Team #1

Team Member Name	PID	UCSD Email ID
Derek Jow	A12595924	djow@ucsd.edu
Seung Suh	A12960850	
Yungmoo Song	A91411506	yus086@ucsd.edu
Pushpak Raj Gautam	A13708485	p1gautam@ucsd.edu
Trevor Petersen	A92023829	tpeterse@ucsd.edu

Milestone 1 - Delivery Phase

Refer the grading rubric (link: https://csemoodle3.ucsd.edu/mod/page/view.php?id=1326) for more details.

Here we are following a sample repository (github.com/CSE-110-Winter-2018/Default-Repository) to provide sample links for code snippets, branch etc. as per requirements.

Remember to push this document in the *docs* folder of your project in the branch "milestone1_delivery", like: github.com/CSE-110-Winter-2018/Default-Repository/blob/milestone1_delivery/docs/Milestone1_DeliveryPhase.pdf

Software design

Checkout a new branch "milestone1_delivery" , push the final code to it and provide a link to the code folder of your team project repository on Github
Sample link: https://github.com/CSE-110-Winter-2018/Default-Repository/tree/milestone1_delivery

Single Responsibility Principle:

1. MyMediaPlayer follows SRP because it is solely responsible for loading and playing media files. MyMediaPlayer delegates calls through the interface MediaPlayerStateChangeListener which is implemented by MainActivity to handle UI events separate from the MediaPlayer. When the media player changes a state, it notifies the respective event to MainActivity. Thus, our mediaplayer is only responsible for changes in media, while our MainActivity handles the UI.
    a. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/mediaplayer/MyMediaPlayer.java#L347-L363

    b. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/mediaplayer/MediaPlayerStateChangeListener.java
    c. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/MainActivity.java#L100-L101
2. Loader follows SRP. We have created a strategy pattern - a family of algorithms - for determining the best way our application can load different songs for playback. Our loaders follow SRP because they are solely responsible for extracting, parsing, and returning Song lists to be played in the MediaPlayer. MediaPlayer then delegates calls through the loader.
    a. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/tree/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/loaders
    b. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/mediaplayer/MyMediaPlayer.java#L250
3. DatabaseHelper follows SRP. It is solely responsible for retrieving, removing, and adding data into our SQL Database. Our Song object then accesses the database through a load method that reads from the database, and then song is responsible for updating its own data.
    a. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/database/DatabaseHelper.java
    b. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/Song.java#L68-L92

Don't Repeat Yourself:

1. StringArrayParser.toString() is a good example of DRY. The method helps in storing new entries into the sqlite database. To do something with the database, an SQL statement has to be formed but adding commas and setting up strings from variable names again and again is not a good idea. Using StringArrayParser.toString() (which is a static utility method) really helps in such a case.

    Code -
    a. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/utility/StringArrayParser.java#L11-L17

    Usage -
    b. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/database/DatabaseHelper.java#L271
    c. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/database/DatabaseHelper.java#L245
    d. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/database/DatabaseHelper.java#L297

2. LocationParser is another good application of the DRY principle. Its purpose is to provide real location names based on a given set of coordinates. This has been used in two places. One in the RealLocationProvider and the other in MockLocationProvider. Initializing the same geocoder and then

extracting addresses from the returned list twice is not a good idea, and this is where LocationParser comes in handy.

Code -

a. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/location/LocationParser.java

Usage -

b. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/location/RealLocationProvider.java#L88

c. https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/main/java/com/cse110/ucsd/flashbackmusicproject/location/MockLocationProvider.java#L22

(optional) Screencast - demo of all milestone requirements

Screencast of your app either on a real Android device or an emulator covering all the required points in the "demo" section of the grading rubric (link: https://csemoodle3.ucsd.edu/mod/page/view.php?id=1326).
Note: The screencast is optional, the actual demo will be in-person, details about which shall be posted soon.

Helpful link to know more about screencasts and ways to capture it:
http://edutechwiki.unige.ch/en/Screencast

Testing

Link(s) to all the JUnit and/or Espresso tests and logs of untestable non-trivial methods.

Something like this:
https://github.com/CSE-110-Winter-2018/Default-Repository/blob/milestone1_delivery/MyApplication/app/src/main/java/com/andriod/helloworld/MainActivity.java#L23-L27
Notice how specific line numbers in the concerned source code file have been mentioned in the URL above.

Test for database
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/JUnitTest/TestForDatabase.java#L38-L158

Test for Location
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/JUnitTest/TestForLocation.java#L40-L43

Test for MyMetadataExtractor

https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/JUnitTest/TestForMyMetaDataExtractor.java#L27-L68

Test for Song
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/JUnitTest/TestForSong.java#L46-L233

Test for Timer
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/JUnitTest/TestForTimer.java#L42-L56

Espresso Tests:
MainActivityTest.java
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/MainActivityTest.java#L32-L92

MainActivityTest2.java
https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/blob/master/app/src/androidTest/java/com/cse110/ucsd/flashbackmusicproject/MainActivityTest2.java#L43-L84

ZenHub

1. Insert a valid link to your Zenhub board covering all the required points from the rubric

- *Link:*
  https://app.zenhub.com/workspace/o/cse-110-winter-2018/cse-110-team-project-team-1/boards?repos=119338282

2. Insert a valid link to the burndown chart
   Link:
   https://app.zenhub.com/workspace/o/cse-110-winter-2018/cse-110-team-project-team-1/reports?report=burndown&milestoneId=3073339&selectedPipelines=5a6f572a2be4db6d1638849a&showPRs=false

   For the first iteration, we did not automatically close the issues with our commits, instead we manually closed all the issues on the last day. However, if you look at our commit history we have been working all the way through
   https://app.zenhub.com/workspace/o/cse-110-winter-2018/cse-110-team-project-team-1/reports?report=burndown&milestoneId=3073336&selectedPipelines=5a6f572a2be4db6d1638849a&showPRs=false
   See Here:
   https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/commits/master?after=ad82fb5fead13b98e31d12c005064f4357f9b8c3+104

GitHub

1. Insert a valid link to the contribution chart of all the contributors,
   Link: https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/graphs/contributors
   Note: Seung and Jung both contributed with 2 different github accounts. Also much of the work was performed with pair programming.

2. Checkout a new branch "milestone1_delivery" , push the final code to it and provide a link to it for final code review
   Link: https://github.com/CSE-110-Winter-2018/cse-110-team-project-team-1/tree/milestone1_delivery