Android Development Document

***Purpose****: To outline the main problems encountered during android development of the Chappy application and their associated workarounds.*

* Problem 1:

Q: How do we connect to the Service methods that the middle tier created?

A: To accomplish this, a class titled “Connection” was created. This class extends the android API’s

“AsyncTask” class. Utilizing this class had two main benefits:

1. To establish a connection to the server that won’t take too long for the user.
2. To allow updating of the UI without worrying about thread access that could quite possibly crash the entire application. For example: (Trying to access the main application thread after you’ve manually created a new thread to connect to the server)

Inside the “Connection” class, we had to create an instance of Java’s “HttpUrlConnection” class to connect to the server. After that, it was a matter of setting the connection to use the Http web “Post” method, and telling it to use JSON as the data transfer type. Finally, to connect to specific service methods, the “Connection” class’s execute method (inherited from “AysncTask”) took in two strings:

1. String one: The url of the Chappy\_Host.svc location along with the name of the method.
2. String two: This string lets the “Connection class know which connection method to call when the execute() method is invoked. (e.g.: “GetBuildings”)

Inside of each of the “Connection” class’ connection methods, the connection is either wrote to using Java’s “OutputStreamWriter” class, read from using java’s “BufferedReader” class, or both.

* Problem 2:

Q: How do we translate the raw JSON returned from the connection or write JSON to the connection?

A: Reading JSON: Read the data into an instance of “StringBuilder”. Next, if the connection method only returns one JSON object, use android’s “JSONOBJECT” class with the StringBuilder variable in the constructor to retrieve each field from the string. (e.g.: obj.getInt(“ClientID”))

If the connection returns a collection of json objects, do the same, but split the string into an array of strings at every closing curly brace (“}”). Next, remove all of the slashes, brackets, and quotation marks. Lastly, append each closing curly brace back on into each string index before attempting to read each JSON object’s data.

Writing JSON: If the service method expects a primitive data type, the JSON format must look something like “{“Value”:1}”. On the other hand, if the parameter wants a class object, the JSON format must include each of the class’s properties represented as individual JSON objects. (e.g. :{“ClientID”:1}{“Name”:”Ricky”}etc..)

* Problem 3:

Q: How do we share resources between the different views of the application?

A: To do this, a Singleton class was created titled, “ActiveSession”. This class stores data such as the active Client class, active Chaperone Class, and connection parameter strings.

* Problem 4:

Q: How do we get the GPS location of a client or chaperone?

A: Google’s API was used for this problem. It was found that Google’s GPS functionality is slightly more accurate, and updates locations more frequently than android’s native GPS methods.