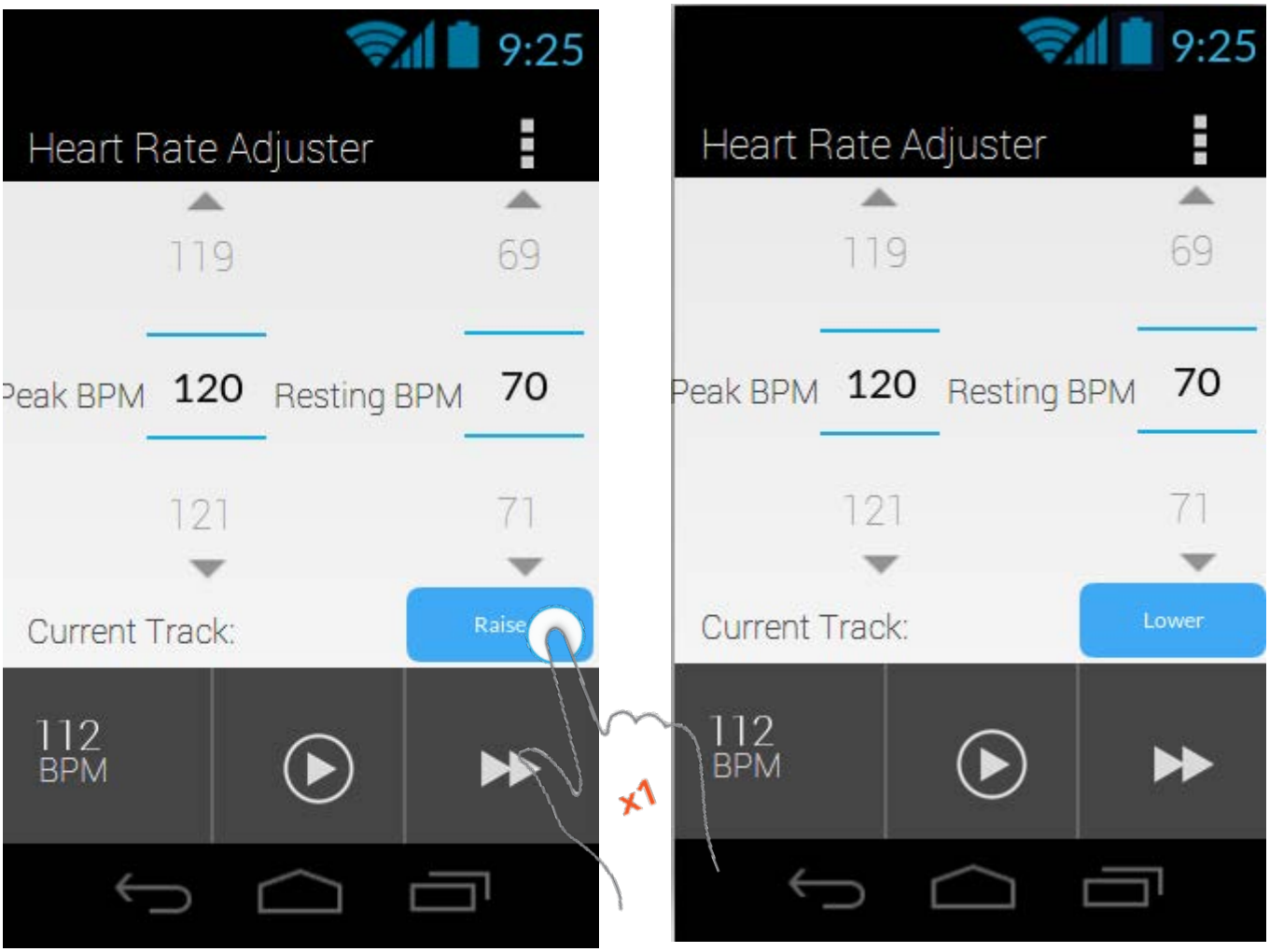
Preliminary Design

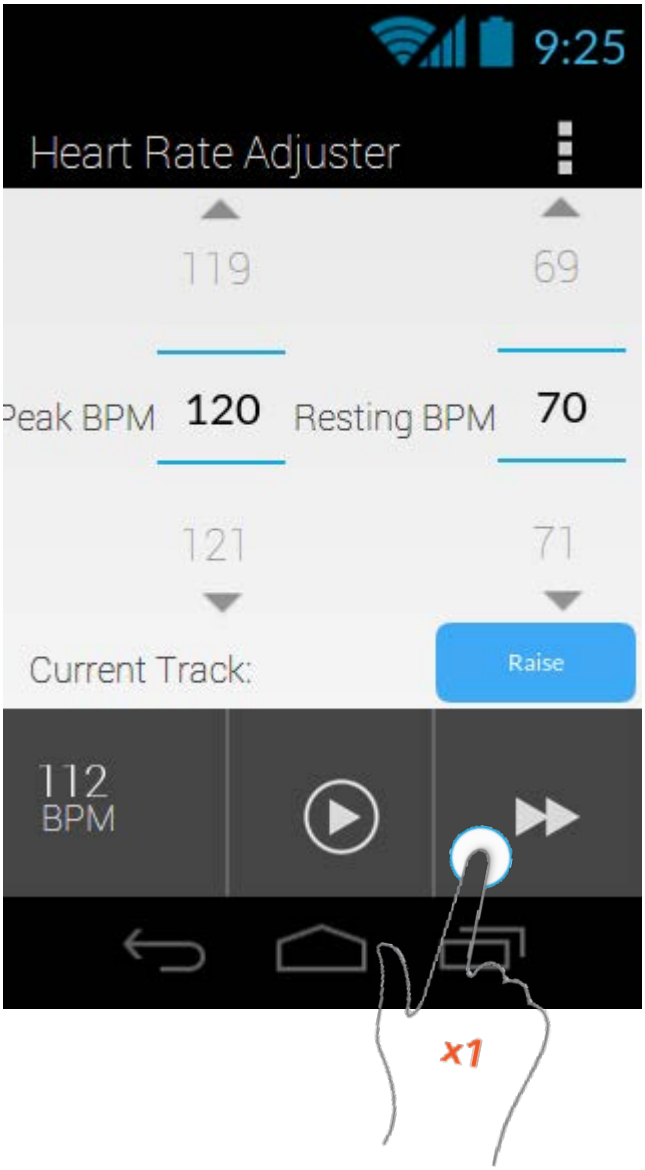
Use Case UC-2: decreaseHeartRate



For the first two use cases, the user's goal is to select a target heart rate for his workout. Because they are essentially two versions of the same task, we only include a single preliminary design of this use case. As seen from the screenshots of our "home" screen above, we seek to minimize user effort in accomplishing his desired goal. 1) To decrease heart rate, a user simply needs to press the "Raise" button to toggle the setting to "Lower." This means that he is now in the correct mode to lower his heart rate. 2) Next, the user must select his target BPM. There are two options: Peak BPM and Resting BPM. The user needs to swipe upwards or downwards to scroll and select his desired Peak BPM, for the workout, and then choose a more relaxed, Resting BPM.

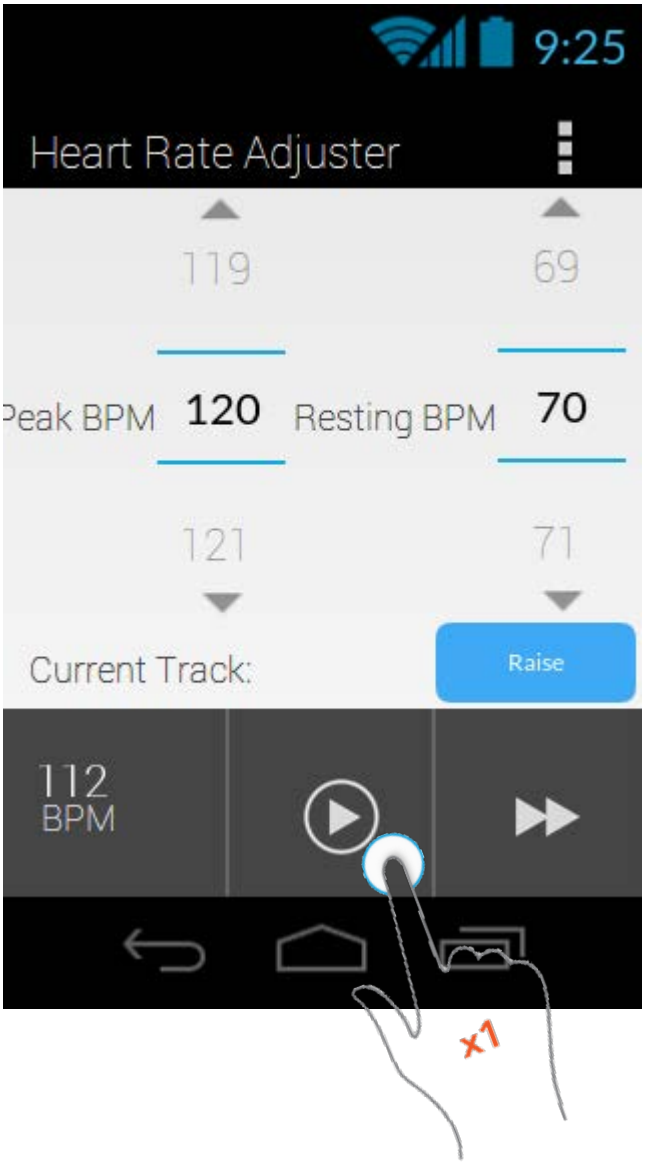
When finished, the system does the remainder of the work, helping the user to increase or decrease his heart rate, and then indicating the continuous effect in the bottom left corner, allowing the user to monitor his condition. This feedback is also furthered discussed in the getStatistics use case later on. (Again, fulfilling use case 1 and increasing heart rate is self-explanatory. The user merely needs to toggle the button back to "Raise" and select his BPM.)

Use Case UC-3: getNextSong



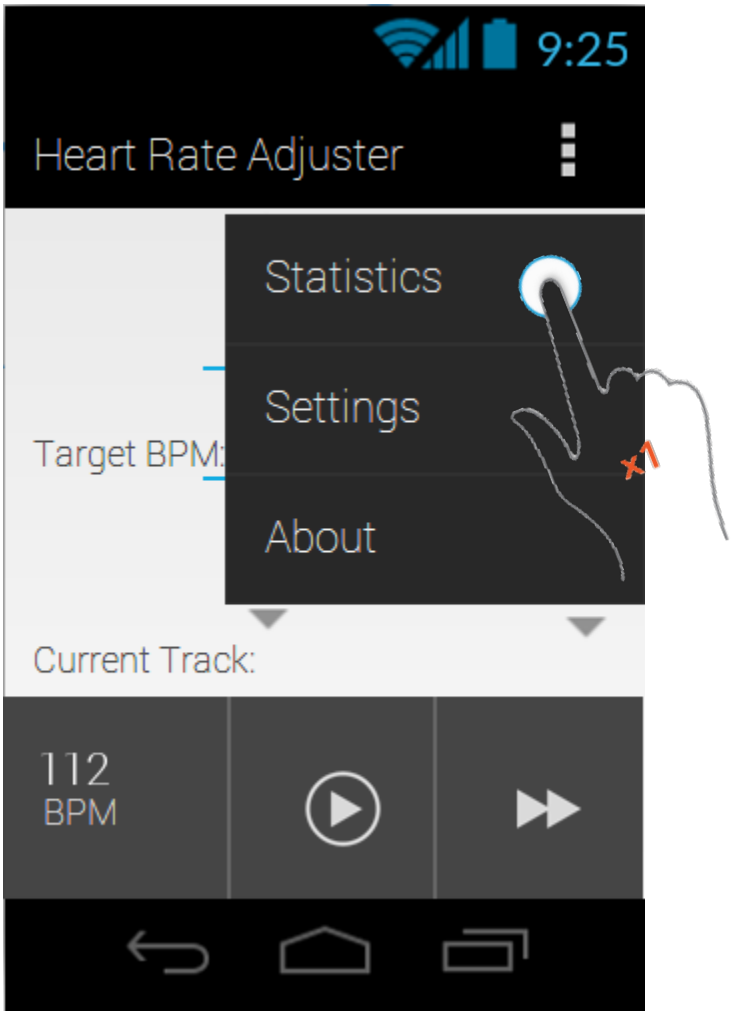
To switch tracks is also very simple. It takes the user one simple tap to achieve his desired outcome. On the provided image of our concept interface, our application appears very similar to a mainstream music player. In the bottom right corner is the double-arrowed fast forward button. The user taps this button to advance to another song, and then the system fulfills that request by running its algorithm and picking out another track from the user's music library. The "Current Track:" label will also be updated accordingly, and possibly even the cover art of the album.

Use Case UC-4: pausePlayback



Use Case 4, pausePlayback also proves to be extremely intuitive. Just like most music players, our application has a button located on the bottom center of the screen designed for the purpose of pausing the current song, or playing it, depending on the current state. The user just needs a single tap on the universal play/pause button to achieve his goal of pausing the song. When this is done, the system responds by stopping its collection of heart rate data, and freezing the screen in its current state. (This would be readily updated upon playback.)

Use Case UC-5: getStatistics



For use case 5, the user desires to view the statistics of his workout. To simplify the process for the user down to two clicks, we added a menu button in the top right corner of the screen. After pressing that menu button, a scroll-down menu with three options appears. The user needs to tap “Selections” to bring up his workout information. The system is constantly logging the user data, and compiles a few useful graphs such as heart rate versus time. Other different metrics may also be selected, allowing the user instant access to data that can help him improve his workout.

We omit use case 6 in our preliminary design because we feel that it is self-explanatory. The user may tap the BPM square in the lower left corner to update his BPM, but most likely, the BPM will be updated in real time. In that case, the user simply needs to look at the screen to view his current BPM.