Design Overview

Our product can be divided up into three components: the API, the mobile web client and the Android smartphone application. The API interacts with the back end MySQL database (specifically the `persons` and `encounters` tables), and the web and android clients interact with the API. This format allows for easy extensibility and abstracts direct database interaction away from any user-visible code. We created two APIs, one for creating connections (connectapi.php) and the other for getting a list of names and IDs for a find-as-you-type style search scenario (getNameList.php).

Both the mobile web application and the Android application allow connection creation, including a user-defined comment, and both allow the user to enter IDs by searching for users by name rather than typing an eight-digit number (the current attendee identification scenario) manually. These functions use connectapi.php and getNamelist.php, respectively. The application also saves a user's own ID on device so it does not need to be re-entered on every connection, and optionally includes Bump, which allows two smartphone client users to gat each other's IDs by physically bumping their phones together.

For our mobile web application, we designed two web pages, one with find-as-you-type search functionality and one without. We used JavaScript to resize the CONNECT logo banner to fit device screens, depending on the device's screen resolution (desktop or mobile) and orientation (portrait or landscape). Because low-end mobile browsers do not support JavaScript, we made sure that basic functionality could still be accomplished with JavaScript disabled.

Figure 2 is a revised architecture diagram detailing the changes that were made by our team:

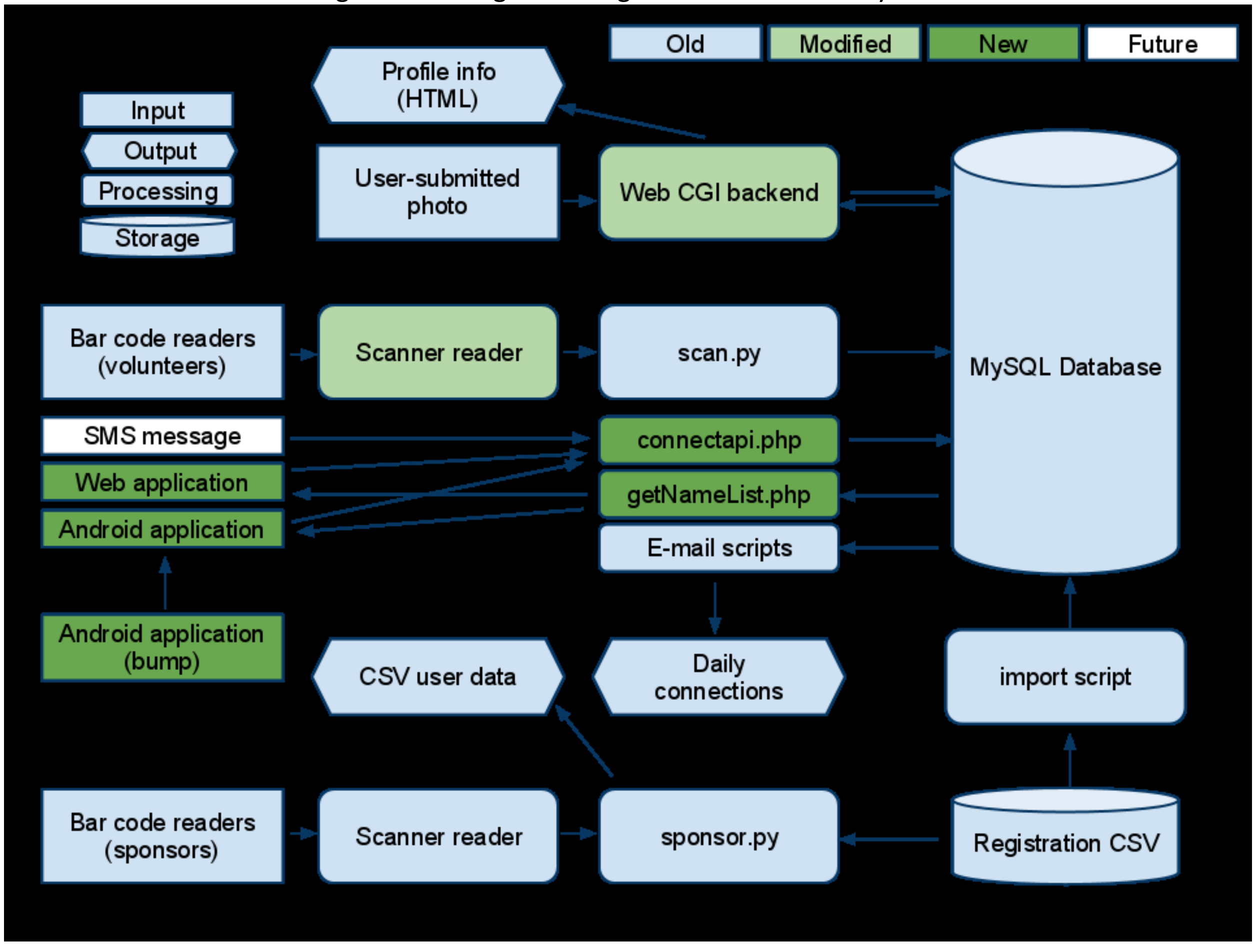
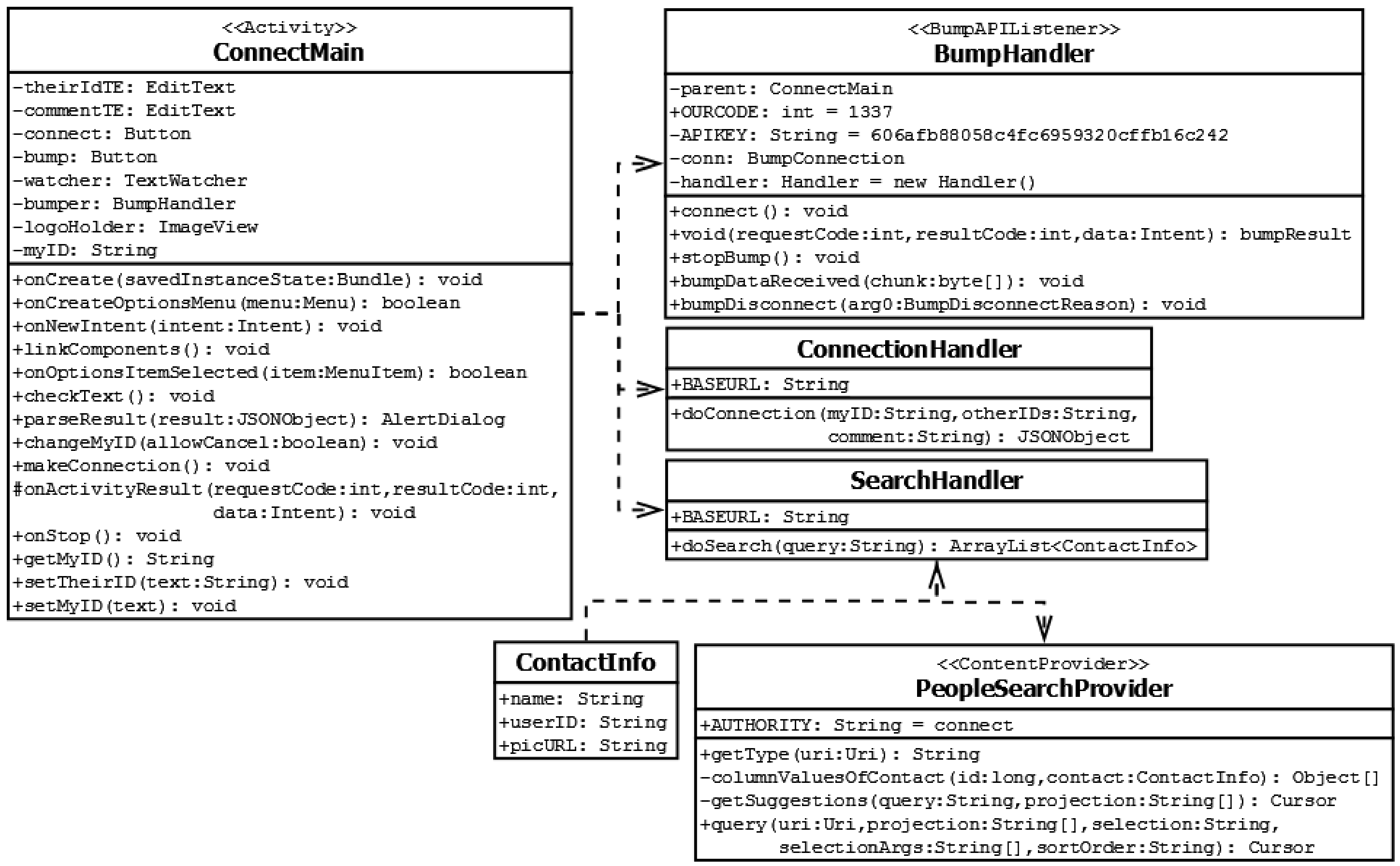


Figure 2 – Post field session architecture/data flow diagram

Android Details

One snag our team ran into was that the Bump API for Android currently crashes when called if an XML file is in the Android res/xml directory for a program. The problem with this in our case is that Android's search API requires programs using it to have an XML file in that specific folder. The problem took awhile to track down, however as it turned out we were not the first ones to experience this difficulty. See http://groups.google.com/group/bump-api-android/browse_thread/thread/3caa1f58019b73eb for more details on this issue.

Our Android application ended up being divided between five Java classes in all. Our main (ConnectAndroid.java) file contained the majority of program code, however both search and connection API calls were given their own classes: `ConnectionHandler.java` and `SearchHandler.java`, respectively. Android requires a ContentProvider implementation to deliver properly formatted suggestions to its search facility for find-as-you-type; this was provided in `PeopleSearchProvider.java`. Our customized Bump API listener implementation can be found in `BumpHandler.java` in the Bump branch of the Android project code.



<<Activity>>
ConnectMain
-theirIdTE: EditText
-commentTE: EditText
-connect: Button
-bump: Button
-watcher: TextWatcher
-bumper: BumpHandler
-logoHolder: ImageView
-myID: String
+onCreate(savedInstanceState:Bundle): void
+onCreateOptionsMenu(menu:Menu): boolean
+onNewIntent(intent:Intent): void
+linkComponents(): void
+onOptionsItemSelected(item:MenuItem): boolean
+checkText(): void
+parseResult(result:JSONObject): AlertDialog
+changeMyID(allowCancel:boolean): void
+makeConnection(): void
#onActivityResult(requestCode:int,resultCode:int, data:Intent): void
+onStop(): void
+getMyID(): String
+setTheirID(text:String): void
+setMyID(text): void
<<BumpAPIListener>>
BumpHandler
-parent: ConnectMain
+OURCODE: int = 1337
-APIKEY: String = 606afb88058c4fc6959320cffb16c242
-conn: BumpConnection
-handler: Handler = new Handler()
+connect(): void
+void(requestCode:int,resultCode:int,data:Intent): bumpResult
+stopBump(): void
+bumpDataReceived(chunk:byte[]): void
+bumpDisconnect(arg0:BumpDisconnectReason): void
ConnectionHandler
+BASEURL: String
+doConnection(myID:String,otherIDs:String, comment:String): JSONObject
SearchHandler
+BASEURL: String
+doSearch(query:String): ArrayList<ContactInfo>
ContactInfo
+name: String
+userID: String
+picURL: String
<<ContentProvider>>
PeopleSearchProvider
+AUTHORITY: String = connect
+getType(uri:Uri): String
-columnValuesOfContact(id:long,contact:ContactInfo): Object[]
-getSuggestions(query:String,projection:String[]): Cursor
+query(uri:Uri,projection:String[],selection:String, selectionArgs:String[],sortOrder:String): Cursor

Figure 3 - UML of Android application