TJR Sensing Software Documentation

July 18, 2018

Adonay Resom

WPI Research Project

TJR sensing is an application that collects gait data from patients that had Total Joint Replacement surgery. This application is a modified version of an application named AlcoGaitDataGatherer. AlcoGaitDataGatherer is an android application that is used to collect gait data (walking behavior) from test subjects via sensors in smartphones and wearable Android devices. The TJR Sensing application was built by applying modifications to the walk type and walk number system from the AlcoGaitDataGatherer application. The wearable feature was also completely removed since it was not required for the scope of this project.

*The box.com upload system code from the AlcoGaitDataGatherer app was kept, but permanently disabled (meaning it can't be activated from user side). The code for this feature was kept because it might be needed in future updates.

Compiling

The application was developed using Android Studio 3.1.3. In addition to the Android Software Development Kit, several other libraries were used through Gradle dependencies (located in the app level build.gradle files).

The Android codebase compiles successfully with Gradle v3.1.3. This application is developed for devices that run the Android API level 27 (Oreo, v8.1.0), however it is also backward compatible with other API levels that are as low as level 23 (Marshmallow, v6.0).

Code

The Java language was used for developing Android code and it was structured into the following modules and packages:

- **app (phone module)**
  - Interfaces package
    - Contains the interface *BoxUploadProgressListener.java*, which was used to update progress to a custom view while uploading a CSV file from the phone to an online Box account.
  - Models package
    - Contains Java classes that possess the data types, structures and methods used to collect and store data during tests/surveys from test subjects.
    - *Gender.java* contains predefined constants and methods that are used to create the enumeration (enum) data type called Gender.
    - *WalkType.java* contains predefined constants and methods that are used to create the enumeration (enum) data type called WalkType. It contains 5 items. Each item

represents a type of walk that the test subject will do during the survey. Along with the type of walk, other data such as the different activity instructions and activity durations limits (in seconds) is stored in this enumeration.

- *SensorRecorder.java* is a class that handles tasks related to recording, storing, and organizing sensor data collected from the smartphone.
- *TestSubject.java* is a class that defines the attributes needed to create objects that represent a single test subject.
- *Walk.java* is a class that defines the attributes needed to create objects that organize and store sensor data from a single walk. This class is only used for sensor data collected from the phone's sensors.
- *WalkHolder.java* is a class that defines the attributes needed to create objects that can store several type of walks (Walk class objects that have different WalkType (enum) attributes). The data is stored in HashMap structure that uses WalkType as a key and a Walk object as a value.
  - This implementation was created in the previous AlcoGatDataGatherer application because it helped decrease the amount of RAM used while recording several walk numbers from a test subject. However, even though the walk number system was removed for this new application, the WalkHolder class was kept in order to make the development of the new app easier because various other components of the app are dependent on the existence of the WalkHolder object.

- Tasks package
  - Contains Java classes that extend the *AsyncTask* class. They handle tasks that require to be processed in separate/individual threads while also accessing the UI thread in order update elements that inform the user about the task's progress.
  - *SaveWalkHolderoCSVTask.java* is a class that extends the *AsyncTask* class in order to create a new thread and save the contents of a WalkHolder object into a CSV file located in the phone's internal storage. While writing the data, it updates a progress bar in the UI thread.
  - *UploadToBoxTask.java* is a class that extends the *AsyncTask* class in order to upload a CSV file to an online Box account while also updating a custom view inside the application's home activity with the progress of the uploading process.

- UI package
  - Activities package
    - *SplashScreenActivity.java* is a class that generates and controls the splash screen that shows up each time the application is started.
    - *HomeActivity.java* is a class that generates and controls the UI elements contained in the applications home screen. This class mainly generates the list of tests/surveys that have been previously completed. The user can also select one of the previous surveys and choose to resume gathering data from the test subject or delete the entire history of that test subject.

* If box.com feature is enabled: It also displays the upload progress of each file if the upload/synchronize button is pressed. The synchronize button is only available in this activity if the user enables the 'Enable Box Integration' option in settings (*SettingsActivity*). It also provides access towards the *SettingsActivity* and *SurveyFormActivity.*

- *SurveyFormActivity.java* is a class that generates and controls the form in which the user enters the test subject's subject ID and demographic information. The class restricts the user form entering invalid data before proceeding onto the next screen. The data limitations are displayed on the screen. This activity provides access towards the *DataGatheringActivity*.
- *DataGatheringActivity.java* is a class that generates and controls the activity in which the user enters BAC data and records gait data. The activity enables the user to start and stop a walk at will. It also enables the user to re-do previous walks before saving them. The activity displays different flows based on the user's preference to use a wearable device. If the user prefers to use a wearable device while recording gait data, additional UI elements are added on to the screen in order to act synchronously with the application in the wearable device. This activity is also in charge of adding and removing listeners that enable the application to read data from sensors and communicate with wearable devices. It also provides access towards the *WalkReportFragment* and *HomeActivity.*
- *SettingsActivity.java* is a class that generates an activity that displays two options. One option enables the application to collect data from wearable devices, while the other one enables it to upload CSV files to an online Box account.

o Fragments package
  - Contains a class called *WalkReportFragment.java.* The fragment generates a panel in front of *DataGatheringActivity*. In this panel a list of check boxes and a text box are generated. The user can use these elements to submit a report along with the collected data. The check boxes in specific can be used to invalidate walks that might have not collected accurate data. Whoever receives the collected data can then read the report and understand the walks that have been negatively affected by events.

o Adapters package
  - *MyWalkReportRecyclerView.java* is a class that is used for generating and getting content from the *RecyclerView* used in *WalkReportFragment*. The *RecyclerView* contains information and a check box for each walk and a single text box for writing a report message. As a result, the adapter implements the *CompoundButton.OnCheckedChangeListener* interface in order to keep track the check boxes that have been selected by the user. When the submit button inside the fragment is pressed, this adapter class receives the user's input and sends it to the *DataGatheringActivity* in order to save it along with the gait data collected throughout the survey.

- *SurveyListAdapter.java* is a class that is used to generate the content displayed on the *HomeActivity*'s *ListView* element. It basically scans the internal storage of the phone and finds previously completed surveys in order to display them in the *HomeActivty* in a chronological order.

- Views package
  - Contains a class called *CustomSurveyView.java.* The class extends th*e RelativeLayout* and implements the *BoxUploadProgressListener* interface. The class is used as the main UI element inside each the *ListView* item that is displayed in the application's *HomeActivity.* The class was created because it was difficult to update each *ListView* element with an upload progresses of the file displayed using normal Android *View* classes. That is why the class implements the *BoxUploadProgressListener* interface. The interface makes it easier to update UI elements contained in the class (view) from outside of the class (for example from the *UploadToBoxTask* class).

Interface Flow

Phone Interface Flow Chart

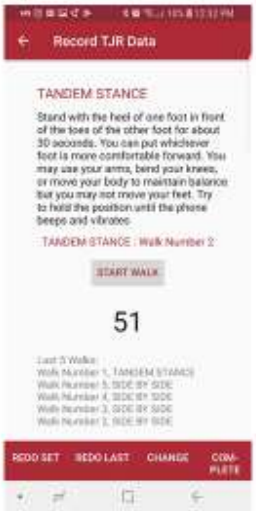
The following flowchart show's how the application's interface behaves (on Samsung Galaxy S9, Android 8.0):

- Enter subject info in form
- Form contains error check
- Once info is filled correctly hit submit

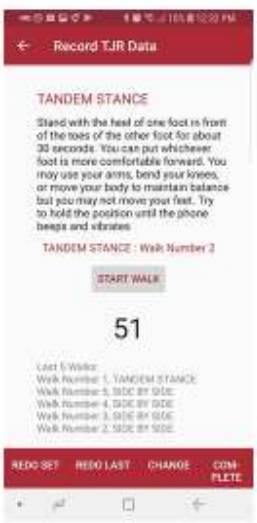
Choose test/walk type you would like to gather data for (accelerometer, gyroscope and compass)

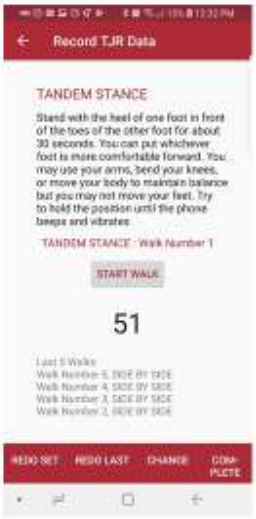
Records walk for the chosen walk type

Change walk type (by pressing the 'CHANGE' button)



- ‘REDO LAST’ removes last walk data
- In this case it removes Walk Number 1 for ‘Chair Rise Test’





- ‘REDO SET’ restarts data collection for the current walk type (deletes all walk numbers within that walk type data set)

- After collecting all data press ‘COMPLETE’ to finish and exit data collection process for this test subject
- There will be an option to submit a report for the sessions before exiting

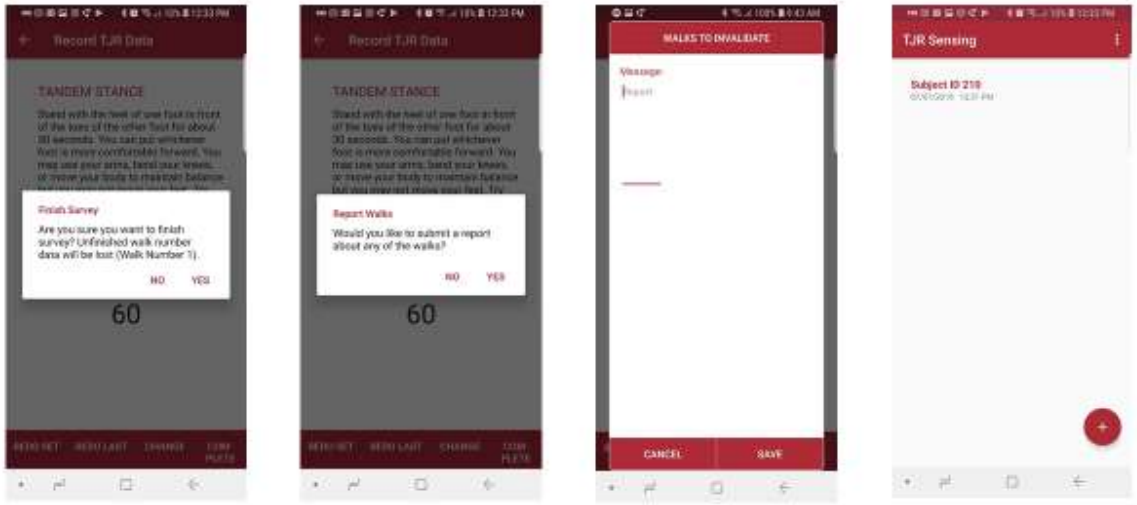


Figure 1.0 Phone Interface Flow