

Sprint 6

Team 13

1. Potential game options

The current tapping game in our application is very basic and does not have much functionality. However, it was the building block of our application as it is through creating that game that we truly understood how flutter and dart worked. In the future sprints, we do hope to make the game better. A few game ideas that we had were creating a small snake game which is fairly simple as there are a lot of resources online that could assist us in doing so. An example is as such:

<https://www.youtube.com/watch?v=9jvJyLhJP00>

Another game that is along the lines of the tapping game we currently have is Flappy bird, we can create a game that has similar functionality as flappy bird and the resources we can use are as such

<https://www.youtube.com/watch?v=vgmVPpFP0fl&t=221s>

2. Recommendation models

We would like for the recommendation models to be more specific to our users. This means that no two users would necessarily have the same recommendations for coping mechanisms. Hence, the way that we would like to improve the recommendations is by increasing more questions in the future sprints which include questions that are more related to personality types, and use the check-in questions to really get an understanding of the kind of personality a user has and what recommendations would be best for them.

Upon doing some research, there are many recommendation models that can be used to help with this process and these include Cognitive Behavioral Therapy. If a user has a specific word such as phobia, or anxiety mentioned in their journals/ checkins, this therapy can help you uncover the underlying causes of your worries and fears; learn how to relax; look at situations in new, less frightening ways; and develop better coping and problem-solving skills.

<https://www.helpguide.org/articles/anxiety/therapy-for-anxiety-disorders.html>

This process is much less invasive as there is no medication required to help individuals face their fears, they are slowly exposed to situations that cause them anxiety. An example of this is, for someone with the phobia of spiders, the recommendation model can set up multiple phases such that in the first phase, the individual is exposed to looking at pictures of spiders. Once they are comfortable enough, they can continue on to the next phase by being exposed to inanimate objects that may be similar in size to spiders, then in the next phase they are recommended to touch the inanimate objects. These phases continue up till a point where individuals are completely exposed to and feel comfortable with the idea of spiders. This will help with issues like phobias and work similarly for anxiety related stuff

3. Hide plaintext passwords in database

Currently in our app, when we store the email and password fields in the database, the password field is stored as plaintext. This is not secure because if the database is compromised then the user's password information will be leaked. In our attempt to make our app equipped with modern app tools and facilities, we want to make the app more secure.

<https://www.edureka.co/community/83656/how-to-hide-password-in-flutter-app>

After doing research, we found that we would need to add to the text field widget to obscure the text in order to hide the plaintext.

<https://stackoverflow.com/questions/70061906/how-to-encrypt-password-while-saving-in-database-in-flutter-sqlite-dart-applicat>

https://www.youtube.com/watch?v=ze_FnNchw4U

A better way to achieve this goal is to encrypt the password when it is entered during sign up. These pages helped understand how to encrypt the data so that when it is stored in the database, anyone looking at the database would see encrypted data instead of plaintext. This would also mean that we would need to decrypt the password when matching to the user input to make sure that they are entering the right password.

<https://www.mongodb.com/docs/realm/sdk/flutter/users/email-password-users/>

While researching this I also found more information on sending reset email/password emails and confirmations. This is something we want to consider implementing in our app if time permits in future sprints.

4. Make journal entries local without database

Currently our journal entries are being stored in the database. We want to make it so that the journal entry data is saved locally rather than having to save every time to the database so that the user will not need access to the internet to access their journal entry data. Our motive behind implementing this is to keep the journal entries easily available

without having to depend on the internet access. We also wanted to learn how local storage works after we were comfortable

After doing a lot of research we found several links that detail how to store different types of data locally. This link is the most useful since it gives step by step instructions on how to easily store data locally rather than in a database.

<https://medium.com/kick-start-fluttering/saving-data-to-local-storage-in-flutter-e20d973d88fa>

As per this source, in order to save the data locally, we will need to find the app data path using path_provider and then we would need to find the local path and store the data to the appropriate app data location.

<https://pusher.com/tutorials/local-data-flutter/>

This link also mentions that the easiest way to store locally is by using path_provider.

<https://levelup.gitconnected.com/the-4-ways-to-store-data-locally-in-your-flutter-app-that-youre-going-to-need-abdafa991ae3>

This link is also useful however I found it to be a little hard to follow. But it also gives steps on how to store locally.

<https://blog.logrocket.com/securing-local-storage-flutter/>

Even though this is not part of the sprint requirement, we went further and also researched on how to keep the local storage secure. Security is always one of the important aspects of the app being considered as good from all aspects. All the modern apps make sure that their user's data is not tampered in any way. In our attempt to build the app to fit the modern app requirements we want to make sure it is secure.

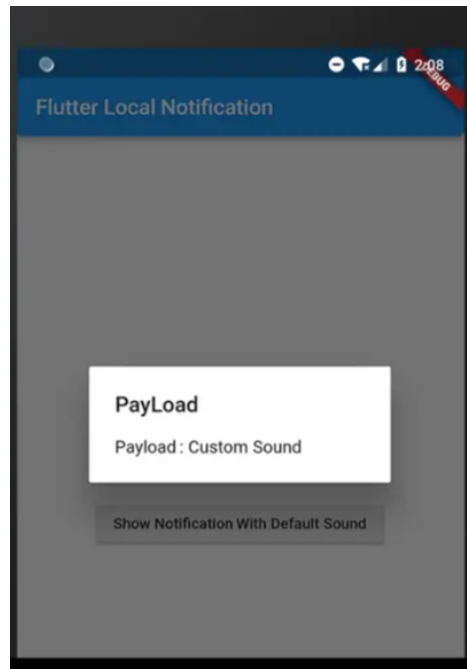
5. Trigger words list:

<div><div><div></div></div><div>List of Trigger Words:</div><div>File Edit View Insert Format</div><div><div><div></div><div></div><div></div><div></div></div><div>100% ▾</div><div>\$</div></div></div>				A		A
A1		abuse	25	failure	45	rage
	A	B	26	finish off	46	ruin
1	abuse		27	flare up	47	sabotage
2	ache		28	frantic	48	sad
3	addiction		29	frustration	49	shatter
4	agony		30	grief	50	shoot
5	angry		31	harm	51	smash
6	anxiety		32	lonely	52	stab
7	annoying		33	heartbroken	53	stress
8	betray		34	hit	54	struggle
9	burden		35	hopeless	55	suffer
10	choke		36	hurt	56	suffocate
11	cruel		37	I don't want to live	57	suicide
12	crush		38	injure	58	sulk
13	cut		39	irritate	59	terminate
14	damage		40	kill	60	threat
15	dead		41	murder	61	trauma
16	death		42	pain	62	triggered
17	despair		43	panic	63	wreck
18	destroy		44	put an end to		
19	die		45	rage		
20	disable		46	ruin		
21	discomfort		47	sabotage		
22	drown		48	sad		
23	eliminate		49	shatter		
24	envy					
25	failure					

6. Notifications pop up.

The purpose of these notifications are to check on the user's state of mind at the current moment. Research of this conducted what libraries or widgets we anticipate to use to send the user a redirection to another screen.

login -> home -> questionnaire right away -> back to homepage



Other things we need to consider is how to track how long its been since the user last filled out this forum. We also should consider if this is an option the user is interested in. Allowing the user to toggle the notification on or off is to be explored as well.

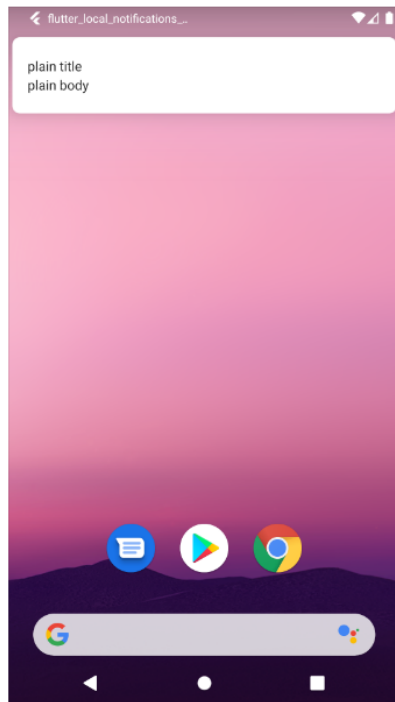
What we will use to reference completing this task is the use of flutters documentation of various widgets.

There are two ways to achieve this notification.

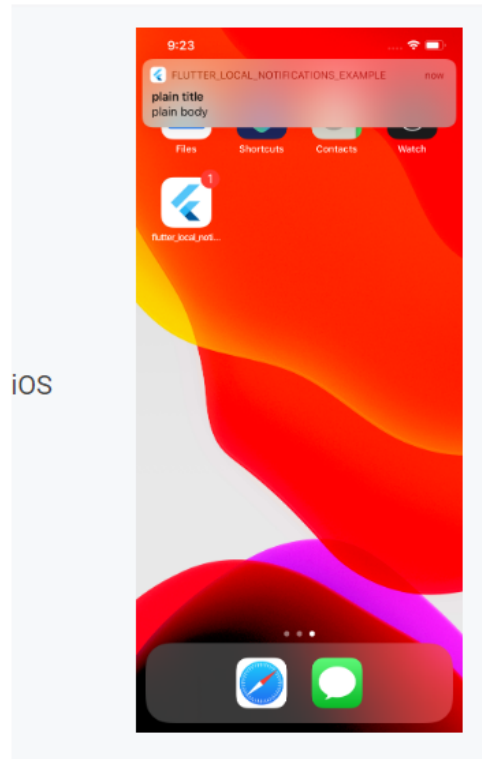
Ideally, we would like to stick with our original plan where the questionnaire pops up after the user log's in.

If we are unable to figure out our first plan then our second plan is to send it locally on the machine.

Android



iOS



websites for reference:

https://pub.dev/packages/awesome_notifications

https://pub.dev/packages/flutter_local_notifications

utilizing notifications for better user engagement.

<https://blog.codemagic.io/flutter-local-notifications/>

7. Spotify connection to flutter.

While doing research there seems to be various bottlenecks on implementing a feature like this. Since we are using an established organizations API a lot of findings have been getting the right permissions as we discuss below.

There are prerequisites when it comes to utilizing Spotify's API.

First each OS system is going to have different installation steps, making changes to the android and iOS folders in the main file.

For both Operating systems the instructions require us to register our app in the spotify developer portal and set up the SDK. All you need is a spotify account and manage your credentials to the appropriate settings.

To use Spotify we are going to get familiarized with Authorization scopes. This allows users to protect their information while using third party apps (Mindly app).

according to the general guide. The web app requires a spotify premium to utilize the Web SDK. This is the only platform that mentions a required membership to use. We will focus our attention on the mobile platforms for now.

website for reference:

[Authorization Scopes | Spotify for Developers](#)

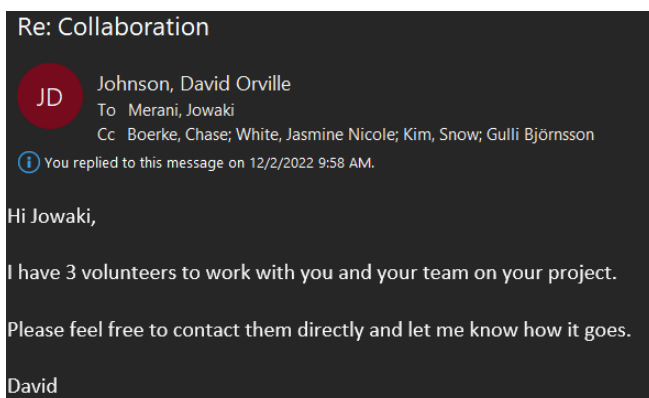
general guide: https://pub.dev/packages/spotify_sdk

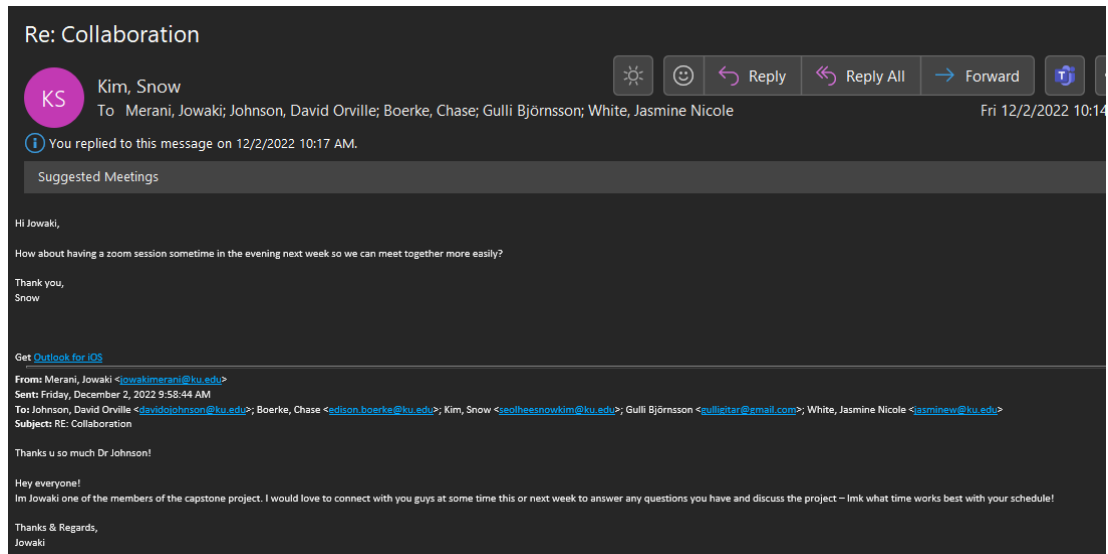
iOS SDK: <https://developer.spotify.com/documentation/ios/quick-start/>

web: <https://developer.spotify.com/documentation/web-playback-sdk/quick-start/>

8. Follow up with the music students:

We have a list of students who we have spoken to and communicated with them the idea of the app. They said they would get back to us with some possible music ideas.





9. Send CAPS documentation:

We sent an email with a documentation of what we have completed which has basic language so it is easy to follow for CAPs. This required us to create a new document with all added pages and functionalities.

10. Create a music Page

For the music page, since we have not gotten music students to help us with tunes to add into our application, we decided to create a basic music page with very limited functionalities. At the moment, when the music icon is clicked, the music page appears with three songs. If the user clicks on the first song, they are taken to a page with functional buttons but with no tunes playing. As we start to record music for our application, the music aspect of this will be added. For now, below are the images of our applications functionalities

mindly

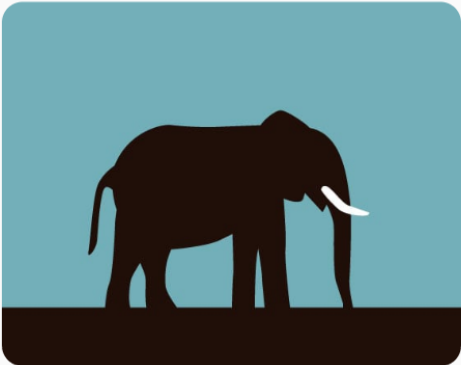
Mindly

Music Page

Tune 1: Song Name

Tune 2: Song Name

Tune 3: Song Name



Song

Sarah

00:0000:00