Application
Design Document

Table of Contents

1 Guidelines

There are several technical guidelines we will be following closely for the duration of the application's development. Development will take place in the Xcode 8 IDE using the Swift 3 programming language. Project planning will take place in Trello and version control will be done using Git, hosted on GitHub. We are restricted to using Mac hardware, either through the provided computers in CSIL or through a personal installation of the OS. All developed features will be tested through Xcode's iPhone emulator as our phone-based features, vibration and voice recording -can be tested through the emulator.

Ethical guidelines include the collection of user data, which will remain on the user's local device and *not* on a remote server. Users will not need to make an account to use the application so there is no ethical concern regarding user accounts. Sensitive data, such as the results of the anxiety/depression tests, can be optionally saved to local storage or deleted after the user receives their results. This ensures that users are always aware that their data will be saved onto their phone's local storage while also limiting the possibility of data stealing/leaking.

2 System Diagrams

2.1 Architecture Model

Our architecture model is based off of the repository architecture pattern. All components output their data into local storage, allowing all other components to access the saved data. This will allow the analyzer to capture data produced by both the quizzes and the diary inputs so that they may be analyzed.

In version three, local storage was used to store music tracks, self-therapy data and diary entries. Each feature, using its own view controller, pulled its relevant saved data from local storage and utilized it before saving it back into local storage.

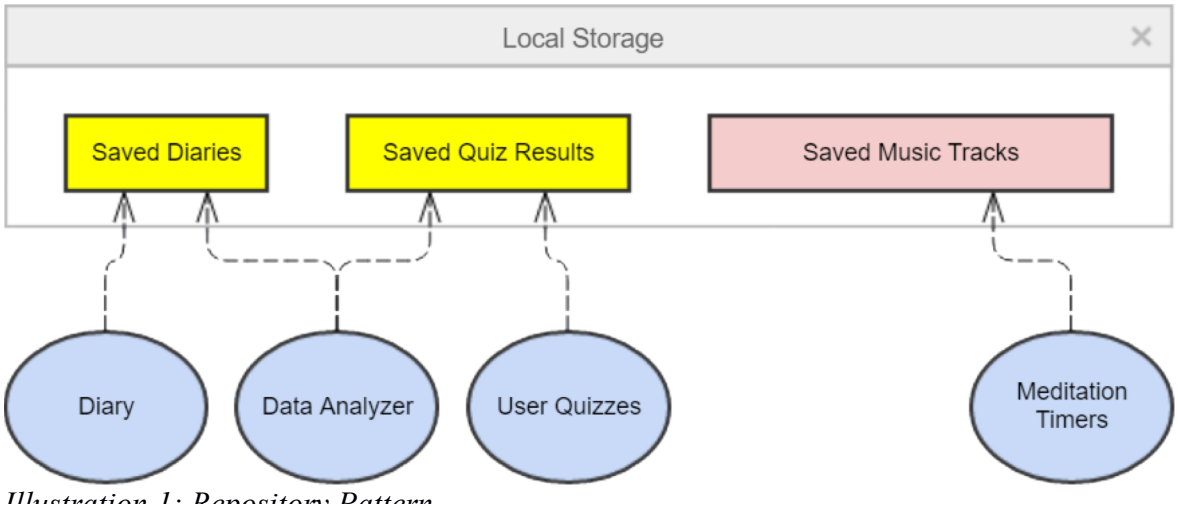


Illustration 1: Repository Pattern

2.2 Interaction Model

Our approach to interaction modeling is through use case modeling. As seen in illustration 2, the various user scenarios are illustrated as interaction states. As shown, the user, whether they are requesting or submitting data, communicates with the view. Said interaction makes contact with local storage and either saves the relevant data or pulls and processes said data.

The final version had users pulling historic data from local storage and interacting with said info using the user interface. Then, the user would save their diary entry/self-therapy session back into local storage for future access.

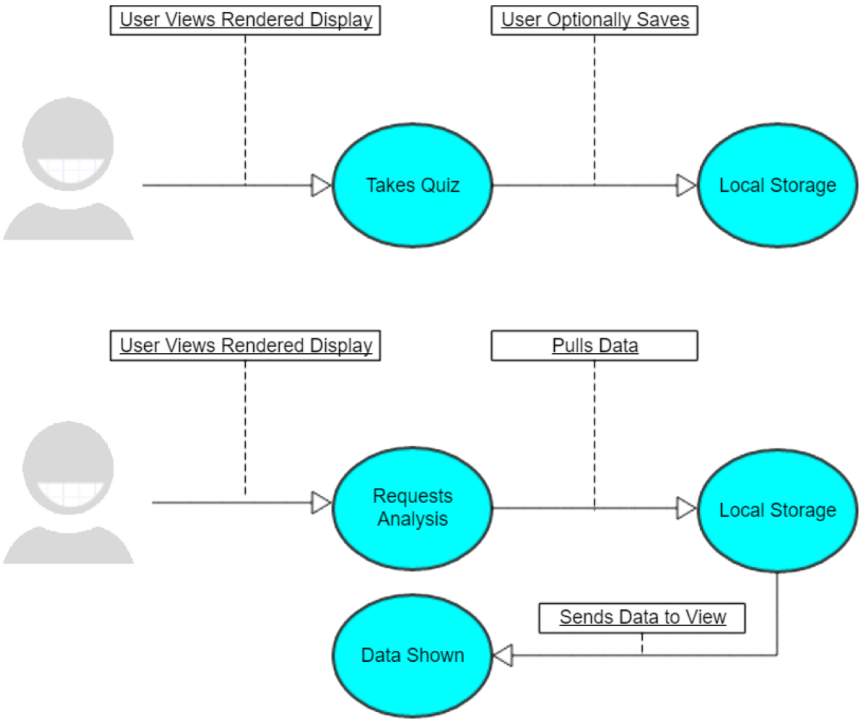
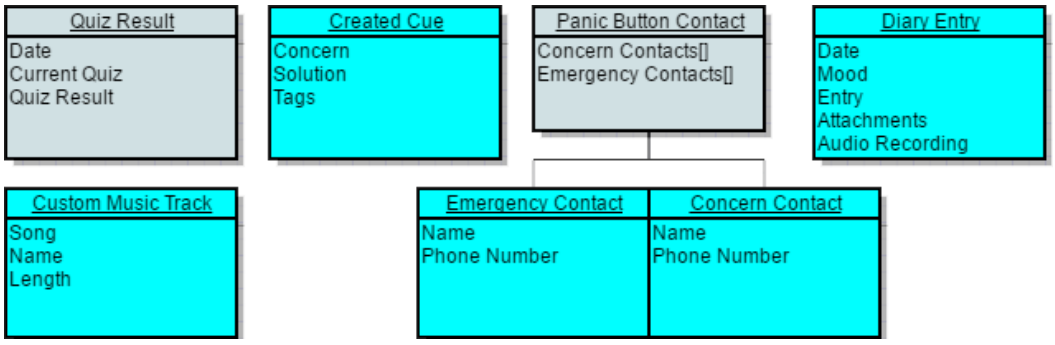


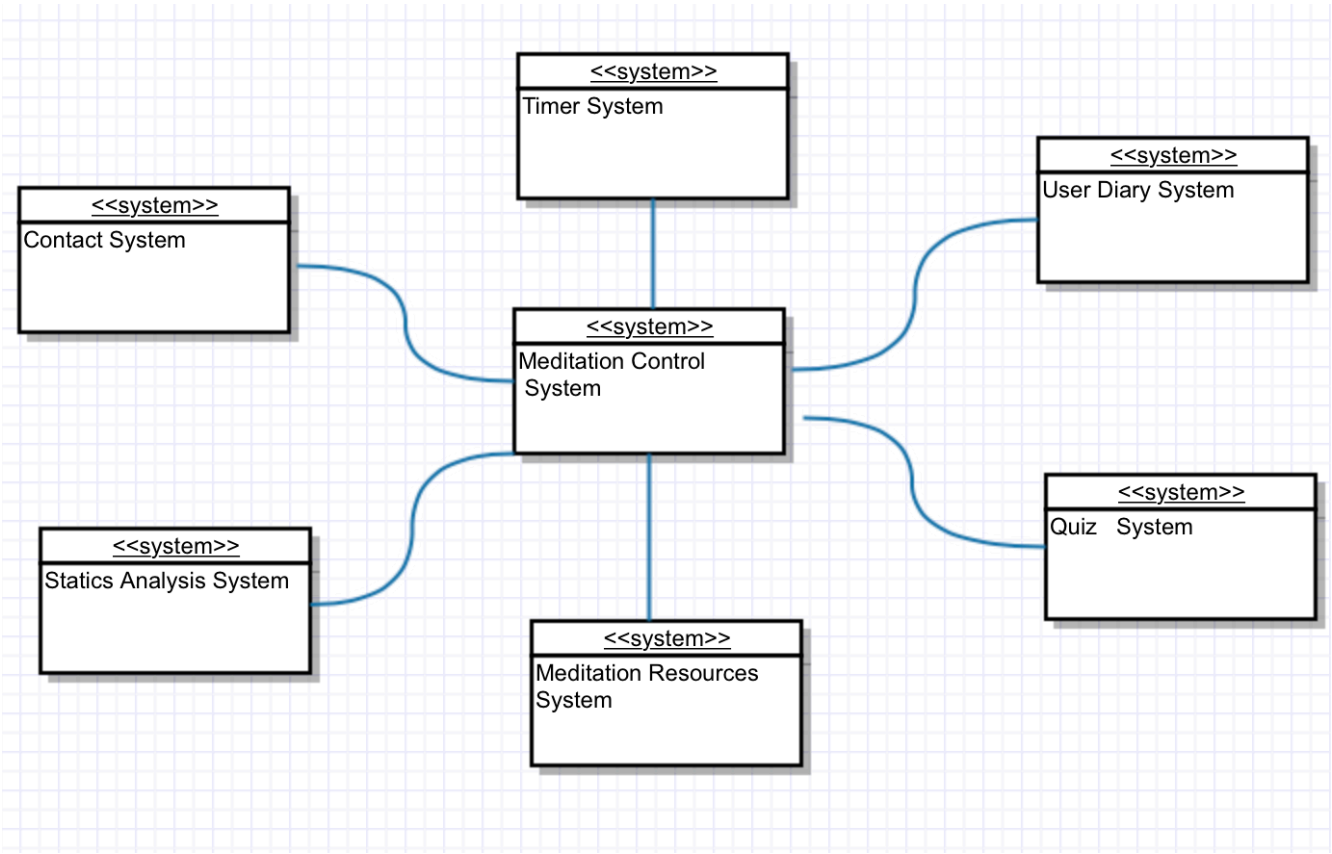
Illustration 2: Interaction Model

2.3 Object Model



In the final version, the panic button was scrapped and the timer object was introduced in its place. All other objects were used—the quiz data saves to local storage, as well as all the other data types—and each individual view controller called and processes the saved data from local storage.

2.4 Context Model



This is the context model of the system which specifies the boundaries based on our functional design. We are going to develop the detailed system by following this model as a blue print and it will somehow reduce our cost and improve the chance of delivering what the users want.

We finalized the app with the same context model as suggested. The model spoke to the individual controllers for each feature/function and interacted with it on the local storage level.

3 Data Requirements

3.1 Data Glossary

This is a summary of all the I / O for the system. All the data is listed below according to 8 main features in the form of tables

3.1.1 Timer Meditation

Attribute Name	Data Type	Associated Function	Description	I / O
1. Timer_Value	Integer	countdown()	Meditation time period set up by user	Input data from user through slider UI
2.Start	Void	Countdown()	Begin the timer for the countdown	No input data required besides the button press. Results in the timer beginning the countdown function.
3. music_List	String	initMusic(music_List) Display(remaing_Time, music_List,stop)	Allow user to choose which playlist they want to play	automatically generated by begin_timed_meditation() , allow user to change value and served as input data for all the behavioural function inside this class

Attribute Name	Data Type	Associated Function	Description	I / O
4. stop	Boolean	countTimer(Timer_Value,stop) Display(remaing_Time,music_List,stop)	Allow user to control the current process	automatically generated by begin_timed_meditation() , allow user to change the value and served as input data for all the behavioural function inside this class

3.1.2 Guided Meditation

Attribute Name	Data Type	Associated Function	Description	I / O
5. session_Type	Char	beginGuidedMeditation(session_type)	Three different mediation type for user to choose	input data from user to choose by select A, B,C on the touchscreen
6. spoken_On_Off	Boolean	sound_Play(spoke_On_Off)	T/F value determined by user to play text reading or not	input data from user to determine sound play on/off
7. user_Session_Rating	Double	rate_Session(user_Session_Rating)	Rate value from 1.0 to 5.0	input data from user to feedback the satisfactory by using scroll bar via the touchscreen
8. current_step	String	omit_Step(current_step)	Tag of each step o the mediation process	input data determined by user

Attribute Name	Data Type	Associated Function	Description	I / O
3. music_List	String	initMusic(music_List) Display(remaing_Time, music_List,stop)	Allow user to choose which playlist they want to play	automatically generated by begin_timed_meditation() , allow user to change value and served as input data for all the behavioral function inside this class

3.1.3 Haptic Meditation

Attribute Name	Data Type	Associated Function	Description	I / O
9. input_Time	Floating	setTimer(inputed_Time) Init_Display(input_Time) init_Vibration(input_Time)	Meditation time period set up by user	Input data from user through scroll bar UI
4. stop	Boolean	countTimer(inputed_Time,stop) init_Display(input_Time,stop)	Allow user to control the current process	automatically generated by begin_timed_meditation() , allow user to change the value and served as input data for all the behavioural function inside this class

3.1.4 Diary

Attribute Name	Data Type	Associated Function	Description	I / O
10. Date	Date	Save_Diary(Date)	Date tag for each diary	Source data obtained from OS system
11. current_Mood	Double	init_Mood_Rating(current_Mood)	Rate value from 1.0 to 5.0 ,representing the mood level of user	input data from user by using the scrollbar UI
12. current_Post	String	init_Text_Box(current_Post)	Text date of user daily info	input data from user by typing the keyboard
13. current_Recording	File	init_Voice_Recording(current_Recording)	users voice record	input data from user by microphone
14. current_Attachments	File	init_Attachment(current_Attachments)	users photo, video file	input data choose by user from local iPhone storage

3.1.5 Quiz

Attribute Name	Data Type	Associated Function	Description	I / O
10. Date	Date	Save_Results(Date, current_Quiz_Resdults)	Date tag for each diary	Source data obtained from OS system
15. quiz_type	Char	request_Quiz(quiz_type) display_Quiz(quiz_type)	Three different quizzes allow user to choose	Input data by user select via selective button displayed on the touchscreen

Attribute Name	Data Type	Associated Function	Description	I / O
16. current_Quiz_Resdults	Floating	dispaly_Quiz(current_Quiz_Resdults) save_Results(date,current_Quiz_Resdults)	Result of quiz as the form of grades	output datâ generated after user finished the quiz and will be displayed if required and stored for further use
17. historic_Result	array	dispaly_Quiz(quiz_type) Save_Results(Date, current_Quiz_Resdults)	Quiz results ordered by data value for further use	Input data when saving and also output data when retrieved for later process

3.1.6 Progress Analysis

Attribute Name	Data Type	Associated Function	Description	I / O
17. historic_Result	array	init_Analysis(historic_Mood,historic_reslut)	all the history quiz results ordered by data value for further use	Input data when saving and also output data when retrieved for later process
18. historic_Mood	array	init_Analysis(historic_Mood,historic_reslut)	all the history mood ordered by data value for further use	Input data when saving and also output

Attribute Name	Data Type	Associated Function	Description	I / O
				data when retrieved for later process
19. user_Selection_Begin	Date	display_Graph(user_Selection_Begin,user_Selection_End) calculate_Data(user_Selection_Begin,user_Selection_End))	Begin Date value allow user to choose for data representation	input data from user
20. user_Selection_End	Date	display_Graph(user_Selection_Begin,user_Selection_End) calculate_Data(user_Selection_Begin,user_Selection_End))	End date value allow user to choose for data representation	input data from user

3.1.7 Cue Creator

Attribute Name	Data Type	Associated Function	Description	I / O
21. current_Concern	String	input_Concern(current_Concern,tags)	users current concern	input data from user by typing via keyboard
22. Tags	Char	input_Concern(current_Concern,tags) search_For_Solution(tags)	Tags for specific concern category and solution related	input data from user by typing via keyboard
23. solution	String	input_Solution(tags,solution) save_Solution(tags,solution)	Solution created by user for further use	input data from user by typing via keyboard

4 Feature Priority

4.1 Iterations

We have evenly distributed all features across the three versions so that each feature has sufficient time to be implemented into the final product:

Version 1
2 Guided Meditation
1 Timer
5 Quiz

Version 2
7 Cue Creator
Timer Meditation
6 Progress Analysis

Version 3
4 Diary
9 Resources
#3 Haptic Meditation

The reason why we assign #2, #1 and # 5 to the version 1 is that those 3 features will perform the basic and fundamental functions, which allows users to experience the main idea of your application also will be used as the prototype for further development. Furthermore, those three features should not be difficult for us to complete before the due.

For the version 2, we will develop 3 more features to improve the completeness of the system structure and which will provide more useful functions for users to a better level so that they can do Hapitc meditation for convenience. Also the progress analysis will allow users to have a better understanding of their anxiety level.

For the last version, we will provide more features as much as we can to give better user satisfactions and more utility. However, we may change our design based on further user feedback to decide whether those features will be necessary or not.