Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

User Manual

deeper

by

Team #6
deeper

Team Members

Nicole Kwon kwonn@bu.edu
Daniel Li dli0793@bu.edu
Harry Li harryli@bu.edu
Byron Mitchell byronmit@bu.edu

deeper User Manual

BUEngineering College of Boston University

Table of Contents

Executive Summary

Our platform, *deeper*, allows mental health consumers and their loved ones to better connect and empathize with one another. The aim of our project is not necessarily to build new connections or provide professional mental health services; rather, our value comes from strengthening the connections that the user already has. Focusing on software, we have created a cross-platform mobile application that provides functionalities that tackle user stories of mental health consumers: monitoring mental wellbeing, engaging in a safe community, journaling, and sending and receiving supportive messages. This monitoring system will be accomplished through machine learning, in which the application will be able to gather information from an individual's interaction with the app (i.e. through journaling and surveys) to "quantify" a user's mental health and display the results through a graph. In addition, with the existing lack of understanding for how to help those with mental illnesses, this app acts as a way to educate and connect loved ones to mental health consumers by them being able to ask questions in the community forums and send messages of encouragement. In short, this application is solving the pressing issue that no other mental health solution is addressing: the miscommunication and misunderstanding between those suffering from mental health disorders and their loved ones.

1 Introduction

From a survey that we conducted, the majority of loved ones who participated stated that the methods that they used to support the mental health consumer was less than a 5 on an effectiveness scale from 1 to 10. Therefore, the app is for the mental health consumer and their loved ones. This application is needed because it is difficult for both parties to empathize with each other and understand each other's needs. The loved one wants to support the mental health consumer but does not know the best strategies usually to do so; at the same time, the mental health consumer wants the loved one to understand and not burden them. Especially on college campuses, mental health resources tend to be difficult to access or there are not enough resources to best support students. This application hopes to solve this issue and provide another supportive mechanism, especially since there are not currently any mental health solutions that target improving the connection between mental health consumers and their loved ones.

This project involves the design of a mobile application that aims to bridge the gap between those suffering from mental health disorders (focusing on depression/anxiety with college students on college campuses first) and their friends and family. Our frontend is React Native, which will allow for smooth user interaction and experience with our application, and our backend and databases are PyFlask, Firebase, and MongoDB, which will allow for user information to be processed, retrieved, and stored through RESTful APIs (these technologies will be further detailed in the Technical Background portion of this manual). We will be testing our application directly with college students and deploying the app on Google Play and the Apple store. We aim to achieve a 75% accuracy on our machine learning algorithm for our journal and plan to continue training and iterating to approach the 100% mark.

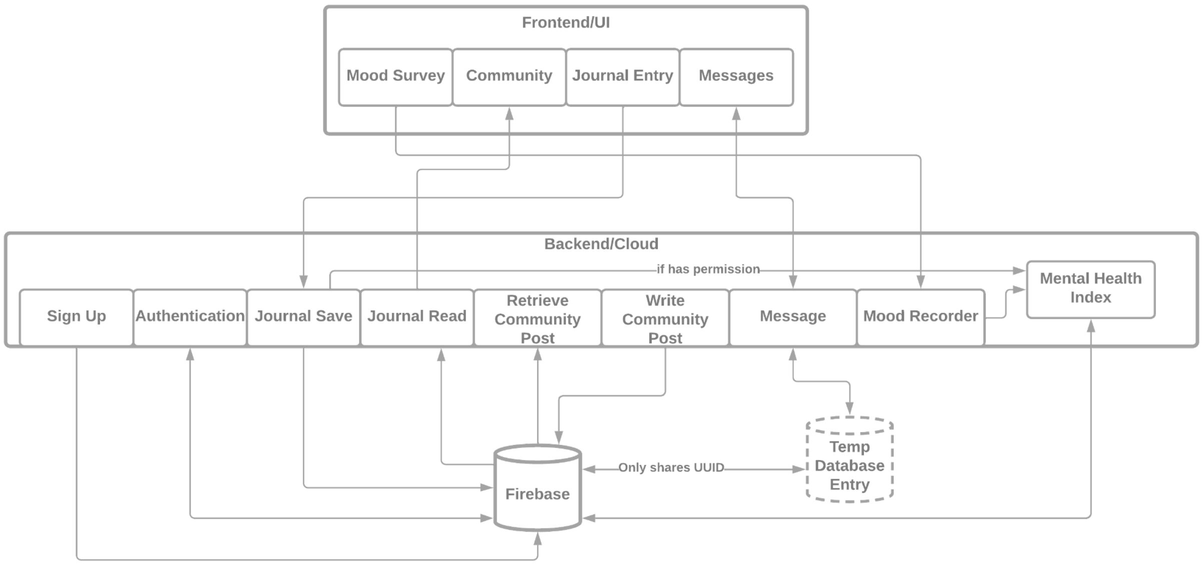
2 System Overview and Installation

2.1 Overview Block Diagram

As the diagram shows, our frontend is connected to the backend via APIs. Each functionality on the frontend only records the input from the user, and then passes onto our backend to be processed. The backend will determine if the input frontend passed needs to be saved to the database or returns an error message.

In our backend, there is a module called Mental Health Index. It is a separate module from other backend functionalities. This contains our machine learning algorithm to analyze user's journal entries. After the user allows our backend to read the journal entries, we will run the algorithm through all the entries. This will return indices showing the emotional fluctuations.

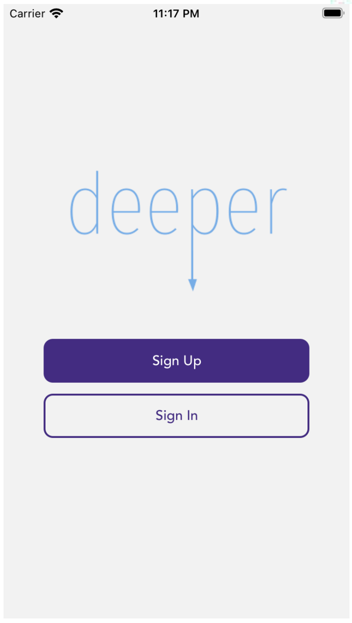
For the messaging functionality, each message will only temporarily be saved into the database as a single entry. After the message is successfully delivered, the entry in database will be deleted.

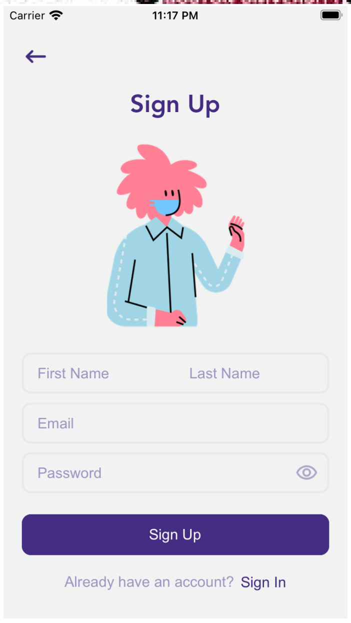


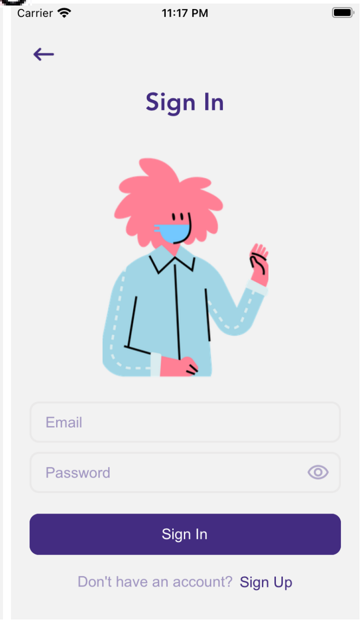
2.2 User Interface

When downloading and opening our application, the user is prompted to the splash screen, which contains the logo of our project and two interactive buttons: sign up and sign in. Sign up allows the user to create an account with their name, email, and password, while sign in allows the user to access an existing account that they have already made.

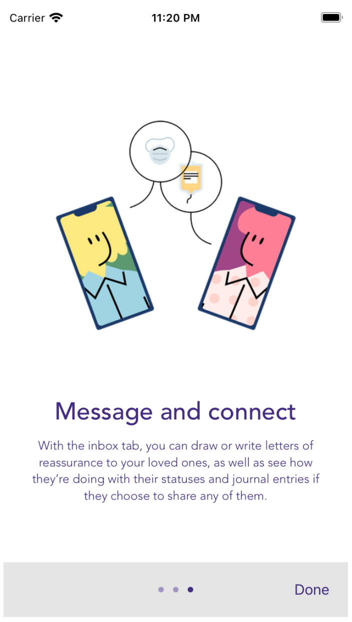
BU College of Engineering BOSTON UNIVERSITY





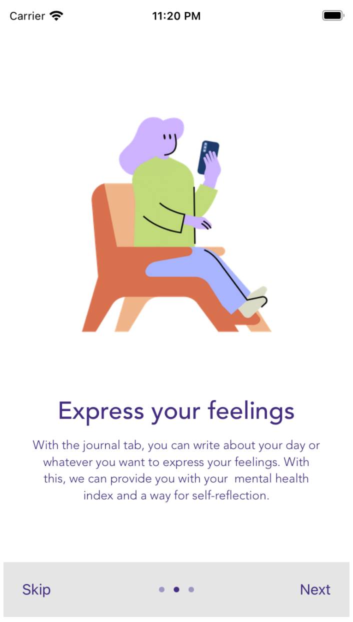


This is followed by onboarding pages, which allow the user to learn more about our application. They have the ability to navigate back and forth between screens through swiping the screen or hitting the next button. They also have the ability to skip these screens entirely.



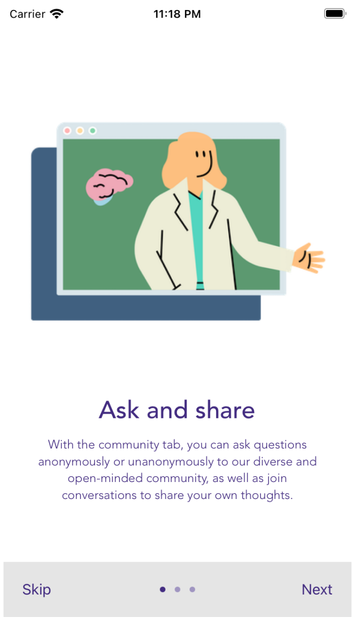
Message and connect

With the inbox tab, you can draw or write letters of reassurance to your loved ones, as well as see how they're doing with their statuses and journal entries if they choose to share any of them.



Express your feelings

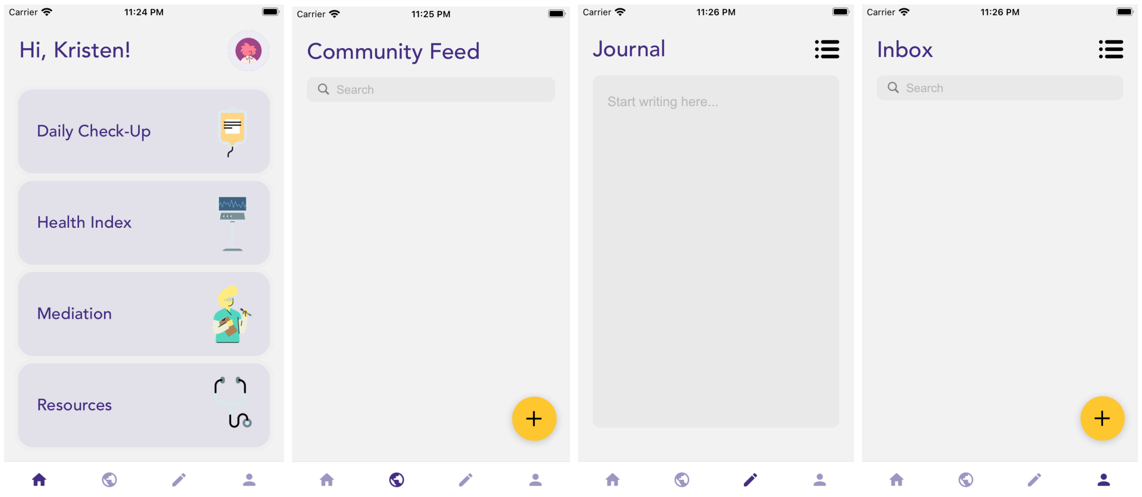
With the journal tab, you can write about your day or whatever you want to express your feelings. With this, we can provide you with your mental health index and a way for self-reflection.



Ask and share

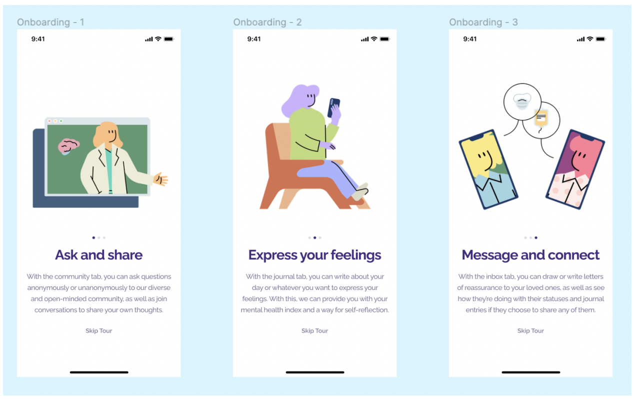
With the community tab, you can ask questions anonymously or unanonymously to our diverse and open-minded community, as well as join conversations to share your own thoughts.

Finally, the user can interact with the main functionalities of our application through pressing the different icons of the navigation bar at the bottom of the screen. There is the home page, which allows the user to monitor their mental health and take a survey through the daily check-up; the community feed page, which allows the user to post something and interact or search up existing posts; the journal page, which allows the user to write about anything and check their past entries; and the inbox page, which allows the user to send their own messages, read and filter messages sent to them, and search up messages.



2.3 *Physical Description*

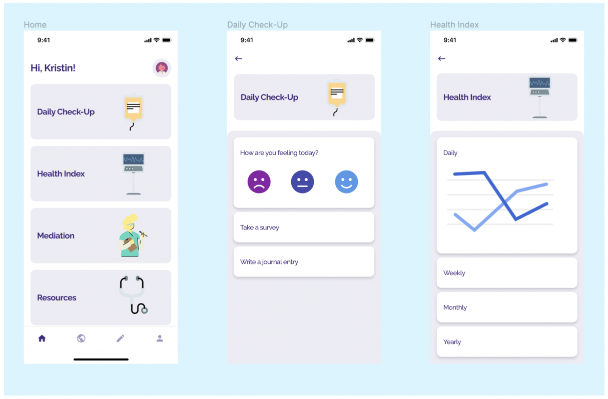
Although there is no sketch of our project hardware (as our project is solely software-based), we have a Figma which allows us to wireframe our application. With the target of college students as mental health consumers in mind, we decided on a singular user interface that will be simple and easy-to-use, emulating social media for building bonds and familiarity. Shown below are some parts of our wireframe, and our user interface created with React Native emulates our Figma wireframe. For the entire wireframe, please refer to the following link: https://www.figma.com/file/8lHPYkZNslN37Y4j8CfBP6/Healthcare-App-with-Blush-Illustrations-Community?node-id=6%3A0.



With the community tab, you can ask questions anonymously or unanonymously to our diverse and open-minded community, as well as join conversations to share your own thoughts.

With the journal tab, you can write about your day or whatever you want to express your feelings. With this, we can provide you with your mental health index and a way for self-reflection.

With the inbox tab, you can draw or write letters of reassurance to your loved ones, as well as see how they're doing with their statuses and journal entries if they choose to share any of them.



2.4 Installation, setup, and support

In order to install the app, we're currently running on expo testbench. This requires users to download our github repository, and direct to the folder and find the "frontend-new-new". Then, open the terminal under that directory, and run "expo start". Then, use iPhone or Android to download the app called "Expo". Then, use the camera to scan the pop-up QR code to run the app on your phone. Make sure that the computer running the app and your phone stays under the same WiFi during the startup process.

We're also planning to release the app on both Apple App Store and Google Store. After we release the app, it is downloadable on both the App Store on iOS and Google Play Store by searching "deeper".

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

In a normal operating mode, the user will interact with the app as follows:

1. The user provides login or signup information to proceed into the homepage, which contains the other main functionalities of our app, including the community feed, journaling, and messaging systems.
2. Users will navigate the different pages through the navigation bar at the bottom of the screen or the back buttons on the top left of the screen.
3. In this mode, the user's interactions with the app are saved and personalized (i.e. the user must press a button to save journal entries, the user can pick and choose who they want to share their journal entries with, if at all).
4. The normal consequences of user actions involve the APIs working successfully and retrieving/sending user input back and forth between the app and database. As such, abnormal results that might occur from this mode involve the APIs not working as smoothly, potentially leading to information not saving properly or taking a long time to be sent or retrieved.
5. The user can sign out in the settings page, which can be accessed by clicking on their profile picture, to return back to the splash page.

3.2 Operating Mode 2: Abnormal Operations

Abnormal operations include the app's pages not loading properly (i.e. the community feed is not showing other people's posts, the journal and inbox are not filtering properly) and the app not retrieving and not sending the user input (i.e. a connection error with the database).

To exit this state of abnormal operations, the user should try to go into the settings to sign out or delete the app completely and redownload. If time permits, a functionality might be implemented where the user can contact us in their settings to report any problems they face/any feedback they want to send to us.

3.3 Safety Issues

Due to the lack of hardware and focus on software, the main safety issues involve the user's data, especially because our app is geared towards mental health consumers. We must be mindful of data security, as the user signs up using their full name, email address, and password, and they might share sensitive information through the journaling functionality. We also must be mindful of the potential inappropriate content that might be posted through the community feed, though this will be filtered out.

4 Technical Background

a. Login and Sign Up:

Our technical approach to realize this feature is by connecting our frontend to Firebase and using libraries that enable 3rd party login. The reason behind using Firebase is:

1. Transiting some of the basic functionalities from backend to frontend in order to make our backend lighter and easier to run;
2. Easier to integrate the functionality into the frontend using React Native since there are existing libraries that do the same functionality.

Both Email signup and 3rd party signup will create a profile in the Firebase database.

b. Journal Saving and Retrieving:

In order to save the journal into the correct user's profile in the database, we use the UUID (Unique User ID) as the identifier. Meanwhile, the backend will save the journal content and the time on the device into the backend. Therefore, when the user is trying to retrieve the journal entries, the backend can just use the UUID as its identifier and direct itself to the profile and retrieve all the saved entries in the order of the creation date. These simple retrieve and create actions can be done by using React Native's code. Therefore, there's no need to add complexity to the Python backend. Also, it is easier to implement this simple functionality using React Native.

c. Community Post:

For the community posts, we've saved the data into Firebase. We use each post ID and save these IDs into each person's profile. By doing so, we are able to group each post into each person's profile, and also have the ability to instantly retrieve all the posts and display them in the community page. Our database is saving these posts using formats like:

Image: *URL of the image*;

Text: *The text content of the post*;

Timestamp: *The timestamp of the server*

Because the image is saved into the Firebase Storage, and Each item stored into the Firebase Storage section has a URL, the backend will first upload the image into Firebase Storage, and then submit the URL of the image to the database. Meanwhile, the text portion of the post is going to be saved into the text section in the data entry. And in order to order them by its creation time, we use the server's timestamp as the time reference of each post.

d. Mental Health Index:

We use a Natural Language Processing algorithm to calculate each person's emotional index by categorizing the negative emotions and positive emotions and quantinize them. We're using text segmentation to filter out all the words that don't have an emotional meaning. Then, we run the text through the NLP algorithm. It'll then match the word with a labeled dictionary to compute all the emotions in the text. Then, we'll collect all the emotions presented in the text, and calculate an emotional index based on the percentage of each emotion presented in the text. The data will be saved under each user's profile.

e. Messaging System:

Our messaging system design is very simple. Each message is saved locally. The server will not preserve any message.

5 Relevant Engineering Standards

The most relevant engineering standards to our project surround software design and coding standards. This includes REST APIs and SSO (single sign on). REST APIs are laced within the functionalities of our project, from signing in/signing up to saving and storing community posts, journal entries, and user messages. SSO is used in terms of Google authentication, as the user can simply sign in with one set of credentials (in this case, their Google account).

In terms of following engineering standards for REST API design, the user should take into account security, performance, and ease of use. There should be consistency of web standards and conventions, and this has been incorporated within our app, as the frontend and backend communicate through JSON and HTTP status codes, which are the standard building blocks of modern software design. In addition, performance should be increased through not returning too much data at a time. In terms of following engineering standards for SSO design, there are standard protocols to define how service providers and identity providers can exchange identity and authentication information with one another. In our case, the user is redirected to an external identity provider (Google), which then provides a better experience for users because they can use their existing credentials to authenticate and do not have to enter credentials as often.

6 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	**Quantity**	**Description**	**Unit Cost**	**Extended Cost**
1	1	Apple Developer Account	$99	$99/year
2	1	Google Play Developer Account	$25	$25
3	1	SightEngine API	$29	$29/mo
4		**MongoDB?**		
5		**Azure?**		
			Beta Version-Total Cost	$153

Being a software based project, the cost breakdown approximate costs of maintaining and deploying our app in the Google Play Store and Apple Store, and the cost for other technologies used such as APIs and database services. The main expenses of the project were the monthly and yearly subscriptions.

The Apple Developer Account yearly subscription is needed in order to publish our app in the App Store and get access to app management, testing, and analytics. The same applies for the Google Play Developer Account, the only difference being that it is only a one time fee of $25.

The remaining costs come from the other technologies such as SightEngine API that are needed for implementing specific features of our app. However, most of the other technologies have a basic tier that is free which we are currently using, but I have provided the worst case scenario for future application growth.

7 Appendices

7.1 Appendix A - Specifications

Requirements	Value, Range, Tolerance, Units
Cost of Database and Deployment	Maximum of $1000
Health Index	75% Accuracy Based on the Machine Learning Algorithm

7.2 Appendix B – Team Information

Byron Mitchell
Worked primarily on the authentication feature and the integration of API services. Graduating as a Computer Engineer who will pursue a Master's degree in Software Engineering at Carnegie Mellon.

Nicole Kwon
Worked primarily on designing and coding the frontend. Graduating as a Computer Engineer and working as a future software engineer at Publicis Sapient.

Harry Li
Worked primarily on designing and coding the backend. Graduating as a Computer Engineer and planning on working as a full-stack or backend developer.

Daniel Li
Worked primarily on the community and inbox features. Graduating as a Computer Engineer and working as a future product manager at Microsoft.