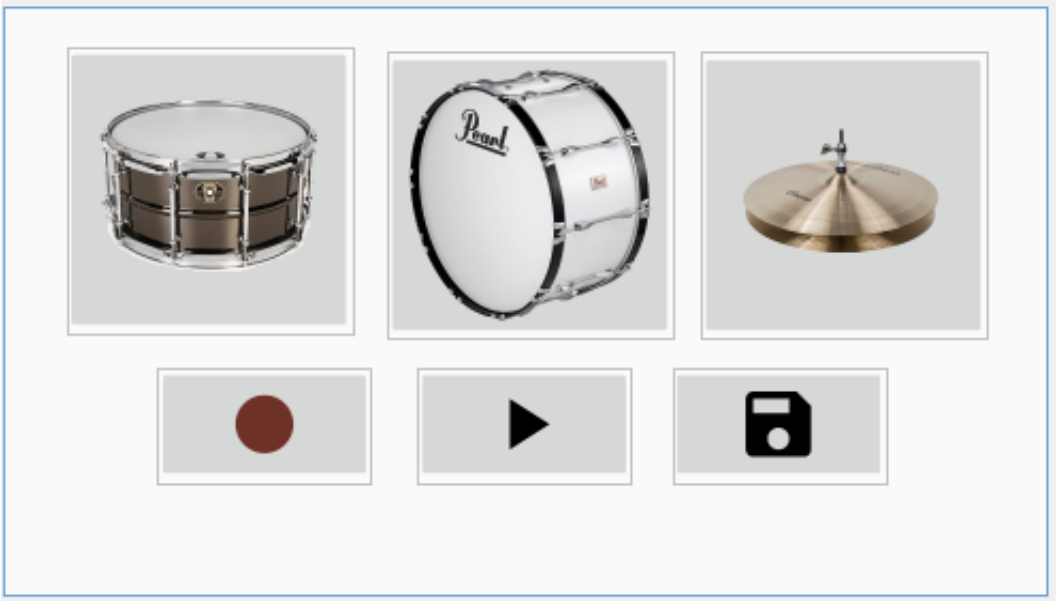
HW5 DrumPlayer

This homework exercises your knowledge of using cloud computing and background processing techniques. The task examines your ability to use Firebase API to store drum beats on the firebase, and retrieve the information from the Firebase and replay the drum score in a WorkManager. The application to be implemented is a DrumPlayer app that allows users to play a drum score, record the beats, and share the plays with each other.

1. The drum app must contain 3 screens: a home screen, a drum practice screen, and a record list screen. This app only needs to run in landscape mode. The home screen contains: a name input, a compose button to navigate to the drum practice screen, and a shared play button to navigate to the record list screen.

2. In the practice screen, the user can play drum beats by clicking three buttons: snare, bass, and cymbal. Clicking the button triggers the sound of the corresponding drum. There should be three buttons which allow the users to record, replay and save the play. Clicking the record button starts recording, and the button changes to the "stop" icon (you can use the vector assets).

Clicking the stop button stops recording (button changes back to record icon). Button status must be consistent after leaving or restarting the app.



No compatible source was found for this media.

* You need to add authors for each music on the list.

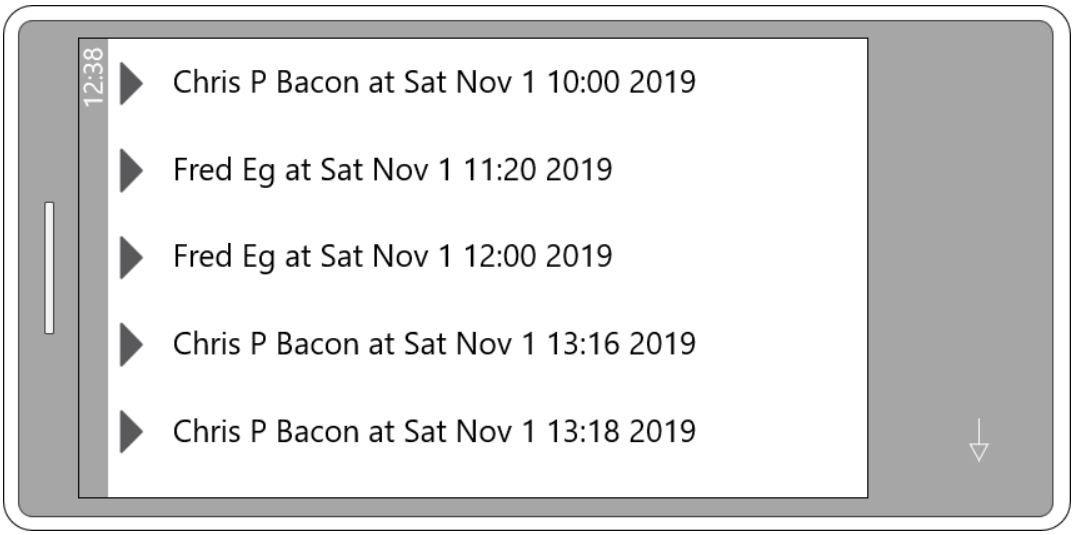
Clicking the play button replays the beats saved during the start and end of the recording. Use the WorkManager to replay the beat sequence. The beats time and sequence must match the recorded play. The recorded beats continue to play even if the users leave the app. The user must be able to record the beats multiple times. The user must be able to replay the last record multiple times.

To play the beats, you can use MediaPlayer to play the snare/bass/cymbal mp3 files when the corresponding buttons are clicked. You need to record the timestamp when the user presses the button. To replay the beats, you are asked to use **Thread.sleep()** in a Worker to pause the thread. For example, when beats are recorded at 10, 100, 300, 500, 1000, 2000 millisecond, then

the sleep interval between two beats will be 10, 90, 200, 200, 500, 1000 milliseconds. To practice WorkManager, you are not allowed to use other scheduling API such as Timer, JobScheduler, or AlarmManager.

Clicking the save button navigates to the record list screen and save the beats to the Firebase. Only the last recorded play should be saved and uploaded.

3. After clicking the save button, the drum play must be uploaded to Firebase. You must properly use coroutines and async to upload the record in a ViewModel. After successfully uploaded, you need to show a Toast notification. The app navigates to the saved list after clicking the save button. The record list screen must contain a RecyclerView to show a list of saved records in the Firebase. Each record item contains a play button and a record name (user name + timestamp). Clicking the play button on each item plays the corresponding record. The record must continue to play after the user leaves the app.



The user must be able to delete his own play by performing the swiping-and-deleting gesture. The users must not be able to delete other's records. The list must be consistent even if the user leaves or kills the app.

4. For this assignment, you don't have to implement the action bar. The

navigation graph and GUI of the app must be properly designed and implemented. Provide instructions or hints by using Toast or Dialogs for improper interaction (e.g. empty user name). The navigation of different Fragment should be direct and intuitive.

5. All required screens (Fragments) must be accessible from one to another. Make sure the user is not stuck at one fragment and won't be able to navigate to other screens.

6. The data, including but not limited to, the user name, beat time and sequence, recording status, playing status, and the record lists must be consistent throughout the app usage. You can assume the user does not power off the device. But the data must be consistent after leaving or restarting the app.