Mindscribe Project Report

Project Title: Mindscribe

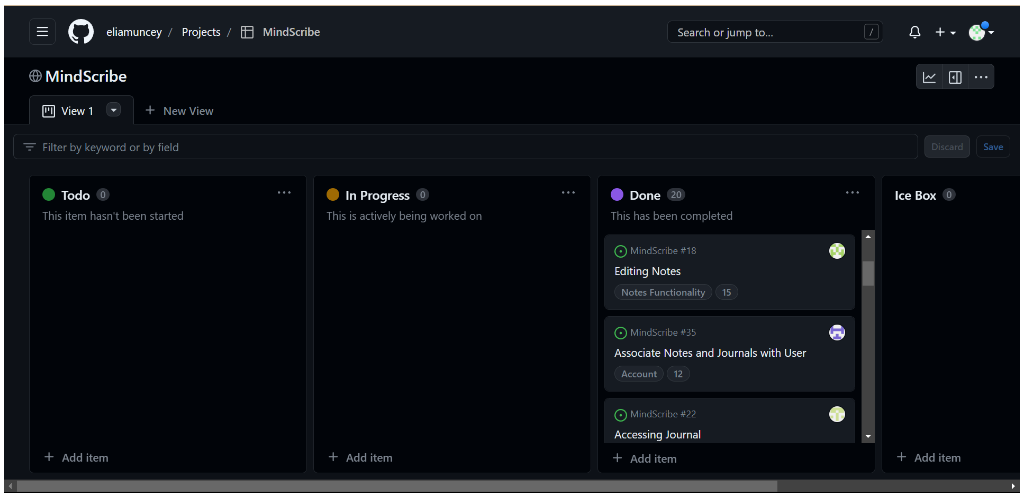
Group Members: John Danekind, Blake DeHaas, Ryan Garrett, Elia Muncey, Jared Roberts

Project Summary:

MindScribe is a smart journaling project that utilizes ChatGPT to help users enhance their mindfulness and improve their journaling experience. With the goal of making journaling more accessible and efficient, MindScribe streamlines the process of revisiting journal entries, while also providing valuable insights into users' moods and emotions. Users can create customized journals complete with a cover theme of their choice. They are then able to write notes freely, without worrying about grammar or formatting. Once a note is written, users can choose to format the note, which corrects any errors and formats the content for readability. The summarize function enables users to get an overview of an entry without having to read through it in its entirety. Mindscribe also has smart mood analysis for notes. By enabling automatic mood analysis for a journal, users can gain a deeper understanding of their emotional states and observe their mood patterns over time. MindScribe is a game-changing project that makes journaling more efficient and effective for users by leveraging the power of artificial intelligence. With its advanced features and user-friendly interface, MindScribe is the perfect tool for anyone who wants to enhance their mindfulness and improve their mental well-being.

<u>Project Tracker</u>

Link: https://github.com/users/eliamuncey/projects/1/views/1


eliamuncey / Projects / MindScribe
Search or jump to...
MindScribe
View 1
New View
Filter by keyword or by field
Discard
Save
Add item

Todo 0
This item hasn't been started

In Progress 0
This is actively being worked on

Done 20
This has been completed

MindScribe #18
Editing Notes
Notes Functionality 15

MindScribe #35
Associate Notes and Journals with User
Account 12

MindScribe #22
Accessing Journal

Ice Box 0

Video:

https://youtu.be/QCUvOopNNic

Version Control System

Link: https://github.com/eliamuncey/MindScribe

Individual Contributions:

John - For the project, I implemented functionality into the notes and journal pages. This functionality allows for editing of notes, saving once again, and deleting notes. In addition, all of the functionality and features mentioned above was implemented for the journals as well. Originally, before the ChatGPT API was implemented, I set up the first iteration of sentiment analysis by allowing the user to pick an emoji from a dropdown that would serve as the "mood" for the note. However, we decided later that we wanted the sentiment analysis to be handled entirely by ChatGPT so this code was refactored. While I was waiting for the ChatGPT API section to be completed, I worked on UI features in the journal and notes pages. This included designing logos for all the buttons, ensuring that each note was the same height and width, and adding a feature that would provide a preview of the note. Lastly, I made our entire PowerPoint presentation and demo video.

Blake - For my contribution, I implemented creating a new note and journal, saving a note and journal, displaying notes and journals, associating a note to a journal, and opening a note. It was very satisfying to see the development of the core functionality of the project from an idea to a fully working implementation. I also worked with Jared on the ChatGPT API and I was able to get the API working. Once implemented, I built the format and summarize smart features. I also focused on leading and organizing the team by creating the project board and release notes.

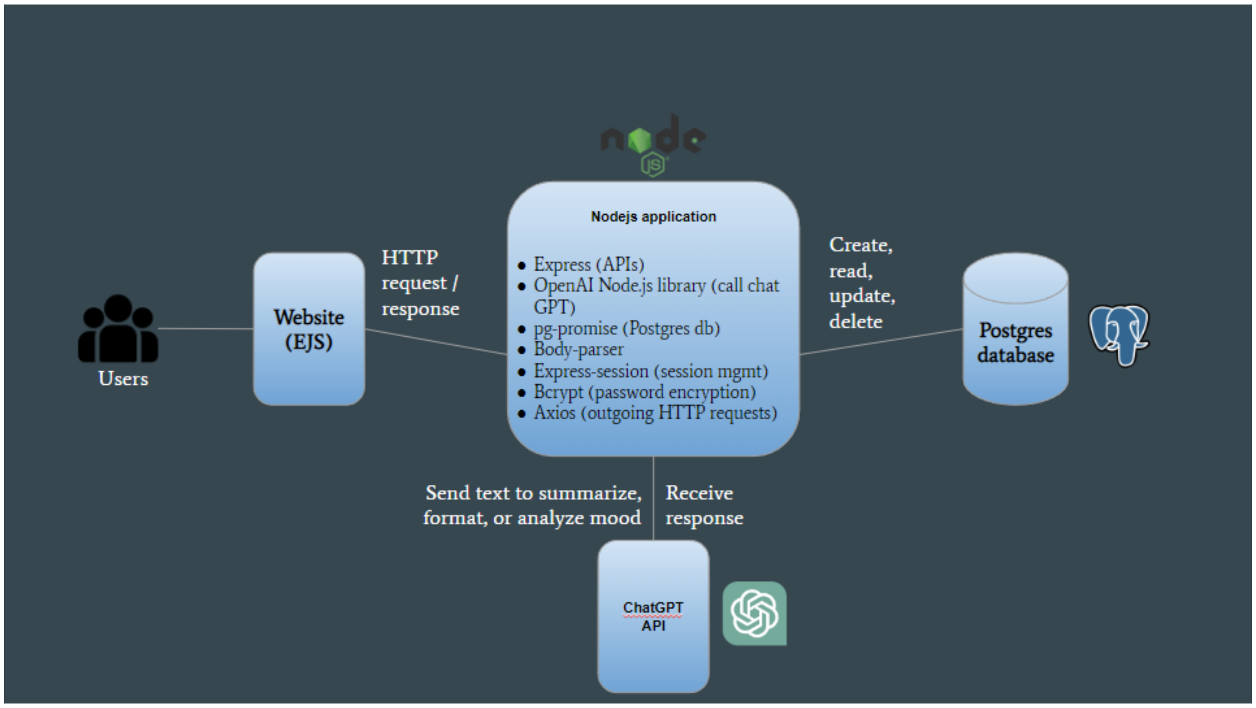
Ryan - My main area of responsibility for the project was related to the journal page functionality. Having journals render when the page was loaded, having those journal cards be formatted correctly, having the entirety of those cards be clickable for user friendly design, customizability of the journal colors as selected by the user, and more were all features that I implemented into our final project. I was also in charge of informational security. I created safeguards that prevented users from accessing the private info of other users, which was vital considering the personal nature of journals.

Elia - I created the SQL databases used throughout the project. I associated notes and journals with users, and did the automatic mood analysis. I also did the majority of the design for the user interface. I made the launch, profile, and home pages and formatted every other page to have the same design elements. On the profile page I added a feature to change username and password. On the home page, I got data for the number of journals, notes, words, and characters a user wrote, as well as implementing a welcome message that changes based on the time of day.

Jared - For my contribution I focused on the ChatGPT API calls, the main selling point of our project. I spent lots of time researching and watching videos on how to correctly set up the OpenAI API and send data using a given prompt. I set up the buttons to format the given text from a notes entry and then added the functionality of the button to call a function and in the index.js file that called the API. Through lots of trial and error and collaboration with Blake, we finally correctly connected to the API. With this done, we used the code to make smart features by giving the API different prompts. Before the presentation, I also implemented some UI changes with the footers and headers. I was also in charge of the deployment of the project to Azure and hosting it for the presentation.

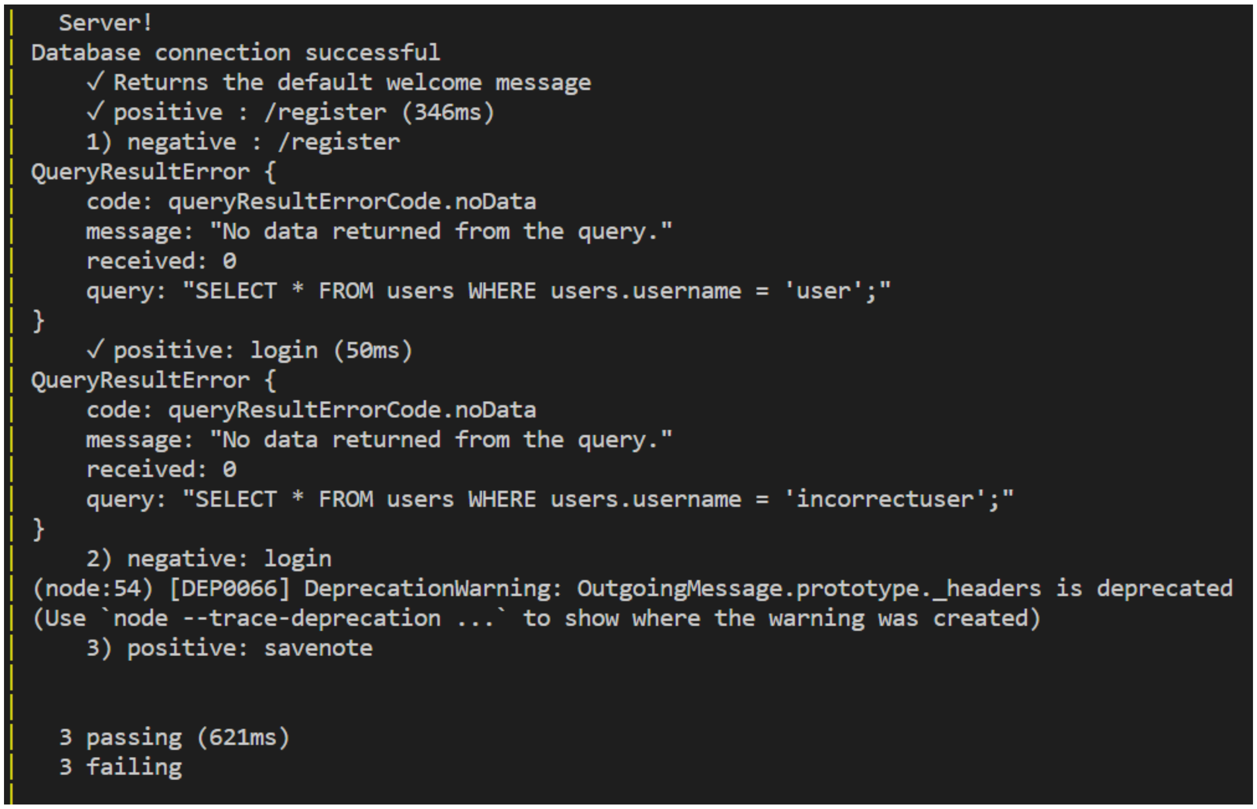
Use Case Diagram:

This diagram was created for our presentation and has been included as is:


Users
Website (EJS)
HTTP request / response
Nodejs application
Express (APIs)
OpenAI Node.js library (call chat GPT)
pg-promise (Postgres db)
Body-parser
Express-session (session mgmt)
Bcrypt (password encryption)
Axios (outgoing HTTP requests)
Create, read, update, delete
Postgres database
Send text to summarize, format, or analyze mood
Receive response
ChatGPT API

Test Results:

For testing, two methods were used. The first method was automated testing using Chai and Mocha libraries. This was used to test the register, login, and createnewnote endpoints. As we developed our project and changed the functionality of these endpoints, we transitioned to User Acceptance Testing with the software development team acting as the users and manually testing the code. This avoided having to re-write test cases as the functionality changed. Below is a screenshot of the current automated test cases. As the features were changed, three of the tests that succeeded previously are now failing.



```
  Server!
Database connection successful
    ✓ Returns the default welcome message
    ✓ positive : /register (346ms)
    1) negative : /register
QueryResultError {
    code: queryResultErrorCode.noData
    message: "No data returned from the query."
    received: 0
    query: "SELECT * FROM users WHERE users.username = 'user';"
}
    ✓ positive: login (50ms)
QueryResultError {
    code: queryResultErrorCode.noData
    message: "No data returned from the query."
    received: 0
    query: "SELECT * FROM users WHERE users.username = 'incorrectuser';"
}
    2) negative: login
(node:54) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
    3) positive: savenote


  3 passing (621ms)
  3 failing
```

These breakages led us to transition to user acceptance testing. For user acceptance testing, we used the following criteria:

- Does the feature perform its intended task successfully (if not, fix feature)

- Do all parts of the feature work as intended (if not, fix or cut out those parts)

- Does the feature interact properly with what is already developed in the project? (if not, prioritize what is already developed in the project as other features are depending on that)

- Is there a condition in place if the feature fails that informs the user? (if not, implement)

If the test meets all of these criteria, then the test will be deemed successful and the feature will be ready for deployment.

Using these criteria, we observed that the vast majority of our features were successfully implemented. We set guidelines that every feature must meet the criteria above before being pushed to the main branch of code and this allowed our team to avoid conflicts throughout most of the development process.

Near the end, features were pushed onto the main branch that did not go through full testing before implementation due to a lack of time. These changes ended up breaking certain aspects of the project. This led to the team observing the value of fully testing features before deployment. To solve these issues, we performed integration testing as a group in-person before our presentation to resolve all the issues that were present. We tested the project on multiple computers to see what parts of the project were not working as expected, then we were able to fix the code and get the project in a working state.

Deployment:

The project was deployed successfully to the Microsoft Azure cloud. We accomplished this by creating a student Azure account, creating a Linux virtual machine, setting up a DNS name, and then deploying the project to the virtual machine using Docker where the app would be hosted . We were able to access the project through the Azure cloud to give our live demo. Currently, the project is not being hosted on the Azure cloud in order to preserve the deployment hours our team was allotted. Below is a link to where we hosted the project.

Link: http://recitation-015-team-4.eastus.cloudproject.azure.com:3000/

To run the project locally, you can follow these steps:

- Start by cloning the github repository. This will get all the code of the project.
- Create a .env file that has the following content:

```
# database credentials

POSTGRES_USER="postgres"

POSTGRES_PASSWORD="pwd"

POSTGRES_DB="users_db"


# Node vars

SESSION_SECRET="super duper secret!"

OPENAI_API_KEY="Your Key Here"
```

- Navigate to the OpenAI API homepage here:

  https://openai.com/blog/openai-api
- Sign up and create a new OpenAI api key
- Replace Your Key Here above in the .env file with your OpenAI api key
- Download docker
- Navigate to ..\MindScribe\All Project Code and Components
- Run the command

  `docker-compose up`

  to start a local environment to run the project
- Navigate to localhost:3000 and you should be greeted with the welcome page