Ashvin Loghashankar, Erick Rosas Gonzalez, Kaitlyn Srinivasan, Sragvi Vadali, Xander Neary
Discussion 1C

Reflect Hub

App Purpose: Our web application, ReflectHub, is an application designed to support mental health and promote overall well-being through personalized journaling and thoughtful interaction. Our platform offers a space for users to document their current mental state through journal entries, allowing users to reflect on their mental health journey and track changes over time. It also encourages a sense of community by providing a platform for users to connect and share their experiences via a shared chat room. Through these interactions, users can find support, empathy, and a sense of belonging in their mental health journey. Lastly, the application integrates Chat Generative Pre-trained Transformers (Chat-GPT), allowing users to receive valuable insights, supportive responses, and advice as they share their emotions and concerns.

The target audience is individuals suffering from mental health issues. This includes students managing academic workloads, professionals balancing work-life demands, individuals coping with mental health issues such as anxiety, depression, or eating disorders, as well as others interested in fostering a positive and resilient mindset.

Overview of Architecture and Technologies

Server/Client Model: The codebase adopts a Client-Server architecture implemented in JavaScript and Node.js, with the client and server residing on a single computer and operating on distinct ports (3000 for the client and 3001 for the server). Utilizing node modules—Express for server-side development, Cors for handling cross-origin resource sharing, and Axios for streamlined HTTP requests—the system follows a straightforward GET and POST interaction pattern. The server receives data from the client, performs operations, and returns the processed data.

Firebase: The backend database is powered by Firebase DBMS, encompassing user authentication data and individual user profiles. Firebase's authentication API is employed for user authentication, utilizing a straightforward email/password model. User-specific information is stored in Firebase's Firestore database, and to ensure data security, we implement a custom encryption model before transmitting the data to users. This encryption model adds an extra layer of protection, preventing unauthorized access to user data directly from the database. Each user's profile includes their name, email, and a log of journal entries, maintaining a comprehensive record while prioritizing data privacy.

Langchain: Langchain is utilized to train the gpt-model. While utilizing OpenAI’s gpt-3.5-turbo base model as a foundation, we create context awareness to this model with langchain. Langchain reads data from a csv file with mental health training data, creating embeddings that the gpt model can use for context. This allowed our chatbot to be utilized mainly for mental health data, being more proficient than the base gpt-3.5-turbo model.

Overview of Distinct Features

Journal Entries: The journal entries feature provides users with a platform to articulate their emotions and thoughts at specific moments. Users can effortlessly create new journal entries by inputting a title and the corresponding entry content. The system securely stores this information in a database, associating it with the user's unique profile. Upon accessing the journal section, users can seamlessly navigate through their previous entries, presented as interactive boxes. Clicking on a past entry opens a new page, where

both the title and detailed entry content are displayed, enabling users to revisit and reflect on their recorded thoughts.

Support Group: Support Group acts as a forum that users can chat on anonymously together. When the user logs in and views the support group page, they can see any messages sent by other users (displayed on the left side of the screen with a white text box) and messages that they sent (displayed on the right side of the screen with a blue text box). All messages are stored in the Firebase database and the displayed messages are updated in real time (ten times a second). Typing into the chat box, the authenticated user can send a new message from their machine by pressing the "send" button. Whatever content was in the chat box is sent to the server, stored in Firebase, displayed on the screen, and the chat box is emptied.

Chatbot: The Chatbot serves as an intermediary between users and a trained mental health model, offering prompt mental health advice akin to a basic conversation with a psychologist. The underlying GPT-model is trained using data derived from an NLP research paper on mental health, facilitated by Langchain. The chatbot feature implements a turn-style communication approach, resembling the interaction style seen in ChatGPT. Users input prompts, and the GPT-model generates and displays its output in response.

Individual Contributions

Ashvin (18%): I mostly worked on the backend with Sragvi and Kaitlyn. I contributed to setting up the backend functionality for the journal. I connected the journal to the Firebase database, and added functions to add entries and fetch entries to and from the database. I then worked with Erick to ensure that these server-side actions would be usable by the frontend of the application. I also wrote the frontend for the chatbot, and was also able to work on the integration for the chatbot where I worked on the connection allowing the client to make requests to the server. I was able to create a working frontend that integrated server calls to allow the user to get responses from the Langchain model. I worked with Sragvi on this as he had written most of the backend for the chatbot.

Erick (22%): I mostly worked on the client side with Xander. I set up the landing page, the journal and sign-in/sign-up page, setting up the css for the pages. I set up the routing to make it easier to flip between pages and access data. I modularized the client-side in order to allow for cleaner code and make it easy to update. I collaborated with Sragvi in order to figure out how to send POST requests to the server to be able to sign in and sign up. When I signed in, I made it so components, like the landing page and header changed dynamically, so a user was allowed to access other pages. For the journaling page, I worked with Ashvin and Sragvi. For the main integration, I worked with Ashvin to send POST request to the server so a user can access all journal entries, view past entries, and create new entries. With Sragvi, we found ways to display these features to the user and make it user friendly and navigate between pages.

Kaitlyn (18%): I worked primarily with Ashvin and Sragvi on the backend. I contributed to setting up our authentication functionality, by writing functions to handle sign in and sign out, as well as connecting this authentication to our Firebase database. I also set up the backend for the messaging functionality used in our Support Group feature, storing messages with information such as timestamp and the user who sent the message so that Xander could integrate this information into the design of the Support Group page. For my contributions to the frontend, I implemented the sign out feature so that clicking the "sign out"

button sends a sign out request to the server. I also created notifications to the user when their sign in attempt fails, perhaps due to an invalid email, an invalid password, or too many failed authentication attempts.

Sragvi (22%): I worked mainly on the backend with Ashvin and Kaitlyn. We worked on making the database model and implementing functionalities for data retrieval and storage. I contributed to the chatbot feature by integrating the Langchain model with the OpenAI API. Additionally, I collaborated with Xander and Erick on enabling dynamic interactions between the frontend and backend, utilizing POST and GET methods. A significant part of my work in frontend involved caching userId data in local storage, ensuring utilization by other functions for backend operations. I also worked on helping with small bugs, helping Kaitlyn and Ashvin with their backend features as well as Xander and Erick with frontend bugs.

Xander (20%): I worked with Erick to set up the basis of the web application client side. We made a basic format for each page that we could go back to and customize however we decided to later. I made the landing page, the Support Group page, and the Chatbot page. The core of the landing page I created was the description and the login button while the core of the Support Group and Chatbot pages was the text box and send button. Until the backend was finished I refined the look of different pages by modifying the css files and once the backend was running I started integrating the support group with the server. This allowed messages to be read and sent to the database. Once that was working, I was able to fix smaller issues we were having like getting the header buttons to work or having the sign in button send you to the landing page.

Notable Challenges Faced

Firebase: The database that we opted to use, Firebase, was a major boon in our web development process. However, what we didn't realize was that with free use there was a limit to the number of requests sent to the server per day. In one of our days of testing, we sent the limit (50k requests) and had to make an entirely new firebase just to continue testing the features we were implementing at the time.

Integrating Backend and Frontend: Because we began our web application with a team of frontend developers and a team of backend developers, and came together afterwards, we ran into some major complications. The team working on backend ran into individual issues with different things like how the journal entries should be stored, which led to the frontend having to make a lot of changes to compensate. Conversely, if the frontend team had a particular way that the page was set up, changes needed to be made server-side to allow frontend pieces to work, or the frontend team would need to update their pages to fit the way the server-side was set up. If we did the project again, one of the things we would've done differently is integrating earlier because it proved more difficult than we expected.

Potential Improvements

Security: To improve our security, we could add two-factor authentication, using something like Duo mobile (similar to how we log into myUCLA). This would stop potential attacks, either through hacking, phishing, or potentially even social engineering, as the user would have to log in from their phone even after the correct password is inputted.

Gamification Elements: To make our app more interesting, we could introduce elements such as challenges, achievements, or rewards to motivate users to consistently engage with the platform and their mental health journey.

Feedback Mechanism: To ensure that user experience is positive, we could implement a feedback mechanism within the application allowing users to share their thoughts, suggestions, and concerns. By actively seeking user feedback, we can better understand users' needs and preferences.

Citations

Training Data:

Mikedelong. (2023, December 4). Visualize therapy responses. Kaggle. https://www.kaggle.com/code/mikedelong/visualize-therapy-responses/notebook

Langchain Documentation:

Vector Stores. 🦜🔗 Langchain. (n.d.). https://js.langchain.com/docs/modules/data_connection/vectorstores/

Firebase Documentation:

Google. (n.d.). Build documentation | firebase documentation. Google. https://firebase.google.com/docs/build