UNIVERSITY OF REGINA

MOBILE COMPUTING

CS 855

Startle (A Jump-Scare App)

Author:
Mehak Jeet Singh Walia
Student Number:
200420810
E-mail:
mwh866@uregina.ca

Supervisor:
Dr. Trevor TOMESH

December 11, 2019

CONTENTS

Abstract

The following document aims to lay out a road map of the project development from the start to finish in detail. By the end of it, the reader will have a complete understanding of the features and design of the application that I have developed. It is an entertainment-based prank app intended to unnerve the user or make them laugh while using a mobile phone. It catches the user off-guard by showing them something scary or funny. The goal behind making this application was to learn basic functionalities of Android along the way. While building the application, I became familiar what to use in a given situation achieve a desired functionality. Also, I learned how to use GitHub, make repositories and upload the code onto the GitHub.

I. Project Description and Flow Control

The application that I am developing is a variation of the Shock App [2] available on the Google Play. The application has a straightforward GUI and is fairly simple to use.
Upon opening the application, the user selects the image and the audio clip of his own choice from a list of options. The user also has the option to download photos from the web into the app as well, hence making it more personalised and fun. After selecting the image, he then clicks on the "Prank" button and switches to some other app. The next step is to click on the notification (generated after the prank button is clicked), to trigger the prank phase. Now, hand over the phone to your intended target (friend/family member) and let the pranking begin. Now if the victim clicks anywhere on the screen, the image pops up unexpectedly accompanied by the audio sound.

I have also included a flow-control diagram to better understand the user-flow of the application (Fig. **??**).

II. Inspiration

While looking out for a project idea, I came across various interesting application ideas. Since I had no prior knowledge in Android and Kotlin, I did not want to take up a project which was too daunting or detailed for a beginner. Instead, I wanted to choose a project that would touch a breadth of topics and concepts. So, the basic ideology was to find a project that covers more concepts rather than covering only a few topics in extreme detail. Also, I wanted to learn the functionalities and methods that act as the basic building blocks of almost any Android application.
Finally, when I stumbled upon this application, it had a perfect mix of the topics I was hoping to cover. The application encompasses all the basic techniques and skill-set required to efficiently develop an intermediate-level application on the Android platform. This project has helped me gain knowledge and learn many things apart from the concepts taught in the class.

III. Design and Implementation

The designing phase of the application was fairly straightforward. I built this application with the help of an online course[3], so the designing was already well laid out for me. Observing how the course approached the design phase was very insightful. The instructor broke down the whole application into smaller modules so efficiently and reasoned everything with logic. This in addition to building the application taught me how to approach big projects and handle the design phase of an application. These are the following modules used to build my application:

A. Surprise Activity

The first thing that I worked on was the Surprise Activity. This is the screen that appears to take the user by surprise when the user clicks anywhere on the screen.
First, I pre-selected an image asset and an audio asset, and then programmed the surprise screen to come right away when the application is launched. The entire screen is filled with the image view and the title bar is made hidden for the image to fill the whole screen. After that, I added the functionality to let the Surprise Screen appear only when the user has clicked on the screen. Fig 3 is the surprise screen which is used to scare the victim.

B. Main Activity

After making the Surprise Activity. I started to design the Home Screen of the application. I have used Constraint Layouts and made them clickable to behave like buttons. I have added ripple effects on the top of the constraint layouts that come into play every time the surfaces are clicked. Fig 4 shows how the application's home screen looks like.
Then I made a notification[11] to be generated (Fig 5) upon clicking on the Prank Surface. After that, I linked the Surprise Activity screen to come on top of the other applications when the notification is clicked. Then, I added a toast "Ready" (Fig 6)that lets the user know that Surprise Screen is one touch away.

C. Shock Utils class

This class is a helper class. This dynamically allocates the id to the resources (images and audios) and then returns the URI (Uniform Resource Identifier) which is the absolute path to the location where the object is stored.

D. ImageStorer class

The next thing to handle was as to how to store the images in the application. The images are basically categorised into two types: asset and non-asset. Assets are the images which are pre-installed in the application. They are from the Drawable folder in the resources and are treated differently from the images downloaded from the web. The images downloaded from the web are stored in an internal file using sharedPreferences[10] into a file as a Bitmap.
I have created a class by the name of "ImageModel" which handles all the images either assets or non-assets. It holds the image id, the location where it is stored and whether it is an asset or not, for all the images.
Glide[9] functionality is used to load bitmaps efficiently into an image view.

E. Audio Storer class

It has the very same functionality and designing as the "ImageStorer" class. A folder by the name of 'raw' is created in the resources folder. It contains all the pre-installed audios. The text-to-speech audios are treated differently and not played using MediaPlayer. The TexttoSpeech library [4] is used to synthesize speech from text. The audios are also stored into the Shared Preferences. Similarly, the "audioModel" class handles all the images.

F. Menu Option

The next order of business was to develop a Menu item box. This menu box contains two options. One is to add an image and another one is to add a sound. Then I inflated the image pop-up menu from the Menu-item box followed by the add-audio pop-up 7. However, the actual functionality was added later in the development phase.

G. Add Image and Add audio pop-up

Then I worked on building pop-ups for add image and add audio option item in the menu box. Fig. 8 shows the add image pop-up. The okay button is used to fetch the text from the text box and pass it to the image downloader function.
Fig. **??** shows the add audio pop-up. Text entered is spoken with the help of text to speech library.

H. Image Picker Dialog Fragment Class

This class is used to create a Dialog fragment[7] of all the images in the application. Fragments are basically a window that floats on the top of the current activity window. The imageView on the home screen is linked to the image dialog fragment. The fragment uses a recycler view to show all the images. A Recycler View[5] is a container used to hold and display objects (data-sets) dynamically. This class is handled with the help of an adapter class called "ImagePickerAdapter". It inflates all the items in imageModel class and implements an interface of the name "callback" to return the inflated images to the fragment. Fig 10 shows the populated recycler view with all the audios.

I. Audio Picker Dialog Fragment Class

The Audio Picker Fragment class has a similar design and functionality as the "ImagePickerDialogfragment" class. Both the fragments use the same Recycler view and it is populated dynamically. "AudioPickerAdapter" is used to populate all the audio items into the fragment. Fig 11 shows the populated recycler view with all the audios.

J. Update the User Interface

After selecting the image or an audio from the fragments, we need to get the selected items and update the home screen with the current selected image and audio dynamically. So, this function uses Local Broadcast Manager[8] to tell the main activity that the user has made an update.

K. Other tasks

The play surface on the home screen has a play icon which is dynamically changed to the pause icon for the duration the audio is played(Fig 12). The function to add an image from the web were also added in the last. Also, the text to speech functionality was realised using the text to speech class in the end.

IV. Timeline

There were a series of intricate steps involved in the development phase. Even after devising a proper action plan before development phase, I was not able to complete and meet some of the deadlines. Here is a list of the desired milestones and the time by which I said they will be completed:

- Milestone 1: Requirements, Framework and GitHub setup – November 10th, 2019
- Milestone 2: Design phase – November 17th, 2019
- Milestone 3: Coding phase – November 23th, 2019
- Milestone 4: Enhancing the GUI design – November 28th, 2019
- Milestone 5: Testing phase – December 1st, 2019
- Milestone 6: Project done by – December 8th, 2019

Although I was able to develop the application in the said time, I was not able to complete the documentation write-up in the given time. Also, I started developing the application a week later than the proposed time in the Gantt chart(Fig. 1).

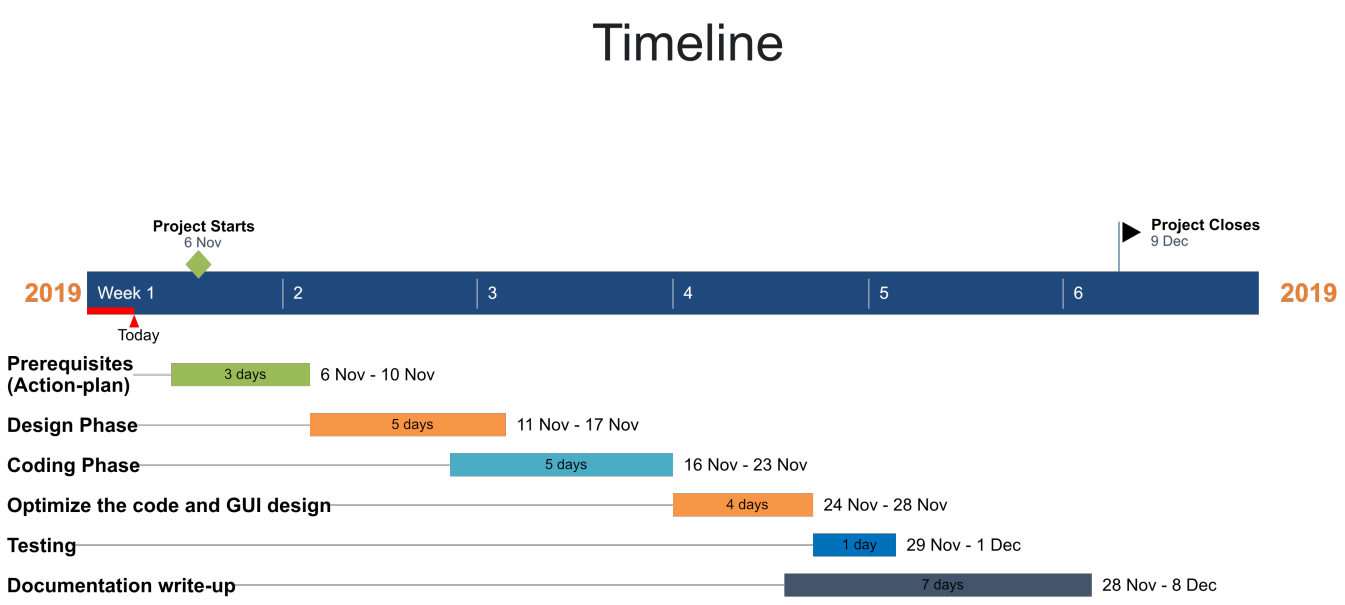

Figure 1. Gantt Timeline Chart to track the software development life cycle.

V. Limitations

There were some interesting ideas that I had which would have helped in increasing the user interaction and making the application much more interesting. Some of those are:

- The user can specifically set the time/timer after which the image will pop-up. If the screen timeout timer is set, the phone gets locked even if it is waiting for the user to touch the screen.
- A user-authentication portal so that only the actual user can access the application and does not ends up getting pranked on his phone.
- To add a feature to cancel a prank that is already in action as the user might wish to undo it or postpone it.

I was looking forward to at least implement one of the above-stated functionalities but was not able to implement any because of time constraints.

VI. Conclusion

This was the first main project that I built using Kotlin and Android Studio. Kotlin is a new programming language and because of that, it was very hard to find resources to help me when I was stuck with any issues. Although I followed an online course to help me guide through the project, I had a tough time understanding the Android classes and libraries. I really wanted to add at least one additional functionality on my own, but was not able to add any. But even then, selecting this as my project was all worth the trouble. I mainly learned how to use Recycler Views to display dynamic content and Shared Preferences to store objects. The experience that I have gained is invaluable and will make it relatively easy for me to build Android applications in the future.

Appendix A
Screenshots

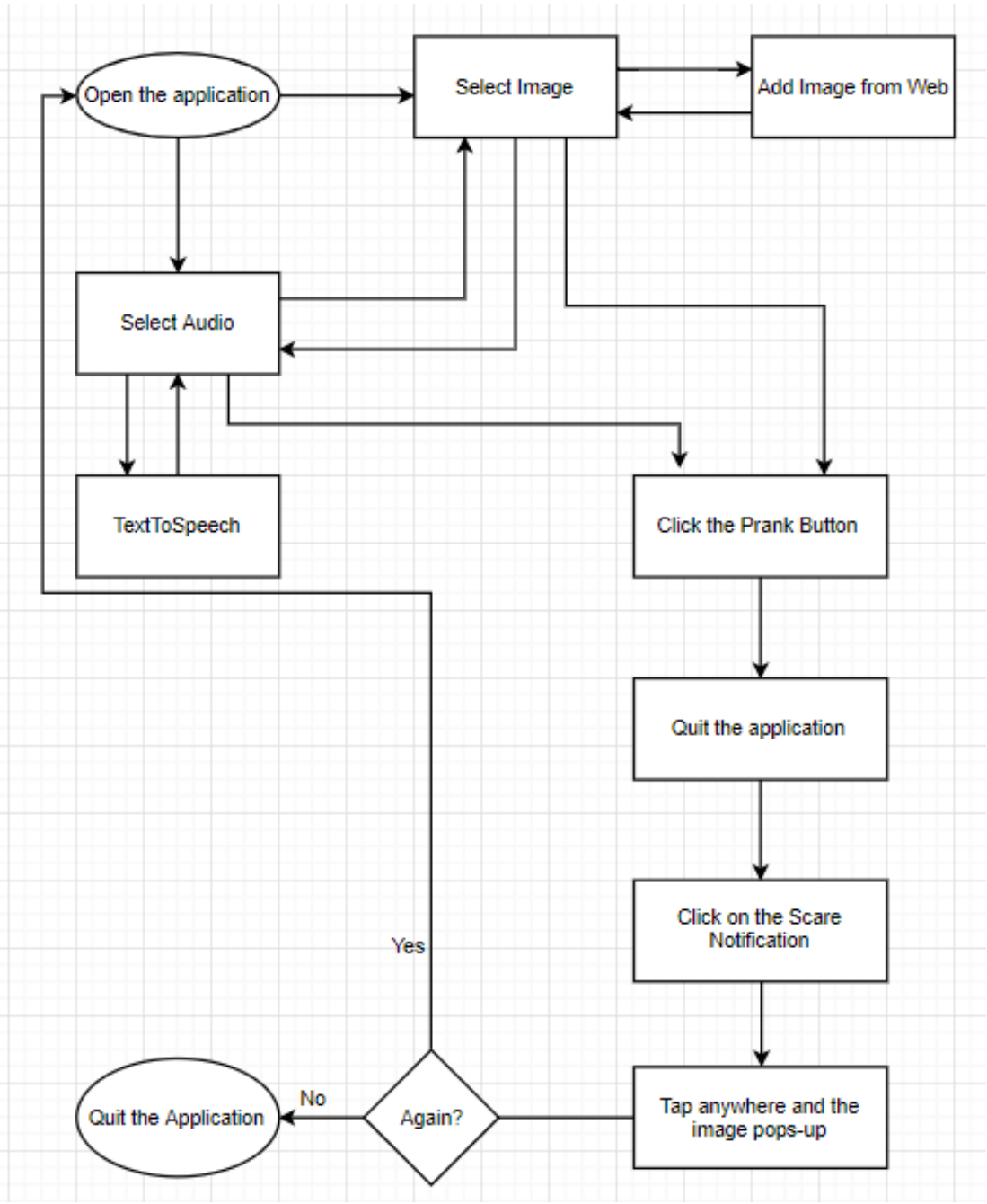

Figure 2. User flow-control diagram.

Figure 3. This is the Surprise Activity.

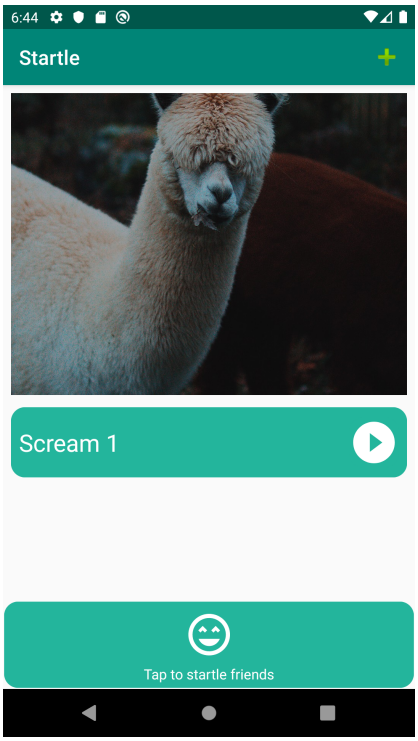

Figure 4. Application Home Screen.

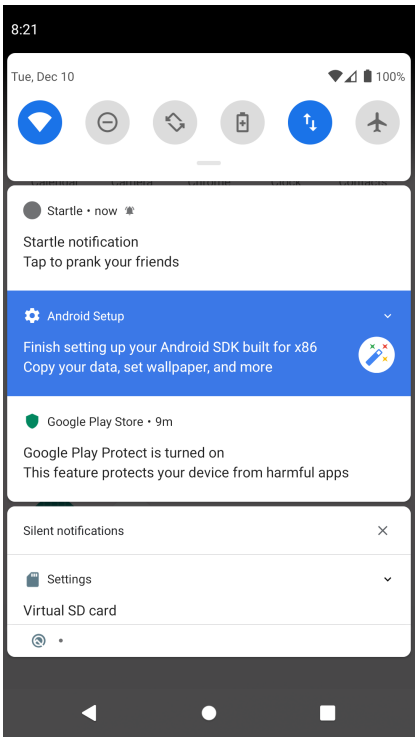

Figure 5. Notification created.

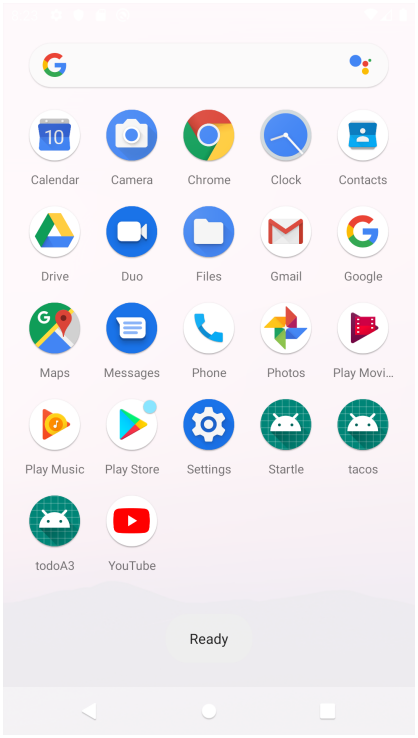

Figure 6. Ready toast is displayed.

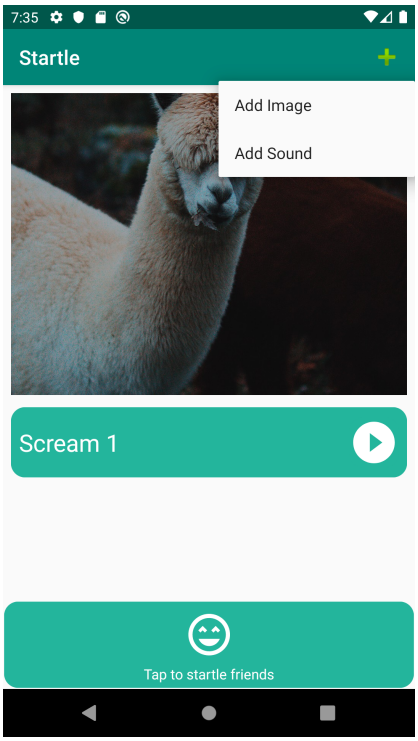

Figure 7. Menu-Item Box.

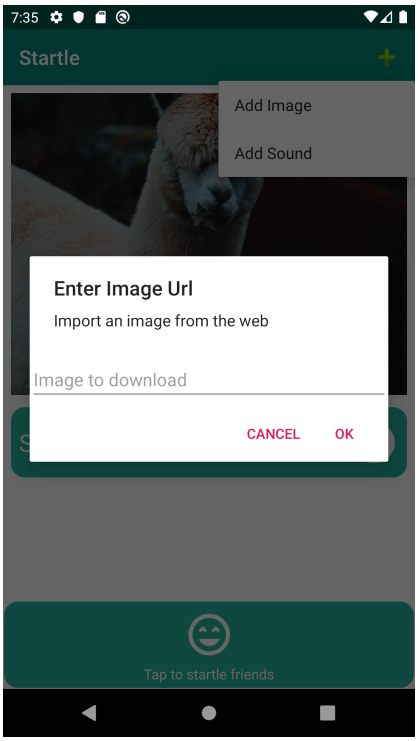

Figure 8. Add image from web by entering the URL.

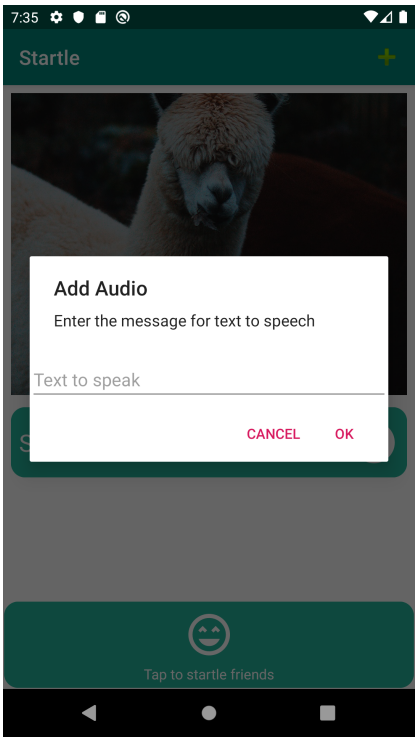

Figure 9. Add text which you would like to speak.

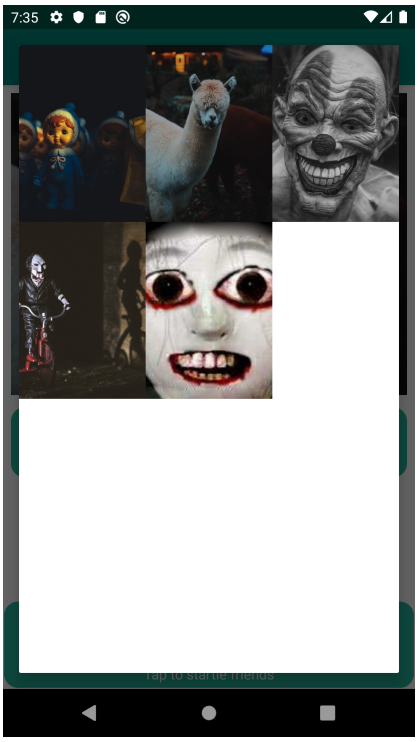

Figure 10. Image Dialog Fragment.

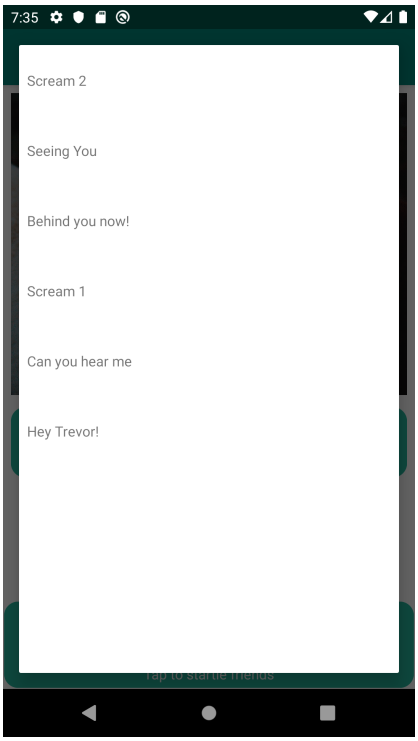

Figure 11. Audio Dialog Fragment screen.

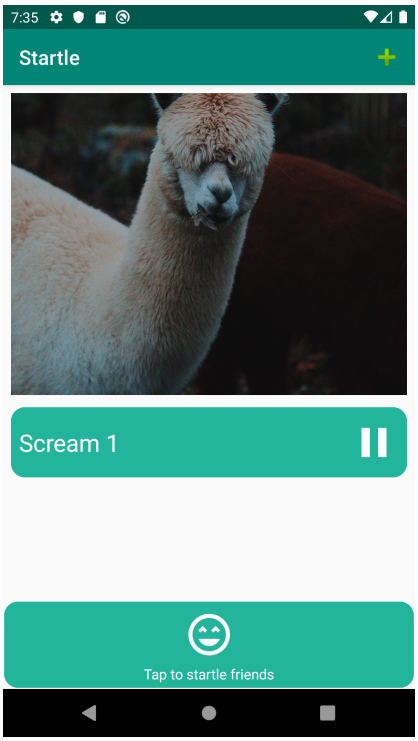

Figure 12. Play icon is changed to pause icon while the media is playing.

REFERENCES

[1] ”waliamehak - Overview”, GitHub, 2019. [Online]. Available: https://github.com/waliamehak. [Accessed: 10- Dec- 2019].

[2] Play.google.com, 2019. [Online]. Available: https://play.google.com/store/apps/details?id=com.tsgotem.shockhl. [Accessed: 10- Dec- 2019].

[3] ”Intermediate Android: Jump Scare App Clone In Kotlin — Crimson Altima — Skillshare”, Skillshare, 2019. [Online]. Available: https://www.skillshare.com/classes/Intermediate-Android-Jump-Scare-App-Clone-In-Kotlin/369357031/projects. [Accessed: 11- Dec- 2019].

[4] ”TextToSpeech — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/reference/android/speech/tts/TextToSpeech. [Accessed: 11- Dec- 2019].

[5] ”Create a List with RecyclerView — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/guide/topics/ui/layout/recyclerview. [Accessed: 11- Dec- 2019].

[6] N. Smyth, Kotlin / Android Studio 3.0 development essentials.

[7] ”DialogFragment — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/reference/android/app/DialogFragment. [Accessed: 11- Dec- 2019].

[8] ”Broadcasts overview — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/guide/components/broadcasts. [Accessed: 11- Dec- 2019].

[9] ”Loading Large Bitmaps Efficiently — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/topic/performance/graphics/load-bitmap. [Accessed: 11- Dec- 2019].

[10] ”Save key-value data — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/training/data-storage/shared-preferences. [Accessed: 11- Dec- 2019].

[11] ”Notifications Overview — Android Developers”, Android Developers, 2019. [Online]. Available: https://developer.android.com/guide/topics/ui/notifiers/notifications. [Accessed: 11- Dec- 2019].