users a sense of value.
  - a. Upload Music: Users must be registered and logged in (TODO)
  - b. Upload Animation: Users must be registered and logged in (TODO)
5. Once a user has accumulated enough usage on Zenful, they will need a search function to find the specific journal entry they are looking for, or to use their own uploaded music or animation.
  - a. Date Search: Users must be registered and logged in, and have existing journal entries, task is available on the search bar on the top right, search results must return any journal entries created in the specified time, if any.
  - b. Title Search: Users must be registered and logged in, and have existing journal entries, task is available on the search bar on the top right, search results must return any journal entries with the specified title, if any.

- c. Content Search: Users must be registered and logged in, and have existing journal entries, a task is available on the search bar on the top right, search results must return any journal entries with the specified keywords in its content, if any.

### Usability Effectiveness

Major Functions	Successful Completion	Errors	Comments
Journals	90%	Unknown errors	Potential lack of feedback resulting in unaware completion?
Music	0%	NYI	NYI
Animation	0%	NYI	NYI
Upload	0%	NYI	NYI
Search	75%	Can not search by content or date	Lacks functionality to search by content and date

### Usability Efficiency

Major Functions	Average Time Used	Difficulty	Layout
Journals	01:45	None	Journal editing is basic, rich text editing could be implemented
Music	XX:XX	NYI	NYI
Animation	XX:XX	NYI	NYI
Upload	XX:XX	NYI	NYI

<b>Search</b>	<b>00:25</b>	<b>Search is small and missable to some</b>	<b>Search Bar needs more visibility and dropdown options</b>
---------------	--------------	---	--

# **QUESTIONNAIRE:**

<https://docs.google.com/forms/d/12SDDy8Zm-JNKMNsIP8p--L-R9HINp7jworVlhE0ZXvg/edit>

## QA Test Plan

### Test objectives:

System Requirements

### HW and SW setup:

HW: Macbook(macOS 13), Windows(11), Iphone 11

SW: Chrome(16), Safari(16)

URL: <http://34.219.66.26/>

**Feature to be tested** - Check Basic Loading Latency, Check Heavy Loading Latency, Check Upload Latency, Check Creation Latency, and Check Deletion Latency.

### QA Test plan:

System Requirements	Description	Test Input	Expectations	Result
#1	Check Basic Loading Latency	Load Created Journals with 0 journals.	Load completes within expected timeframe.	PASS
#2	Check Heavy Loading Latency	Load Created Journals with 100 journals	Load completes within expected timeframe.	PASS
#3	Check Upload Latency.	Upload music file and animation file of same size.	Upload completes within expected timeframe.	PASS
#4	Check Creation Latency	Create a new user account and journal entry from that account.	Creation completes within expected timeframe.	PASS
#5	Check Deletion Latency	Delete journal entries and then the user account	Deletion completes within expected timeframe.	FAIL



**Test objectives:**

Coding Standards

**HW and SW setup:**

HW: Macbook(macOS 13), Windows(11), Iphone 11

SW: Chrome(16), Safari(16)

URL: <http://34.219.66.26/>

**Feature to be tested** - Check Code Documentation, Check Code Warnings, Check Code Utilization, Check Database Utilization, Check Database Warnings

**QA Test plan:**

Coding Standards	Description	Test Input	Expectations	Result
#1	Check Code Documentation	N/A	Frontend and Backend is documented and understandable	PASS
#2	Check Code Warnings	N/A	No formatting errors detected	PASS
#3	Check Code Utilization	N/A	No redundant code utilized	PASS
#4	Check Database Utilization	N/A	No redundant tables utilized.	PASS
#5	Check Database Warnings	N/A	No query errors detected	PASS

**Test objectives:**

Security

**HW and SW setup:**

HW: Macbook(macOS 13), Windows(11), Iphone 11

SW: Chrome(16), Safari(16)

URL: <http://34.219.66.26/>

**Feature to be tested** - Access Unauthorized Journals, Access Journals without Account, Access Own User Profile, Access Own Journals, Access Deleted Journals

**QA Test plan:**

Security	Description	Test Input	Expectations	Result
#1	Access Unauthorized Journals	Load journals page with random ID while signed in.	Redirected to Created Journals	PASS
#2	Access Journals without Account	Load random journals as an guest	Redirected to Login Page	FAIL
#3	Access Own User Profile	Load user profile page while signed in.	Directed to own profile	PASS
#4	Access Own Journals	Load journals from the created journals page.	Directed to Journal	PASS
#5	Access Deleted Journals	Load journals from history after deletion.	Redirected to Created Journals	FAIL

**Test objectives:**

User Data

**HW and SW setup:**

HW: Macbook(macOS 13), Windows(11), Iphone 11

SW: Chrome(16), Safari(16)

URL: <http://34.219.66.26/>

**Feature to be tested** - Create Journal Entry, Upload Music, Upload Animation, Delete Journal, Delete User Account

**QA Test plan:**

User Data	Description	Test Input	Expectations	Result
#1	Create Journal Entry	Randomly generated 500 character text	Journal entry is created and accessible by the creator.	PASS
#2	Upload Music	music.mp3	Music is saved and usable by the uploader.	FAIL
#3	Upload Animation	animate.mp4	Animation is saved and usable by the uploader	FAIL
#4	Delete Journal	N/A	Journal entry is deleted and deallocated.	FAIL
#5	Delete User Account	N/A	User account is deleted and all associated data i.e. Journals/Music/ Animation	PASS

**Test objectives:**

Environmental

**HW and SW setup:**

HW: Macbook(macOS 13), Windows(11), Iphone 11

SW: Chrome(16), Safari(16)

URL: <http://34.219.66.26/>

**Feature to be tested** - Check Website Scaling, Check Browser Compatibility, Check Mobile Compatibility, Check Upload Verification, Check Journal Verification

Environmental	Description	Test Input	Expectations	Result
#1	Check Website Scaling	N/A	Site scales proportionate to display size	PASS
#2	Check Browser Compatibility	N/A	Site loads without issue	PASS
#3	Check Mobile Compatibility	N/A	Site loads without issue	PASS
#4	Check Upload Verification	Upload invalid formats such as music and animation.	Site rejects invalid upload files	PASS
#5	Check Journal Verification	Create new and modify existing journal entries with invalid texts.	Site rejects invalid characters in journals	PASS

## Code Review

Our coding style utilizes the Prettier extension in VSCode to maintain a consistent syntax. Prettier syntax has its own set of rules that include the following: Egyptian brackets, minimal horizontal line lengths by splitting lines up where possible, 2 space indents, vertical indents for logical blocks, functions declared above uses, and else without line breaks. Prettier enforces the majority of these rules by itself, while things such as functions declared above the use of said functions, are a choice by us as a team.

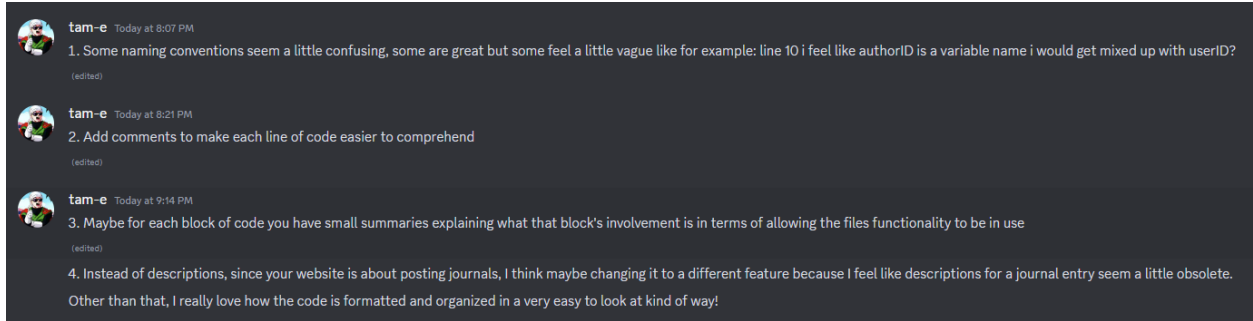
Review sent to other team:

```
1  var db = require("../config/db");
2  const { successPrint } = require("../helpers/debug/debugprinters");
3
4  const PostModel = {};
5
6  PostModel.create = ({title, description, fk_authorId}) => {
7    let baseSQL =
8      "INSERT INTO journals (title, description, created, fk_authorId) VALUE (?, ?, now(), ?);";
9    return db
10     .execute(baseSQL, [title, description, fk_authorId])
11     .then(([results, fields]) => {
12       return Promise.resolve(results && results.affectedRows);
13     })
14     .catch((err) => Promise.reject(err));
15  };
16
17  PostModel.getNRecentPosts = ({numberOfPosts}) => {
18    let baseSQL = `SELECT id, title, description, created FROM journals ORDER BY created DESC LIMIT 16`;
19    return db
20     .execute(baseSQL, [numberOfPosts])
21     .then(([results, fields]) => {
22       return Promise.resolve(results);
23     })
24     .catch((err) => Promise.reject(err));
25  };
26
27  PostModel.getJournalById = ({journalId}) => {
28    let baseSQL = `SELECT u.username, j.title, j.description, j.created
29    FROM users u
30    JOIN journals j
31    ON u.id=j.fk_authorId
32    WHERE j.id=?`;
33
34    return db
35     .execute(baseSQL, [journalId])
36     .then(([results, fields]) => {
37       return Promise.resolve(results);
38     })
39     .catch((err) => Promise.reject(err));
40  };
41
```

```

1  var express = require("express");
2  var router = express.Router();
3  const { errorPrint, successPrint } = require("../helpers/debug/debugprinters");
4  var multer = require("multer");
5  var crypto = require("crypto");
6  var PostError = require("../helpers/error/PostError");
7  var PostModel = require("../models/Posts");
8
9  var storage = multer.diskStorage({
10     destination: function (req, file, cb) {
11         cb(null, "public/images/uploads");
12     },
13     filename: function (req, file, cb) {
14         let fileExt = file.mimetype.split("/")[1];
15         let RandomName = crypto.randomBytes(22).toString("hex");
16         cb(null, `${RandomName}.${fileExt}`);
17     },
18 });
19
20 var uploader = multer({ storage: storage });
21
22 router.post("/createPost", uploader.single("chooseImage"), (req, res, next) => {
23     let title = req.body.title;
24     let description = req.body.description;
25     let fk_userId = req.session.userId;
26
27     return PostModel.create(
28         title,
29         description,
30         fk_userId
31     )
32     .then((postWasCreated) => {
33         if (postWasCreated) {
34             successPrint("if true");
35             req.flash("success", "Your post was created successfully!");
36             res.redirect("/");
37         } else {
38             resp.json({
39                 status: "OK",
40                 message: "post was not created",
41                 redirect: "/createjournal",
42             });

```



I chose this code for review because we as a team want to strengthen our functionality with journals as it is one of our main features with our app. And with this review we can apply the suggestions to the rest of our code.

Team member's code for review:

In the code that was sent, I do think that we need to add more comments because of other people in this team looking at the code. This is because we might add things, and not everyone on this team has knowledge on what that certain code does. This way, they will be able to add their own code and not be as confused. I also believe that we should add better naming conventions too because sometimes “post” might not be as clear as “journal”.

```

1  var db = require("../config/db");
2
3  const MusicModel = {};
4
5  MusicModel.create = (title, description, fk_authorId) => {
6    let baseSQL =
7      "INSERT INTO journals (title, description, created, fk_authorId) VALUE (?, ?, now(), ?);";
8    return db
9      .execute(baseSQL, [title, description, fk_authorId])
10     .then(([results, fields]) => {
11       return Promise.resolve(results && results.affectedRows);
12     })
13     .catch((err) => Promise.reject(err));
14  };
15
16  MusicModel.search = (searchTerm, userId) => {
17    // created FROM journals WHERE fk_authorId = ? AND (title LIKE ? OR description LIKE ?);";
18
19    let baseSQL =
20      "SELECT id, title, description, concat_ws(' ', title, description) AS haystack \
21      FROM journals \
22      having haystack like ?;";
23    let sqlReadySearchTerm = "%" + searchTerm + "%";
24    return db
25      .execute(baseSQL, [userId, sqlReadySearchTerm])
26      .then(([results, fields]) => {
27        return Promise.resolve(results);
28      })
29      .catch((err) => Promise.reject(err));
30  };
31
32  MusicModel.getMusicById = (journalId) => {
33    let baseSQL = `SELECT u.username, j.title, j.description, j.created
34    FROM users u
35    JOIN music m
36    ON u.id=j.fk_authorId
37    WHERE j.id=?`;
38
39    return db
40      .execute(baseSQL, [journalId])
41      .then(([results, fields]) => {
42        return Promise.resolve(results);
43      })
44      .catch((err) => Promise.reject(err));
45  };
46

```

Review notes:

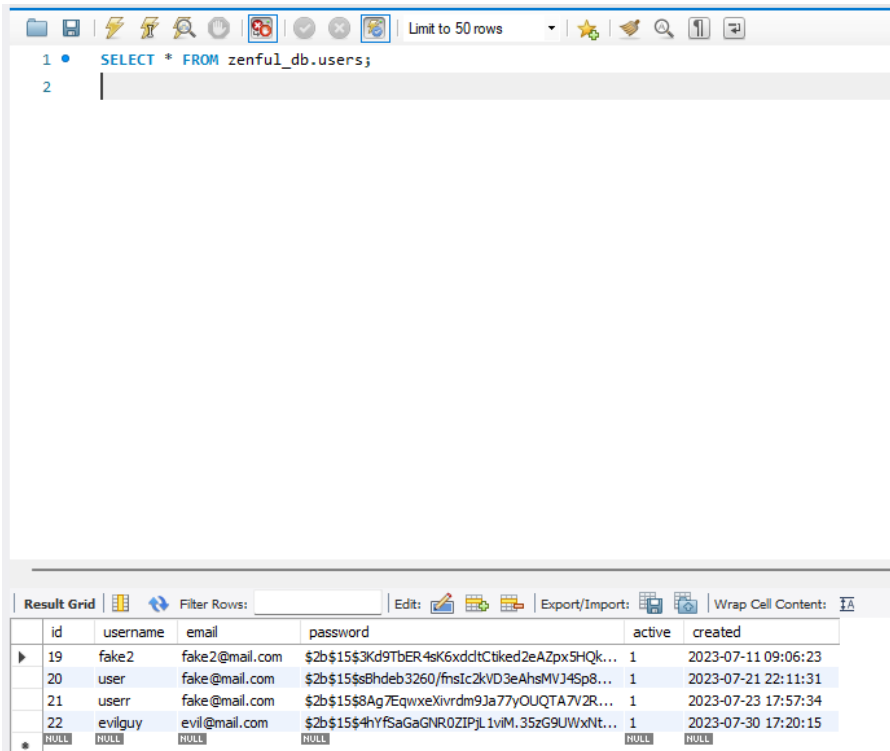
- Need to add comments for functionality so that readers can comprehend them.
- SQL statements are referencing the journals table in certain cases when they should be referencing the music table.
- Last change is that I would change `fk_authorId` to `fk_userId` as the user poster may not be the author



## Best Practices for Security

### Assets To Protect:

- Passwords
- Journals (People could put very personal or identifiable information there)
- Database (Using Bcrypt, when a user is creating an account the password is automatically hashed when being inserted into the database).



The screenshot shows a database management interface. At the top, there's a toolbar with various icons and a dropdown menu set to 'Limit to 50 rows'. Below the toolbar, a SQL query is entered in a text area: `1 • SELECT * FROM zenful_db.users;` and `2` is on the next line. Below the query area, a 'Result Grid' is displayed. It has a toolbar with icons for 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The grid contains the following data:

	id	username	email	password	active	created
▶	19	fake2	fake2@mail.com	\$2b\$15\$3Kd9TbER4sK6xddtCtked2eAZpx5HQk...	1	2023-07-11 09:06:23
	20	user	fake@mail.com	\$2b\$15\$8hdeb3260/fnsIc2kVD3eAhsMVJ4Sp8...	1	2023-07-21 22:11:31
	21	userr	fake@mail.com	\$2b\$15\$8Ag7EqwxeXivrdm9Ja77yOUQTA7V2R...	1	2023-07-23 17:57:34
	22	evilguy	evil@mail.com	\$2b\$15\$4YfSaGaGNR0ZiPJL1viM.35zG9UWxNt...	1	2023-07-30 17:20:15
•	NULL	NULL	NULL	NULL	NULL	NULL

### Data Validation Include:

- Username when registering (Regex: not digit, is alphanumeric character, min of 2 characters long) `/^[\D\w]{2,}$`
- Email exists when registering
- Password exists when registering (No regex as of yet, password is encrypted when saved to Database using bcrypt)
- Username and Password are of an existing user when logging in and that they match.
- User must be logged in before they can create journal entries.
- Search: Checks if there are journal entries that exist to search, contents of search match a journal entry (Date, Title).

## Self-Check: Non-Functional Specs

### 1. System Requirements

- 1.1. Web app should be available on personal computers and laptops. **DONE**
- 1.2. Web app should be able to service a few thousand clients. **DONE**
- 1.3. Will take about 100 GB of bandwidth a month. **DONE**
- 1.6. Web app will be easy for anyone to use. **ON TRACK**
- 1.7. If there is a delay on using site user should only experience a few seconds delay. **DONE**
- 1.8. There should be enough storage for users to upload music to the website. **ON TRACK**
- 1.9. There should be enough storage for users to have hundreds of notes and or journal pages. **ON TRACK**
- 1.12. If there is maintenance there will be a banner on the website for all to see. **DONE**

### 2. Coding Standards

- 2.1. All code is written in JavaScript. **DONE**
- 2.2 All code will have notes attached for easy maintenance on scheduled days. **ON TRACK**

### 3. Content and Appearance

- 3.1. Landing page should have a neutral background. **ON TRACK**
- 3.2. The forefront of the page should be a highlight reel for the sites features. **ON TRACK**
- 3.3. There will be the site name and or logo in left hand upper corner. **DONE**
- 3.4. Middle top will have maintenance schedule. **DONE**
- 3.5. The upper right hand side should have the login/signup option. **ISSUE: We decided to have login/signup on the left sidebar of our web-app.**
- 3.6. The highlight reel will be in the middle of the page. **ON TRACK**

3.7. Site will have a neutral and cool toned theme for the website until user chooses their own theme for their usability. **ON TRACK**

3.8. There will also be a setting for light or dark. **ON TRACK**

3.9. There will be an option for logging in on the upper right hand side of the page with links to account settings. **ISSUE: We decided to have login/signup on the left sidebar of our web-app.**

3.10. Under the logo will be the header on the left hand side: Notes, Journals, Music, Avatar, Themes, Account. **ON TRACK**

3.11. On the bottom of the page shall be links to about us, faqs, contact us, terms of use, privacy policy. **ON TRACK**

3.12. The side should have a menu option to flip between the music section and the current page. **ON TRACK**

3.13. The companies information: name, address, establishment, last time site was updated should be on the bottom of the page. **ON TRACK**

3.14. Above companies information should be the Faq, About Us, Contact Us, Terms of use, Privacy Policy. **ON TRACK**

3.15. When in the notes tab, there will be title of note, date, saving function on the left hand side. **DONE**

3.16. The title and date will be larger than the saving function. **ISSUE: Note is saved automatically, no need for a saving button.**

3.17. Under the title of the note will be the file options, edit, insert, format, and help. **ON TRACK**

3.18. File options, edit, insert, format, and help should be slightly smaller than the title and date. **ON TRACK**

3.19. Under file options should be the Font, Zoom, Text Style, Text Size, Highlighter, Template, and Print function. **ISSUE: We may not be able to implement so many advanced features, but we will try our best.**

3.20. On the left hand side there will be a area where the titles of notes are listed. **ISSUE: We may not be able to implement this feature, but we will try our best.**

3.21. On the right hand side there will be the Account function. **ISSUE: We decided to have all functions on the left sidebar.**

3.22. The header from the dashboard will be visible in the middle of the top of the page. **DONE**

3.23. At the bottom of the page is the links to About Us, Faqs, Contact Us, Terms of Use, Privacy Policy. The company information and last update of site/feature. **ON TRACK**

3.24. In the bottom right should be the animated gif/animation. **ON TRACK**

3.25. The music player should be in the bottom left. **ISSUE: Upon discussion we decided to have the music player at the bottom rather than bottom left.**

3.26. The background should be user pick and in the foreground should be the 'paper' the user is typing on. **ISSUE: Background needs to be picked by our software engineering team rather than the user.**

3.27. When in the journaling tab, there will be a date at the top left hand side. **ON TRACK**

3.28. There will be the choice to add a title or leave just the date. **ON TRACK**

3.29. There will be the files, insert, format, edit, and help functions under the date. **ISSUE: We are currently implementing edit, may not have the time to implement files, insert, format, and help functions.**

3.30. The date should be larger than the files, insert, format, edit, and help functions. **ISSUE: We are currently implementing edit, may not have the time to implement files, insert, format, and help functions.**

3.31. Under files, insert, format, edit, and help should be font, font size, zoom, and print function. **ISSUE: We may not have the time to implement all of these advanced features, but we will try our best.**

3.32. On the right hand side should be the account function as well as the passcode toggle. **ISSUE: We decided to have all of our functions on the left sidebar.**

3.33. At the bottom of the page should be the About Us, Faqs, Contact Us, Terms of Use, Privacy Policy, Company Information and last update of site/feature. **ON TRACK**

3.34. In the bottom right should be the animated gif/animation. **ON TRACK**

3.35. The music player should be in the bottom left of the screen. **ISSUE: Upon discussion we decided to have the music player at the bottom rather than bottom left.**

3.36. The background should be the users pick. **ISSUE: Background needs to be picked by our software engineering team rather than the user.**

3.37. In the foreground should be the 'paper' the user is typing on. **ISSUE: Currently we just have a form for the user to type into rather than a paper background image.**

3.38. When in the music tab there should be a mini player in the top left hand side. **ISSUE: Upon discussion we decided to have the music player at the bottom rather than bottom left.**

3.39. It should have a play, pause, and back and forth function. **ON TRACK**

3.40. On the top left under the player should be the functions: Playlist(s), Browse, Just for You, Trending. **ISSUE: We may not have the time to implement these advanced features.**

3.41. On the main page should be the pages that are toggled. **DONE**

3.42. The page should show the background that the user has chosen. **ISSUE: Background needs to be picked by our software engineering team rather than the user.**

3.43. At the bottom of the page should be the About Us, Faqs, Contact Us, Terms of Use, Privacy Policy, Company Information and last update of site/feature. **ON TRACK**

3.44. In the right hand side should be the account function in the upper right. **ISSUE: We decided to have all of our functions in the left sidebar.**

3.45. In the bottom right should be the animated gif/animation. **ON TRACK**

#### 4. Security

4.1 The user will need to login in order to access any of the files on the server. **DONE**

4.2. This will enable them to ensure their information is kept private and not open for strangers to engage with their work. **DONE**

4.3. When toggled on the user will be prompted to enter a four number code for their passcode. **ISSUE: We may not have the time to implement this functionality.**

4.4.Users data should be stored on site storage. **DONE**

## 5. Marketing

5.1. To market this product we will provide information on the key aspects of the site such as the ability to have a little animated friend that will help keep users motivated to use the app. **DONE**

5.2.We will also tote about how you can add your music to the library and ensure that they can customize the page in preset themes. **ISSUE: We are currently working on adding music functionality, however, may not have the time to have the page be customizable with preset themes.**

5.3.The big point is that the app is able to have music, an animated gif or animation in the corner, customizable page themes, journal and note taking and the ability to passcode notes. **ISSUE: Music (on track), animation (on track), customizable page themes (may not have the time), journal (on track), passcode (may not have the time).**

## 6. User Data

6.1 User data that will be collected is as follows: music genres liked by user, user age, user gender. **ISSUE: We are saving usernames, emails, and passwords only for the user entity.**

6.2. This information is gathered to help further the understanding of what the users enjoy and what can be added in the future for more user engagement. **ISSUE: We are saving usernames, emails, and passwords only for the user entity.**

6.3.This information is only used for the furthering of user enjoyment and is not sold to any outside parties. **ISSUE: We are saving usernames, emails, and passwords only for the user entity.**

6.4.User information is tracked through cookies. **ON TRACK**

6.5. User's have all rights to their privacy information and can opt to decline accepting trackers. **ISSUE: Users have a checkbox for Terms of Service and Privacy Policy. They can either accept it and create their account, or decline it, and not create an account.**

6.6. If a user decides to decline having their information stored, their information will be placed on a list of do not track. **ISSUE: Users have a checkbox for Terms of Service and Privacy Policy. They can either accept it and create their account, or decline it, and not create an account.**

6.7. Any change to the privacy policy will be sent out in an email to all users for them to accept again and or decline. **ISSUE: This functionality is important, but might need to be implemented in the future, as we are presently preoccupied with urgent functionalities.**

6.8. As well as a notice placed on the website header. **ON TRACK**

6.9. Last update to privacy policy will be placed at the bottom of the page. **ISSUE: This functionality is important, but might need to be implemented in the future, as we are presently preoccupied with urgent functionalities.**

6.10. The user has the right to request a copy of what information we have collected from them at any time. **ISSUE: This functionality is important, but might need to be implemented in the future, as we are presently preoccupied with urgent functionalities.**

6.11. Registered users will be able to accept or decline terms of conditions upon registering. **DONE**

6.12. Users will have the right to retain the information and usage data. **DONE**

6.13. User shall have access to privacy policy upon registration **ON TRACK**

6.14. User will have access to privacy policy in a link at the footer of the page. **ON TRACK**

## 7. Performance and Scalability

7.1. Web app will be reliable at all times and few crashes. **DONE**

7.2. There will be a fast and efficient switch between page options. **DONE**

7.3. Web app should be able to handle 10 times its capacity. **DONE**

7.4. The website should be up at all times unless there is maintenance. **DONE**

7.5. Peak loads are expected to be about 5000 users. **DONE**

## 8. Environmental

8.1. The environment should be compatible with web browsers of all kinds. **DONE**

8.2. The environment should support a filter with dates for journal/to-do list **ON TRACK**

8.3. The environment should support a filter with animation/sounds **ON TRACK**

8.4. The environment should support a stable internet connection. **DONE**



### List of Contributions

Name	Contribution	Score
Braden Dalit	-Music functionalities -Best Practices for Security	2
Kimón Monokandilos	-Self-Check: Non-Functional Specs -Edited doc	2
Jada Campbell	-Product summary -Edited doc	2
Lucas Soto	-Code review -Edited doc	2
Edward Chen	-Usability test plan -QA test plan	2

**Emails: Sent**