**Design Description**

There is three main components to our project : the client, the server and a database. This document will describe each of their role.

**Client**

This side handles everything that has to do with GUI, user interaction and game processing.

We use the Android Studio library which is mainly java and some xml pages for the display. The GUI is mainly handled by xml files and some assets we have been gathering that will be drawn onto the layouts. As for the communication, we use two ways two communicate with the server. If the thread is allowed to continue computation while the client processes the request, we use Volley as it offers great way to communicate with http servers. In the case we need the thread to block while the request is processed ( for example to get data from the server in order to draw it ), then we use AsyncTask, that allow to force a thread to wait before proceeding and protects from deadlocks since the request will always be replied by server. Moreover we use the Gson library to process data from and to the server very easily and with more efficiency.

As for the game logic, most of the computation is computed on the client side. The client periodically polls the server for an update on the server, and sends his Gamestate periodically to the server so that other clients can use it. This means that our game is partially independent of the server, we have decided this design way for two main reason : limit communication between the server and the client, allow for client more reliability in case he has a bad network. There will be a timeout period for the client so he can’t start playing totally offline.

**Server**

The server is coded all in php. We put the php files on the server and opened up port 80 permanently so that any client can at any time make a php request. There is a subfolder of php files which are copy of the class used for communication with the client. The reason to have them is again to make for very efficient and easy JSON encoding. This way, to make a JSON object response we just need to make a “copy” of the Class from java to php and json\_encode / json\_decode make most of the work for us.

The server also handles all of our connections to the database. It uses the built in methods to access the mysql database we have set up. This way we can let the server take care of all the database access and limiting access from outside to it.

The main goal of the server is hence to be an intermediary between either client-database, or client-client.

**Database**

The database holds a few tables :

* Player : a table holding all of the players and their statistics.
* Friends : a table holding a list of the friends for each player.
* Maps : a table holding metadata for all the maps ( description, author etc..)
* MapData : a table holding the actual data for all the maps.