# Sprint 3 Feedback

## Client

* There seems to be a huge lack of design on the client side. It looks like it is being built with little design forethought being put into it.
* Resources in ClientApi are stored in vectors of keys and values ???? Is there some reason you are not using maps or hashtables? On the surface the use of vectors looks to be hugely inefficient.
* Event handlers are registered as function pointers. This is a very non-object-oriented way of doing event handlers. Normally, you would create an class like MouseEventHandler with an abstract method like handleButtonPress(Event e) that would be overridden by concrete subclasses. The way things are, it looks like you are a refuge from C writing your first C++ program.
* Despite minor concerns and areas for improvement, implementation of GUI looks good. Having some design would have allowed you to see areas where you could get additional efficiency.

## Database

* The class diagram to save a game seems strange. I suspect it should be an activity diagram, not a class diagram. As a class diagram, it provides less info than in the rough class diagram for the data structures.
* The database API in the implementation section has a lot of get methods which return void and have const parameters. Not having a way to return a value is likely a problem here. Saving a game should take a pointer to a game to save, should it not?

## Networking

* Why are Serializable and Deserializable two separate classes? I would have made them one class with methods for serialization and deserialization.
* Primitive serialization is done via global functions. Why not make them static members of Serializable and reduce pollution of the namespace?
* I am happy to see testing of the serialization methods. It would be good if you used a test harness like cpptest. This would allow tests for all of the different parts of the software to be combined and run at once easily.
* The testing of packets and connections is also welcome but should be moved into a harness.

## Server

* I like the design work on the server command protocol. What I am not seeing (which does not mean it might not be there) is how an executed command which results in a change that all of the clients need to be notified about is able to notify the clients of the change.
* In the client class diagrams it looks like the aggregation and composition diamonds are on the wrong end of the association.
* The design of the server is starting to come together and look like something which could be implemented. A single template class could be built to implement resource pools of all types and this would save time.
* I can find neither design nor implementation of a class for a database connection. Tell me where to find it if you want marks for it.
* Cannot find either design or implementation of game lobby and associated classes. Tell me where they are to get marks.