Mental Illness Chatbot – Final Submission

Term: Spring 2022

Team: Mental/Physical Illness Chatbot

Project: MAPI Chatbot

Section I: Project Details

Team Members: Tusheet Goli, Tejas Pradeep, Akshay Sathiya, Pranav Khorana, Rahul Chawla, Sanket Manjesh

TA Mentor: Tushar Aggarwal

External Mentor: N/A

Project Abstract

A primary motivation behind this project is the increasing prevalence of mental health issues in recent years, in addition to ongoing physical health issues. From the National Survey on Drug Use and Health, researchers estimate the number of adults with some degree of serious psychological distress has increased by 71% from 2006 to 2017, among young adults (Rosenberg, 2019). Furthermore, due to studies at Pew Research Center, it is known that more than 70% of people aged 18-24 use Snapchat and Instagram regularly (Ortiz-Ospina, 2019).

We wish to implement a project that would be relevant to both of these trends. A chatbot system resembling a social media interface can use machine learning to predict if the user is at risk of any mental illnesses or physical illnesses.

We define mental illnesses as illnesses mainly pertaining to mental health. We plan to support anxiety, depression, and bipolar disorder in our application. We define physical illnesses as illnesses mainly pertaining to physical health. We plan to support 41 physical illnesses (specified in the Disease Symptom Classification dataset (Patil, 2020)) in our application.

In terms of features, our application intends to provide:

- Interactive UI for a mobile application that takes in user input and visually returns feedback

- Machine learning/natural language processing techniques on the backend to analyze user symptoms
- A database connection to store user information and messages
- Accurate predictions pertaining to potential mental and physical illnesses

Research

Some of the major preliminary things we need for this project are datasets, reputable and relevant research papers and articles, and a software architecture plan. We have done extensive research and have found all the datasets and papers that we plan on using and can help us in our approach. We also have a robust software architecture plan and data pipeline for our project which has been explained in our technologies and architecture diagrams. We have been able to find some good research papers that enable us to perform novel NLP applications like sentiment analysis, and other speech/text recognition patterns to identify and classify mental illness.

Tools and Technology

Through the project, we plan on using various different tools and techniques to achieve our goals.

- Mobile App
    - Mobile applications frontend shall be built in React Native to enable us to easily use the app on both Android and iOS
    - Further data for the mobile app shall be stored on SQL, using a hosting platform such as PostgreSQL with Heroku.
    - The mobile app shall also have a backend developed on Python Flask to enable Restful API development and ease of integration with other aspects of the app.
- Server
    - We plan on using a server hosted on the cloud through free-to-use services like Heroku, with data being stored in SQL.
    - The backend code for the mobile app shall be built in Python Flask and shall serve as the link between the frontend and the server.
- Machine Learning Models
    - The machine learning models shall also be built with Python, specifically using NumPy and pandas for data cleaning and data analysis and scikit-learn libraries for the ML models to be used.
    - We chose to use Python for both the backend and the ML models for better integration between the two segments.

Data Sources

- Disease Symptom Classification (Patil, 2020)
    - The Kaggle dataset lists multiple symptoms and precautions to be taken along with weighted importance values for 41 unique diseases, including GERD, AIDS, diabetes, and gastroenteritis. We plan to train an ML model on this data to recognize these diseases from the user's descriptions of symptoms and predict if they are at risk of certain physical illnesses.
    - Link - https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset?select=dataset.csv
- Twitter Emotion Analysis (Merin S, 2020)
    - This Kaggle dataset contains several tweets labeled by emotion (happy, sad, anger, etc.). We plan to develop and train an ML model on this data to predict the sentiment of the user from their messages and use that information to predict if they are at risk of certain mental illnesses.
    - Link - https://www.kaggle.com/code/shainy/twitter-emotion-analysis/data
- Emotions dataset for NLP (Praveen, 2020)
    - This Kaggle dataset contains several sentences labeled by emotion (joy, sadness, fear, etc.). We plan to train an ML model on this data to predict the sentiment of the user from their messages and use that information to predict if they are at risk of certain mental illnesses.
    - Link - https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp

System Architecture

Figure 1 below shows the architecture of the chatbot system and the flow of information between the user and the chatbot system.

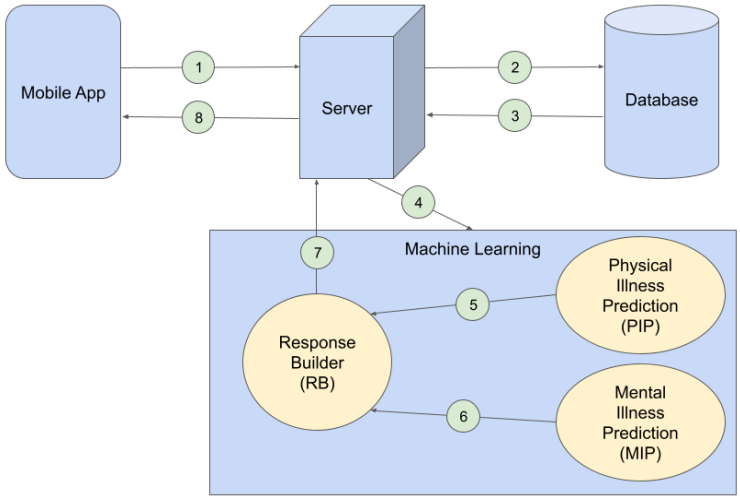


Figure 1—Architecture and information flow diagram for the chatbot system.

The chatbot system consists of a mobile app, a server, a database, and a machine learning suite. The ML suite contains an ML model for predicting physical illness (PIP), an ML model for predicting mental illness (MIP), and a response builder (RB) that builds a response to the user from the results of both models.

The information flow is described below as a series of numbered interactions that correspond to the numbered interactions shown in the diagram.

1. The user sends a message from the mobile app, which is received by the server.
2. The server stores the message in the database.
3. The server gets the last X messages from the database. The value of X will be tuned during the development and testing of the system.
4. The server sends the X messages to the ML suite.
5. The last message is passed to the PIP model to predict the physical illnesses the user may be at risk.
6. All X messages are passed to the MIP model to predict the mental illnesses the user may be at risk.
7. The RB builds a response to the user from the results from the PIP and MIP models.
8. The server sends the response to the user.

Team Member Roles & Responsibilities

Tusheet Goli - API Design, Database Implementation and Hosting, Backed Development

Tejas Pradeep - API Design, Backed Development, Project Integrations

Akshay Sathiya - Physical Illnesses Prediction ML Models, Project Integrations

Pranav Khorana - Frontend Design, React App

Rahul Chawla - Mental Illness Prediction ML Model

Sanket Manjesh - Mental Illness Prediction ML Model

Final Gantt Chart

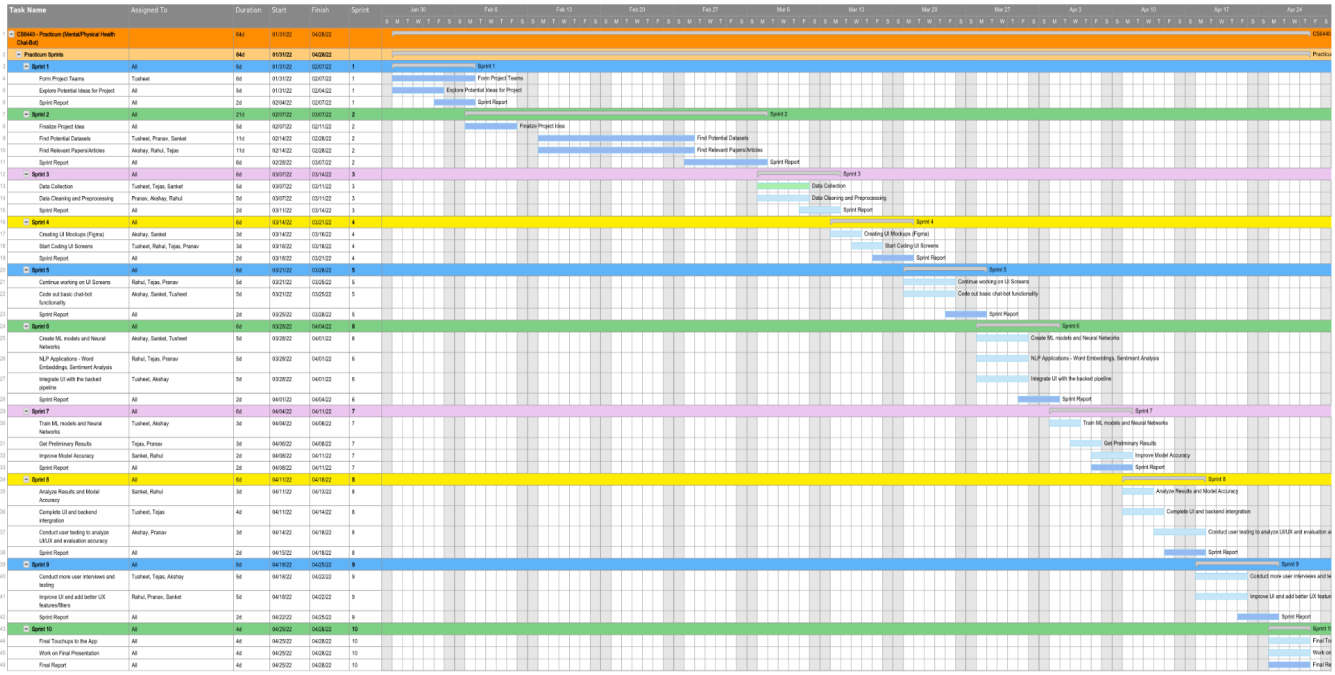


Figure 2—Gantt chart outlining project timeline.

Section II – Application or Solution

Final Git Commit: (cca2f78)
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/commit/cca2f78ca17cab5129c1dab385231a9bfd32f5a1

Github Link:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot

Branch: (main)
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/tree/main

Application or Solution Details

App or Solution Name: MAPI Chatbot

App or Solution URL: App is not deployed, but can be run as a regular react application from our Github repository linked below.
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot.

App or Solution Description

Our project idea, MAPI Chatbot, tries to provide a unique approach to self-care and self-diagnosis. We aim at diagnosing illnesses based on the symptoms that people are feeling and refer them to the right practitioners. For diagnosing mental health issues, we implemented an NLP application to analyze emotion and sentiment (Uban, 2021) in peoples' conversations with the chatbot that poses specific questions to check up on their wellbeing. Based on these evaluations, we can diagnose potential mental health issues like stress, anxiety, depression, etc. We used a similar NLP-based approach to identify physical illnesses as well. The UI of the chatbot is interactive and resembles a social media platform helping the user feel more comfortable while mentioning their symptoms, Further to facilitate natural conversation, the chatbot uses Natural Language Processing to extract symptoms from the conversation and further extract the emotions of the user messages to facilitate the mental illness predictor. Currently, the chatbot can diagnose a wide variety of physical illnesses but is limited to diagnosing three mental illnesses namely depression, anxiety, and bipolar disorder.

Section III – Project Presentation

Link to Video Presentation - https://youtu.be/B7Up9SlytSI

Link to Slides -
https://docs.google.com/presentation/d/1H_-HUZiL_Ps6Vox8ky3GHBFYtxIRQo6dOesN9dASrA8/edit?usp=sharing

Section IV – Project Documentation

Final Delivery Directory:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/tree/main/Final%20Delivery

Final Gantt Chart:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/blob/main/Final%20Delivery/Gantt%20Chart.png

Application or Solution Manual:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/tree/main/Final%20Delivery/Application%20Manual

Special Instructions (if not included in the manual): None

Research Directory:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/tree/main/Final%20Delivery/Research

Documentation Directory:
https://github.gatech.edu/gt-cs6440-hit-spring2022/Team-2-Mental-Illness-Chatbot/tree/main/Final%20Delivery/Documentation

Mental and Physical Illness (MAPI) Chatbot

Tusheet Goli, Tejas Pradeep, Akshay Sathiya, Pranav Khorana, Sanket Manjesh, Rahul Chawla
tgoli3@gatech.edu, tpradeep8@gatech.edu, asathiya6@gatech.edu, pkhorana3@gatech.edu, smanjesh3@gatech.edu, rchawla36@gatech.edu

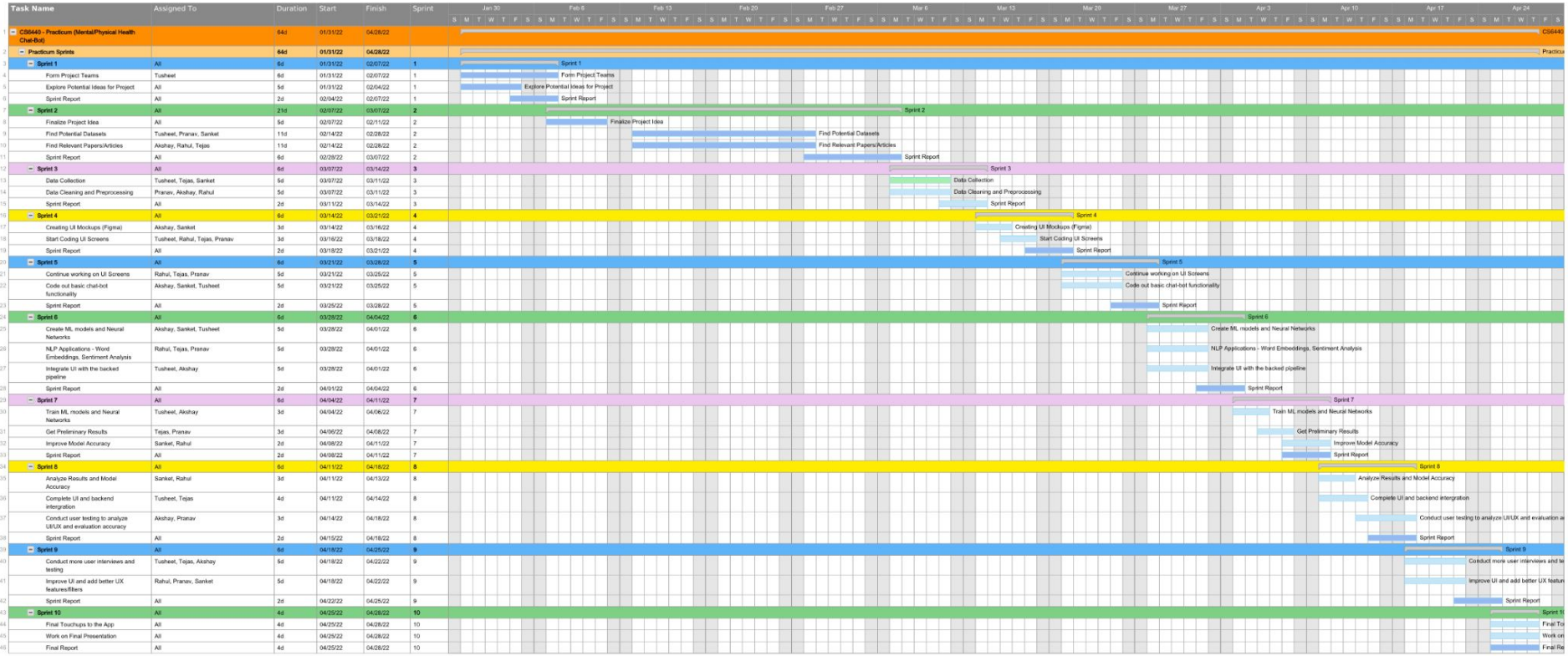
Introduction

- Definitions
  - Physical illness: illnesses pertaining to one’s physical health (common cold, flu, etc.)
  - Mental illness: illnesses pertaining to one’s mental health (anxiety, depression, etc.)
- Problem
  - People may not be able to immediately consult primary healthcare provider upon experiencing symptoms of mental or physical illness.
  - Determining illness risks helps them know if a doctor’s visit is required how to better take care of themselves.
- Solution
  - Mental and Physical Illness Chatbot

Introduction

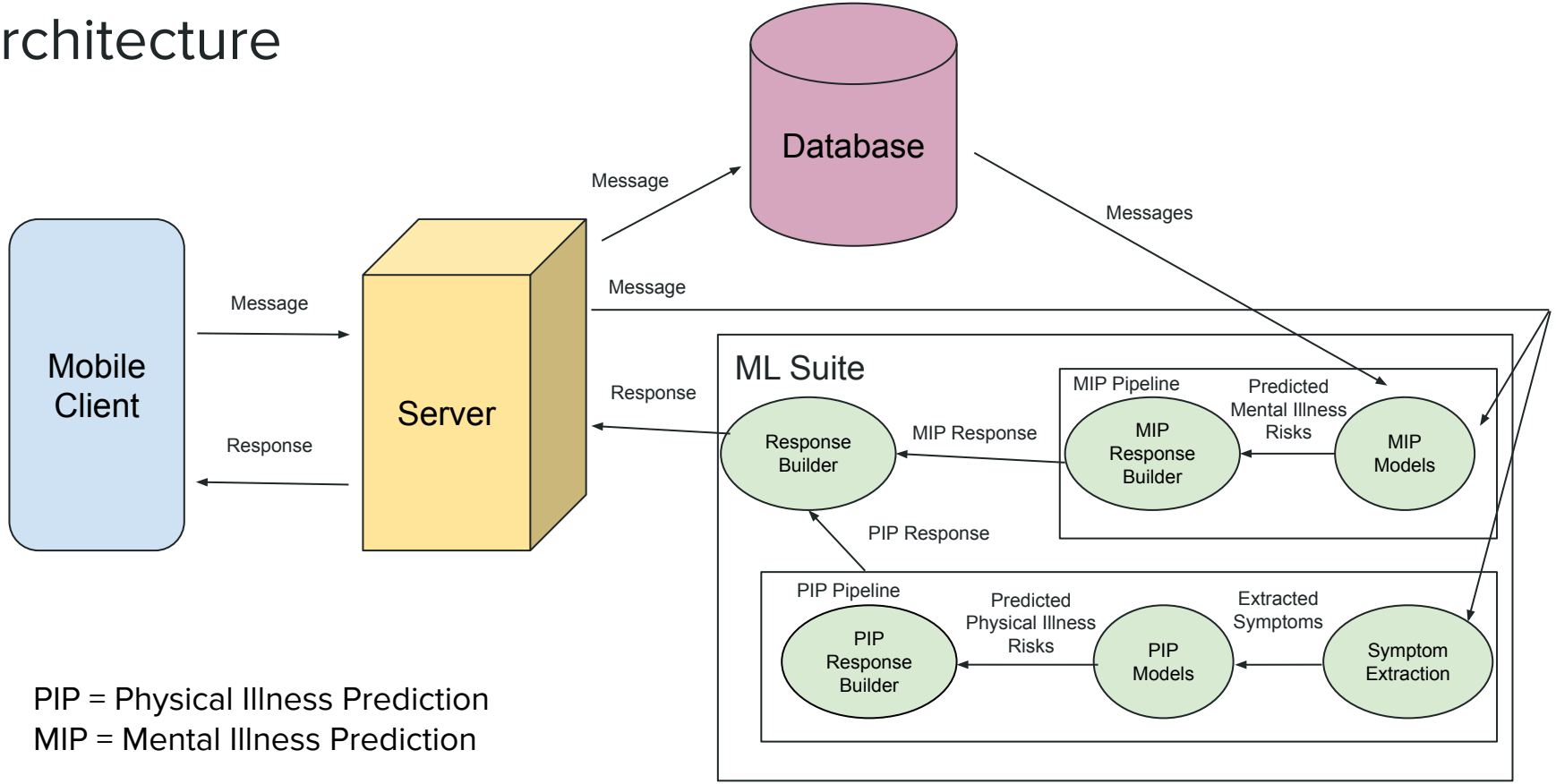
- Team contributions
  - Akshay Sathiya: Data processing for Kaggle physical diseases dataset (Patil, 2020), physical illness prediction (PIP) pipeline, and the PIP response builder to generate messages containing physical illness prediction results and precautions to send back to user.
  - Pranav Khorana: Created front-end React Native application and integrated w/ backend
  - Tusheet Goli: Worked on API design, backend implementation, and database hosting (Heroku)
  - Tejas Pradeep: Worked on the backend API design, and integrated front end w/ model
  - Rahul Chawla: Worked on the mental illness prediction (MIP) pipeline, created the logistic regression model and data processing for emotions dataset (Praveen, 2020)
  - Sanket Majesh: Worked on the mental illness prediction (MIP) pipeline

Project Implementation Status



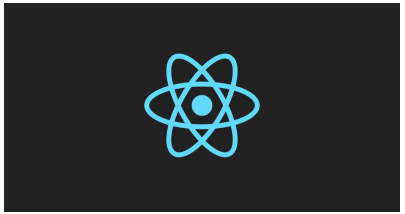
	Task Name	Assigned To	Duration	Start	Finish	Sprint
1	CS6440 - Practicum (Mental/Physical Health Chat-Bot)		64d	01/31/22	04/28/22	
2	Practicum Sprints		64d	01/31/22	04/28/22	
3	Sprint 1	All	6d	01/31/22	02/07/22	1
4	Form Project Teams	Tusheet	6d	01/31/22	02/07/22	1
5	Explore Potential Ideas for Project	All	5d	01/31/22	02/04/22	1
6	Sprint Report	All	2d	02/04/22	02/07/22	1
7	Sprint 2	All	21d	02/07/22	03/07/22	2
8	Finalize Project Idea	All	5d	02/07/22	02/11/22	2
9	Find Potential Datasets	Tusheet, Pranav, Sanket	11d	02/14/22	02/28/22	2
10	Find Relevant Papers/Articles	Akshay, Rahul, Tejas	11d	02/14/22	02/28/22	2
11	Sprint Report	All	6d	02/28/22	03/07/22	2
12	Sprint 3	All	6d	03/07/22	03/14/22	3
13	Data Collection	Tusheet, Tejas, Sanket	5d	03/07/22	03/11/22	3
14	Data Cleaning and Preprocessing	Pranav, Akshay, Rahul	5d	03/07/22	03/11/22	3
15	Sprint Report	All	2d	03/11/22	03/14/22	3
16	Sprint 4	All	6d	03/14/22	03/21/22	4
17	Creating UI Mockups (Figma)	Akshay, Sanket	3d	03/14/22	03/16/22	4
18	Start Coding UI Screens	Tusheet, Rahul, Tejas, Pranav	3d	03/16/22	03/18/22	4
19	Sprint Report	All	2d	03/18/22	03/21/22	4
20	Sprint 5	All	6d	03/21/22	03/28/22	5
21	Continue working on UI Screens	Rahul, Tejas, Pranav	5d	03/21/22	03/25/22	5
22	Code out basic chat-bot functionality	Akshay, Sanket, Tusheet	5d	03/21/22	03/25/22	5
23	Sprint Report	All	2d	03/25/22	03/28/22	5
24	Sprint 6	All	6d	03/28/22	04/04/22	6
25	Create ML models and Neural Networks	Akshay, Sanket, Tusheet	5d	03/28/22	04/01/22	6
26	NLP Applications - Word Embeddings, Sentiment Analysis	Rahul, Tejas, Pranav	5d	03/28/22	04/01/22	6
27	Integrate UI with the backed pipeline	Tusheet, Akshay	5d	03/28/22	04/01/22	6
28	Sprint Report	All	2d	04/01/22	04/04/22	6
29	Sprint 7	All	6d	04/04/22	04/11/22	7
30	Train ML models and Neural Networks	Tusheet, Akshay	3d	04/04/22	04/06/22	7
31	Get Preliminary Results	Tejas, Pranav	3d	04/05/22	04/08/22	7
32	Improve Model Accuracy	Sanket, Rahul	2d	04/08/22	04/11/22	7
33	Sprint Report	All	2d	04/08/22	04/11/22	7
34	Sprint 8	All	6d	04/11/22	04/18/22	8
35	Analyze Results and Model Accuracy	Sanket, Rahul	3d	04/11/22	04/13/22	8
36	Complete UI and backend intergration	Tusheet, Tejas	4d	04/11/22	04/14/22	8
37	Conduct user testing to analyze UI/UX and evaluation accuracy	Akshay, Pranav	3d	04/14/22	04/18/22	8
38	Sprint Report	All	2d	04/15/22	04/18/22	8
39	Sprint 9	All	6d	04/18/22	04/25/22	9
40	Conduct more user interviews and testing	Tusheet, Tejas, Akshay	5d	04/18/22	04/22/22	9
41	Improve UI and add better UX features/filters	Rahul, Pranav, Sanket	5d	04/18/22	04/22/22	9
42	Sprint Report	All	2d	04/22/22	04/25/22	9
43	Sprint 10	All	4d	04/25/22	04/28/22	10
44	Final Touchups to the App	All	4d	04/25/22	04/28/22	10
45	Work on Final Presentation	All	4d	04/25/22	04/28/22	10
46	Final Report	All	4d	04/25/22	04/28/22	10

Architecture

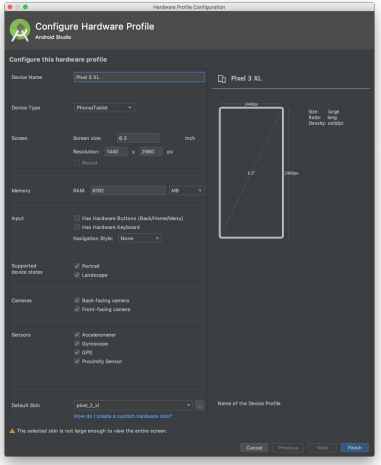


PIP = Physical Illness Prediction
MIP = Mental Illness Prediction

Setting Up Frontend



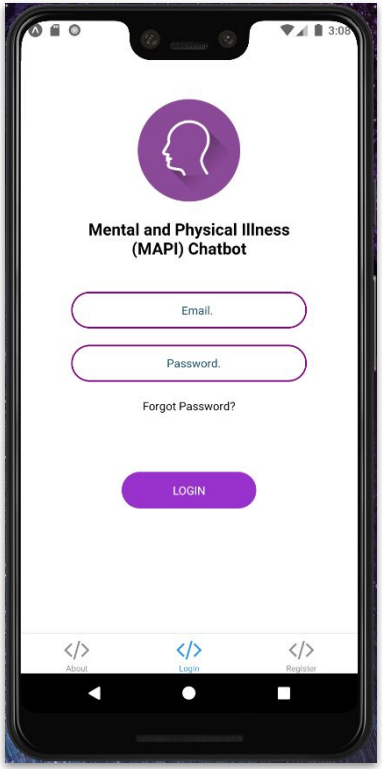
- Used React Native to create front-end application
  - Easy to deploy mobile apps on both Android and iOS operating systems
  - Utilized Javascript with Typescript configuration
- Android Emulator for testing and debugging
  - Pixel 3 XL with Android 8.0 operating system
- Utilized Expo, a framework that allows users to easily develop, build, and deploy React Native apps

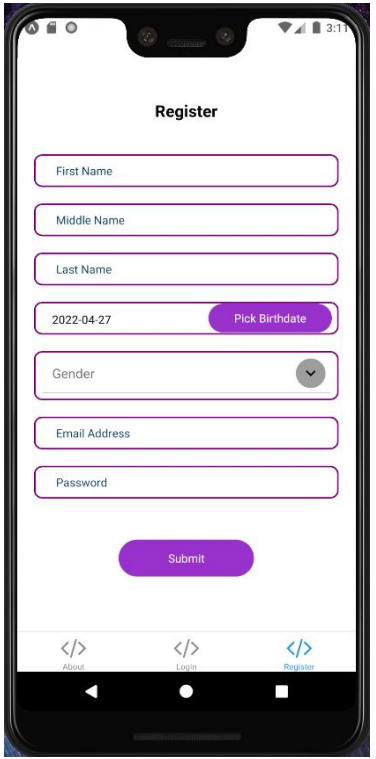


Creating React Native App

- React-Navigation and Bottom TabNavigator
  - Organized hierarchy for user to navigate (logging in and out/switching tabs)

- TextInput, Touchable Opacity, Dropdown, DateTime Picker
  - Utilized a variety of components for user to input info
  - Registering and logging in

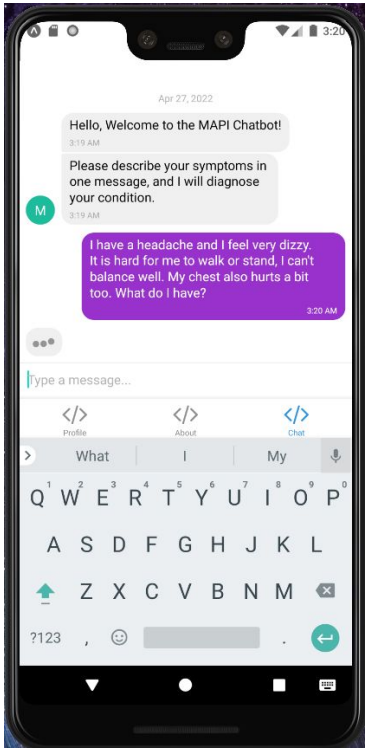


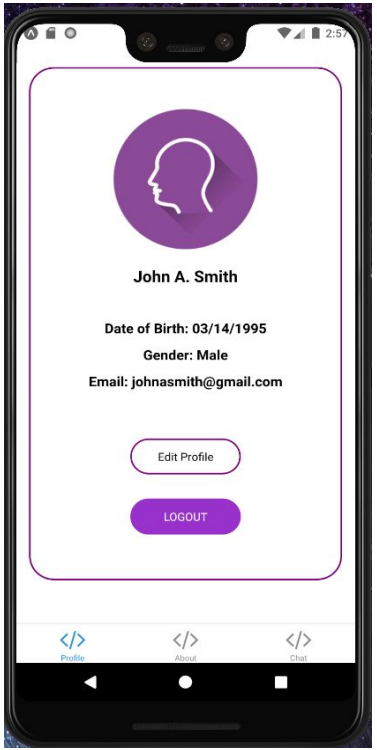


Creating React Native App

- Hooks to keep track of state & lifecycle components
  - useState: keep track of info (ex: name, date of birth, email, etc.)
  - useEffect: side effect based on rendering of components (ex: initial population of profile screen)

- React-Native-Gifted-Chat
  - Utilized component to create interactive chat UI
  - Customizable attributes for sending messages, bubbles, users, typing indicator





Integration with Backend

- Utilized axios library to make requests to Flask server

- Utilized promises/callback functions to deal with asynchronous requests
  - Make sure application doesn't crash or displays incomplete information

- Backend Functionality
  - Authenticate user on login screen
  - Register user with info on register screen
  - Get user data to load data on profile screen
  - Send message on chat screen

```
const getResponse = async (message: string)=> {
  const obj = {
    message: message
  }
  axios.post(baseUrl + '/send-message', obj)
  .then((response) => {
    var responseText = response.data['RESPONSE'];
    setTyping(false)
    sendBotResponse(responseText.trim())
    console.log(response)
  }, (error) => {
    console.log(error);
  });
};
```

Backend

- The backend for the app is a Python Flask APP
- It is built using a REST API model, and servers as the layer in between the front end and the model

- The backend consists of 4 key endpoints listed below - each endpoint server to implement logic for the frontend
- Backend Calls
  - POST /send-message
  - GET /authenticate-user/<email>/<password>
  - POST /register-user
  - GET /get-user-data

Database

- Postgresql database consisting of chat information and user information

```
CREATE TABLE chat (
    user_id varchar(64),
    message varchar(1024),
    response varchar(1024)
);
```

```
CREATE TABLE users (
    user_id varchar(64),
    fname varchar(256),
    mname varchar(256),
    lname varchar(256),
    dob varchar(10),
    gender varchar(1),
    email varchar(256),
    password varchar(64)
);
```

Backend Code Walkthrough

- Code Walkthrough!

PIP Pipeline

- Data processing on Kaggle physical diseases dataset (Patil, 2020)
  - Determine unique symptoms and assign a numeric label for each unique physical illness
  - Produces dataset for symptom extraction (enumerates through combinations of 1, 2, 3, and 4 symptoms, subsampling for combinations of size 3 and 4).
    - Features: messages describing symptoms, built from pre-written message templates.
    - Data augmentation (done with 40% probability): symptom tokens reversed
    - Labels: unique symptoms (0 if not present, 1 if present)
  - Produces dataset for physical illness prediction given symptoms
    - Features: unique symptoms (0 if not present, 1 if present)
    - Labels: numeric label for physical illness corresponding to present symptoms

PIP Pipeline

- Symptom extraction
  - TF-IDF vectorization (scikit-learn developers, *sklearn.feature_extraction.text.TfidfVectorizer*, 2022)
  - Multi-output classifier (scikit-learn developers, *sklearn.multioutput.MultiOutputClassifier*, 2022).
    - Neural networks trained on dataset for symptom extraction (scikit-learn developers, *sklearn.neural_network.MLPClassifier,* 2022).
- Prediction of physical illness given symptom information
  - Random forest classifier (scikit-learn developers, *sklearn.ensemble.RandomForestClassifier*, 2022).
  - Neural network classifier (scikit-learn developers, *sklearn.neural_network.MLPClassifier,* 2022).
- Model evaluation
  - Model only trained once, validated on several different splits of the data
    - Accuracy on train/test split
    - F1 score on train/test split (scikit-learn developers, *sklearn.metrics.f1_score*, 2022)
    - K-fold cross validation (scikit-learn developers, *sklearn.model_selection.KFold*, 2022)
      - Accuracy and F1 score on each fold

PIP Pipeline

- Response building
  - Top three physical illness predictions and classification probabilities, rounded to one decimal place.
  - Corresponding precautions from physical diseases dataset, for preventing/mitigating each predicted physical illness (e.g. drink sugary drinks for hypoglycemia) (Patil, 2020).
  - Assembled into string to show the user, sent to backend, sent to frontend.

MIP Pipeline

- Data processing on emotions dataset
  - Determine most probable mental illness based on distribution of sentiment of series of user messages
  - Utilizes sklearn logistic regression model to train the classifier (scikit-learn developers, *sklearn.linear_model.LogisticRegression*, 2022)
- Predictions of most likely illness are determined by performing a time series analysis on the messages passed in by the user
  - Depression: Consistently sad or little joy/happiness
  - Bipolar Disorder: Wide fluctuations in sentiment between messages
  - Anxiety: Mix of surprise, fear, and obvious absence of strong sad/joy emotions
- Response string is built by providing the user with their most likely illness
  - Assembled in string to show to the user, then sent to backend, then frontend

MIP Heuristics

- Created heuristics to determine if patients fall into one of 4 categories: depression, anxiety, bipolar, or no mental illnesses
- For depression, checked if sentiment of sadness was the most common in a message
- For anxiety, checked if sentiment of fear was most common in a message
- For bipolar, checked if sentiments often switched from sad/fear to happy throughout a message and if these sentiments were balanced throughout
- If the patient fell into none of these categories, he/she was diagnosed with no mental illness

Research

- Datasets
  - Physical diseases dataset: *Disease Symptom Prediction* (Patil, 2020).
  - Twitter dataset: *Twitter Emotion Analysis* (Merin S, 2020).
  - Emotions dataset: *Emotions dataset for NLP* (Praveen, 2020).
- PIP Pipeline
  - Symptom extraction
    - Keyword extraction (RAKE: Rapid Automatic Keyword Extraction) (Saxena, 2020)
    - Multi-output classifiers (scikit-learn developers, *sklearn.multioutput.MultiOutputClassifier*, 2022)
- MIP Pipeline
  - Logistic Regression (sklearn developers, *sklearn.linear_model.LogisticRegression*, 2022)

Local Testing

- PIP pipeline
  - Four stages: data processing, fitting symptom extraction model (multi-output classifier of neural networks), fitting each PIP model (random forest, neural network). See command to evaluate already-trained models below.

```
$ python3 ./models/physical_illness_prediction.py --proc_data 1 --fit_se_nn 1 --fit_pip_rf 1 --fit_pip_nn 1
> ./models/physical_illness_prediction_logs.txt
```

- Response builder (for PIP pipeline)
  - Tests responses generated by PIP pipeline (14 test cases, 12 / 14 passed). See command to run test cases below.

```
$ python3 ./test.py > ./response_builder_test_logs.txt
```

- MIP pipeline
  - Stages: Data processing, Accuracy Training for LR model

```
$ python3 ./models/mental_illness_prediction.py
```

- See README.md file and corresponding log files for more details.

Solution Demonstration

- Demo time!

Future Work

- PIP Pipeline
  - Improve data processing and machine learning techniques (sentence templates, data augmentation, etc.) to improve symptom extraction.
  - Obtain more data about symptoms corresponding to illnesses, improve performance of PIP models and apply it to more symptoms and illnesses.
- MIP Pipeline
  - Improve heuristic to determine mental illness risks from sentiment analysis information
  - Support more mental illnesses beyond depression, anxiety, and bipolar disorder.
- General
  - Integration with healthcare organizations to keep providers in the loop with what the patient may be experiencing before any necessary appointments.
  - Integration with pharmacies to suggest medications that align with the precautions provided to the user by the app.

References

Merin S, S. (2020, April 17). *Twitter Emotion Analysis*. Kaggle. Retrieved March 6, 2022, from https://www.kaggle.com/shainy/twitter-emotion-analysis/data

Patil, P. (2020, May 24). *Disease Symptom Prediction*. Kaggle. Retrieved April 27, 2022, from https://www.kaggle.com/itachi9604/disease-symptom-description-dataset

Praveen. (2020, April 16). *Emotions dataset for NLP*. Kaggle. Retrieved March 6, 2022, from https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp

Saxena, N. (2020, September 6). *Extracting Keyphrases from Text: RAKE and Gensim in Python*. Extracting Keyphrases from Text: RAKE and Gensim in Python | by Nikita Saxena | Towards Data Science. Retrieved April 27, 2022, from https://towardsdatascience.com/extracting-keyphrases-from-text-rake-and-gensim-in-python-eefd0fad582f

scikit-learn developers. (2022). *sklearn.ensemble.RandomForestClassifier*. sklearn.ensemble.RandomForestClassifier — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

scikit-learn developers. (2022). *sklearn.feature_extraction.text.TfidfVectorizer*. sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

References

scikit-learn developers. (2022). *sklearn.linear_model.LogisticRegression* — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

scikit-learn developers. (2022). *sklearn.metrics.f1_score*. sklearn.metrics.f1_score — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

scikit-learn developers. (2022). *sklearn.model_selection.KFold*. sklearn.model_selection.KFold — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

scikit-learn developers. (2022). *sklearn.multioutput.MultiOutputClassifier*. sklearn.multioutput.MultiOutputClassifier — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputClassifier.html

scikit-learn developers. (2022). *sklearn.neural_network.MLPClassifier*. sklearn.neural_network.MLPClassifier — scikit-learn 1.0.2 documentation. Retrieved April 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Thanks for Listening!