Project – iRemember

*An app for telling **YOUR stories***

Objectives:

iRemember captures the stories of your life, connects them to the places where they happened, and lets others see locations in a new way -- through your eyes.

With iRemember, users capture their stories in a variety of media: text, audio, photos or video. After capturing the story, the app allows users to give it a title and to annotate it with text and location information. For example, a graduating University of Maryland student might use iRemember to capture stories of a classroom in which they learned something very important to them, the library where they met their future spouse, an auditorium in which they heard an important speech, a laboratory in which they made an important discovery.

As more and more people tell and share their stories, the set of stories will become interesting and useful to other people as well. For example, first-time visitors to the University of Maryland could use iRemember to search for and replay the stories of others, seeing the campus through the eyes, hearts and minds of those whose lives were changed by the time they spent there.

For this project, you will develop your own version of iRemember, based on skeleton files we provide to you. For this class, iRemember will only support a single user. All this user's data will be stored locally on the device, and no data from any other user can be viewed. Students taking the other courses in our Mobile Cloud Computing with Android Specialization, however, will be able to add additional functionality to expand the app, eventually allowing user stories to be stored to the cloud, and allowing different users the search for, retrieve and replay stories created by other users.

This project is logically divided into three parts, which are:

Part 1: Capturing Multimedia Stories

The goal of this Part is to allow users to capture and replay multimedia stories. You will receive a skeleton for the iRemember app, and an apk file containing a ContentProvider that stores the app's data.

Part 2: Managing Story Data in a ContentProvider

The goal of this Part is to implement the ContentProvider that manages and filters a user's stories. In this step, you'll replace the apk file, containing the app's ContentProvider that you used in Part 1, with your own ContentProvider implemented from a skeleton file. After completing Part 2, you will submit your app's code for both Part 1 and Part 2 along some screenshots for Peer Assessment.

Part 3: Improving the User Interface

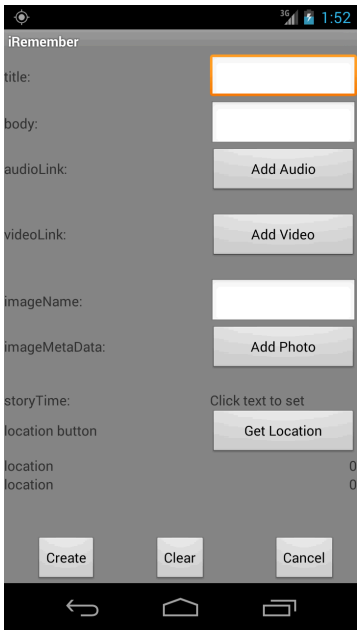
Parts 1 and 2 implement the app functionality. Your grade on the project will be based on your ability to get this basic functionality working. This Part should be done only **after** you have submitted your solution to Parts 1 and 2.

As you will certainly notice, the interface and usability of this app are quite poor. Therefore, in this Part you can unleash your creativity to improve the look and feel of the app, improve its usability, add functionality and make any other changes you think will improve the app. **This Part is optional and ungraded**, but we hope you'll do this Part and share your improvements on the course forum.

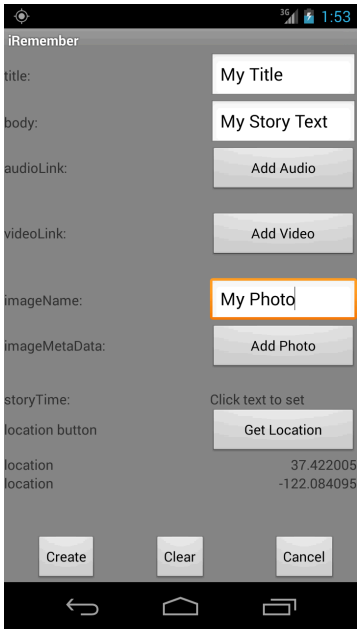
iRemember Overview

When the app starts you'll see a simple interface as shown below. This interface includes an "Add" Button for adding a new story, a ListView, showing current stories (initially empty), and a "Filter" EditText box that allows users to filter the list of current stories based on a text match against each story's title.

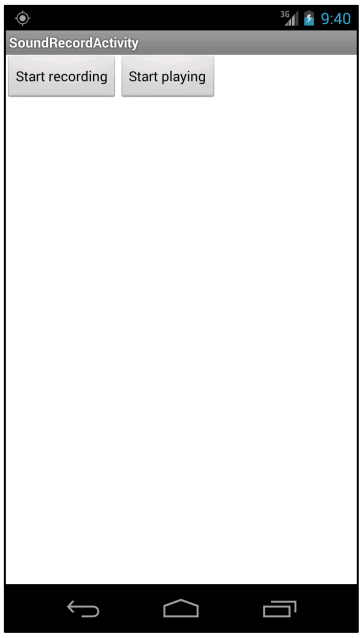
When the user hits the Add Button, a new interface will be shown, allowing the user to enter story information. This interface appears below.



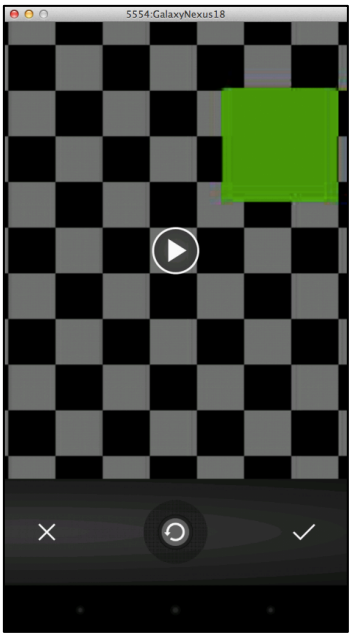
The user can enter text into several of these boxes, indicating information such as the story's title, its body (text description), and a caption for a photo used in the story. The user can also enter a date for the story and can capture the story's location.



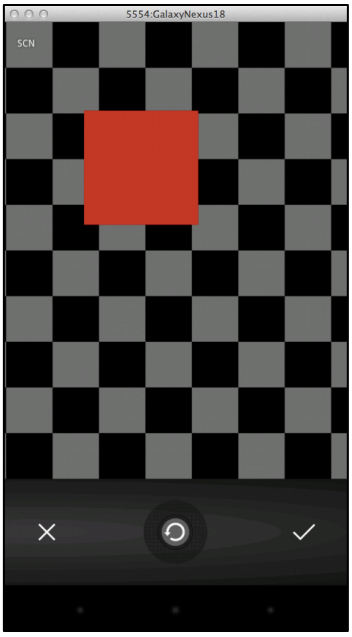
In addition to this, the user can also attach an audio recording, a video recording, and a photo to the story by clicking on the Buttons labelled, "Add Audio", "Add Video" and "Add Photo", respectively. If the user clicks on these Buttons, new interfaces appear allowing the user to capture the particular multimedia element. For example, if the user clicks to add Audio, the following interface appears.



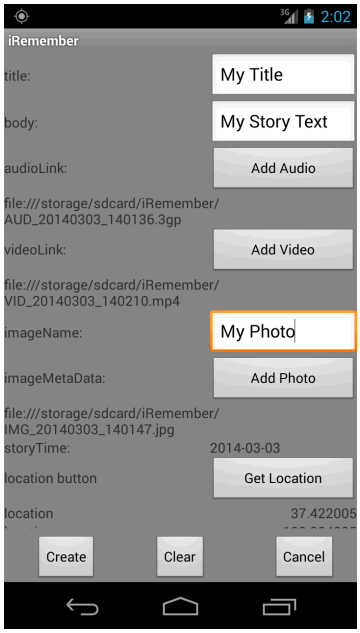
If the user clicks to add Video, the following interface appears, allowing the user to record video (this example is using an emulated camera).



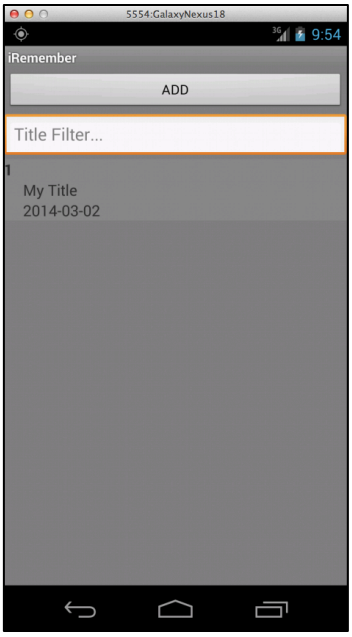
Finally, if the user clicks to add a Photo, the following interface appears, allowing the user to take a photo (this example is using an emulated camera).



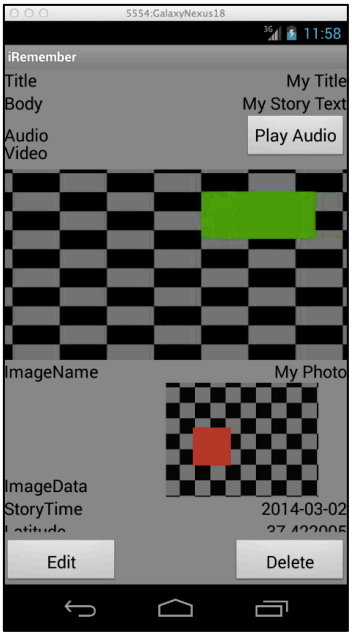
After adding this information to the story, the app's interface might display the following information.



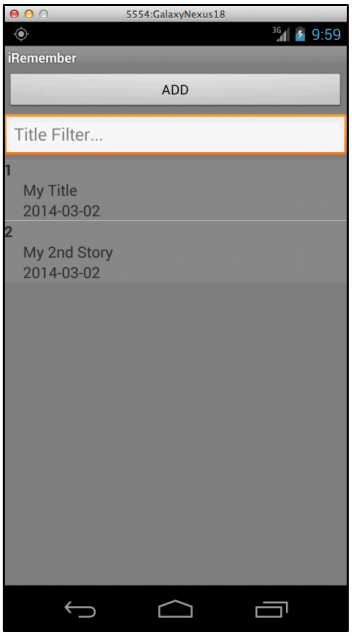
If the user now clicks on the "Create" Button, the story will be stored in the app's ContentProvider and the app's story ListView will be updated to include the new story.

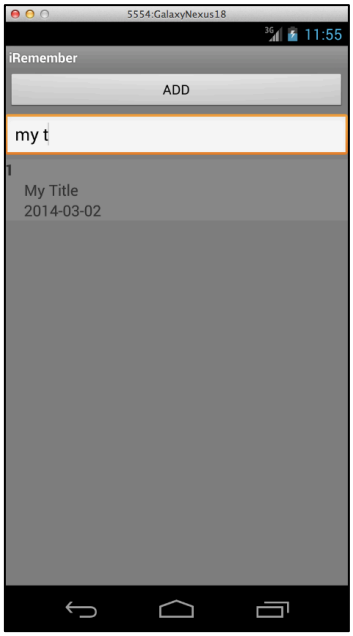


If the user later clicks on a story in the ListView, a new interface appears allowing the user to view and play the story, or to edit or delete it.



Finally, after the user has added multiple stories, he or she can use the filter box to filter stories by their titles.





Implementation Notes:

1. Take a look at the video as a reference for how your application should work once completed.
2. Download the application skeleton files and import them into your IDE.
3. For Part 1 we are providing an apk file that implements a ContentProvider that is used by this app to store and retrieve story data. That file is in iRemember/SourceFiles/ Skeleton/ iRememberContentProvider.apk. Install this file on your AVD, by opening a terminal window and entering the following adb command:
   % adb install ***pathToFiles***/iRemember/SourceFiles/Skeleton/iRememberContentProvider.apk
4. Look for comments containing the string "TODO" and follow the associated instructions. Although the app has several files, your modifications will be limited to two files, StoryViewFragment.java and CreateStoryActivity.java.
5. Study the MediaStore API in the Android Developer's Reference for more information on how to use built-in Intents to start the camera for video or image capture:
   a. http://developer.android.com/reference/android/provider/MediaStore.html

Warnings:

1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test you app against a similar AVD.
2. Your AVD must have access to a camera. In our testing, we configured the AVD to have both emulated front-facing and back-facing cameras.
3. Your AVD must have space allocated for its SD card.
4. If you are having problems running the app in the emulator, restart the emulator without using a stored snapshot.
5. In our testing with actual devices, we encountered bugs with certain HTC and Sony Ericsson video and camera apps. Be aware of this if you are developing on an actual device.