For the front end, we ran into a few problems with connectivity to some of the functions of the back end. From the start, we found that using customized buttons presented themselves a challenge. However, using the ImageButton class of Android Studio allowed us to make our own buttons online, adapt the color scheme in Microsoft Paint, then upload them seamlessly into Android Studio. The ImageButtons had different IDs however, and needed to be wired differently in order to interact with the back end. The most difficult object was the progress bar. While creating its own .xml file in order to make it a drawable was simple, the actual motion of the bar was tough. We had to include the ActionBarDrawerToggle library from Android v4 to accomplish this. We also had to utilize the Thread.sleep() function to allow the progress bar to update. Finally, we had to initiate a Handler class to start a Runnable in order to get the progress bar to assist in this. We were going to import our own fonts, however, we came too close to the deadline. The seekbar used to switch the running units was simple to make; albeit the lack of better design style.

Some important component connections:

Start! Button: This button begins everything. Upon clicking it, the timer begins, the progress bar begins, the app starts tracking location, and the app as a whole starts. This is utilized with the onClick() class, and had its fair share of difficulties in coding. Also, a text phrase documenting how progress completion appears. The progress bar is updated at runtime when the user enters in their desired distance range. This was actually easier than expected.

Stop! Button: This button stops everything. Upon clicking it, the timer ends, and the progress bar is supposed to end. There were many problems with the stop button. First, the progress bar does not respond to the stop button; it merely refreshes slightly. This could be due to the difficulties in starting a Runnable within a loop instead of an entire class.

In addition to these two buttons, there are many buttons that connect the user to the database, leaderboards, and unit changing. These are self explanatory and work very well. Finally, error messages appear whenever the user types in an incorrect value. This was used with a Toast() key.

We had two main parts for our application: the application itself and the database.

For the application, we used a

* Main.java: This class initialized the Go, Stop and Select Units buttons. Depending on which button was pressed, they had different functionalities (this was done using a switch statement). It creates messages to pop up if there are errors, such as no distance being entered or the distance being too small. It also creates the timer when the Go button is pressed and it makes the Progress Bar.
* myLocationListener.java: This creates the location manager, which uses the GPS provider to find out where the user is. It also gets the longitude and the latitude, and sets up conditions depending on whether the location is changed, the provider is disabled, the provider is enabled and the status of the GPS provider is changed.
* Result.java: This class makes the display for the Select Units screen.
* SharedPreference.java: This saves all the preference from the Result class into a save file. This file is called every time the user opens the app.

For the database, we used

* config.java: The class is used to store values of variables to be sent to the database. For example, “public static final String URL\_GET\_RECORDS” stores the URL of the PHP file that is like an application within the server. The PHP file interacts with the database that is on the server. Another example of code found in this class, “public static final String KEY\_USR\_EMAIL” stores the way that the variable should be written so that it can modify the correct variable in the PHP file. The names required by the PHP file are stored in this class.
* registration.java: This class has the code that displays the leaderboard and personal records.
* RequestHandler.java: When any function wants to interface with the database, it has to go through this class. It is the middle man between the local application and the minuteman.
* signup.java: This class creates the screen to allow users to either create an account using their e-mail and name, or sign in.

One of the challenges we had was figuring out how to make the distance update automatically. We learned about the LocationListener class, which helped us with the requestUpdateLocations() method, but the distance still refused to update automatically.

Another challenge we had was having the progress bar stopping when stop was pressed. We tried to make a second OnClick, using while loops and making Runnable classes, but finally neither the location nor the progress bar were working the way we wanted them to be.