CS 4800 Chat-Bot

Author: Marcos Herrera, Enrique Hernandez

Team: COVID-69

Code Link: <https://github.com/Inh3ritance/CS-4800-Server-Psychiatry-App>

**Abstract**

Accessibility to self-treatment comes from a new wave of self-care hospitalization methods. Health organizations and hospitals provide Booking appointments for therapy and commuting to therapy centers can certainly be expensive, and money is certainly not an additional tax that you would like to factor into your recovery of normalized behavior. This age of technology shows that we can achieve immediate clerical attention and accurate diagnosis for patients at home. Paired with other traditional means of ensuring mental safety, we can cover more bases for the protection of others. Our solution is a chat bot that can have a similar conversation as to an actual therapist, remedying the need to commute and personalize the security and confinement of your personal conversation.

**Glossary**

* NVM
* Node.js
* Mocha/Chai
* Express.js
* Firebase/GCP (Google Cloud Platform)
* JavaScript
* Functional Paradigm
* HTTPS Requests
* JSON
* Unit Testing
* Git Version Control
* Cloud Functions
* Webpack

**Solution**

The current infrastructure and design layout of our project ties its foundations to the backbone of Google’s Firebase. Firebase provides support across multiple platforms and strong documentation. Firebase can be deployed either as a standalone application or as a service. Covid-69’s solution is to create a backend service using Firebase’s API and its two forms of data storage: Realtime Database and Firestore. Among the two storages, Firestore is used for its component of saving the state of conversations between unique users.

**Goals**

*The therapist chat bot will include:*

1. User-interfaced screen, serving as the chat medium (window) between user and therapist bot
2. Log-in mechanism to allow users to resume their previous communications
3. Database/service functionality to both allocate space for logging user input and mitigating responses.

**Product Requirements**

* Log-in functionality ensures no duplicate user login credentials are created

**Open Questions:**

* How will we manage input that is consistently not understood by the chat bot?
* Will we log misunderstood user input in a separate database location for future fixes?

**Approach:**

We will work with the web-app team(s) to model a database service which will handle user input based on the design layout of their front-end user interface. Depending on their approach of the log-in functionality, this will determine the structure of the initial entities we will have to create on the back-end for user/chat logging. The chat bot’s decision-making logic will be incorporated in an standardized pattern-matching method, with a user-input-first sequence that will be imposed across all front-end teams.

**Other Options Considered:**

We considered machine-learning and neural networks for handling the chat bot’s decision-making when responding to the user since it would be a more remarkable method when compared to pattern-matching; however, these options are significantly more complex and would require more time than originally allotted.

**Front-end:**

* Design login page
* Design chat window
* Relay communication logs from front-end to back end (vice versa)

**Back-end Team:**

* Structure a database which will store user login credentials, user chat input, as well as chat bot responses
* Provide service which will allow chat bot to respond based on user-input
* Provide service which will allow chat bot to assess the user’s emotion score at the end of the chat session