**SOEN 423: Distributed Systems**

**Assignment 1 Design Documentation**

**Jordan Hubscher**

**Student ID: 27019696**

The overall architecture was quite simple. It was an implementation of a Client-Server model architecture with a communication model by use of the Remote Method Invocation (i.e. RMI) protocol.

For this assignment, I had set up three separate project directories using Gradle as my build, configuration, and dependency management tool.

The first project was a commonly shared interface definitions library (i.e. named “RMI-lib”.) that would be built and archived as a jar and included as a resource in both the Client and Server projects. Here I defined the the CenterServer Interface which extended the Remote interface to allow for remote object stub generation on runtime and remote invocation of its respective methods from it’s registered implementing classes.

The second project was the Server implementation (i.e. named “CenterServer”.). In this project, the CenterServer interface was implemented through the CenterServerImpl class, which also extended the UnicastRemoteObject class. By extending the UnicastRemoteObject class, CenterServerImpl can call the super() or super(int port) constructors from it’s own which would call the UnicastRemoteObject classes constructor which exports itself onto the server’s object registry located on the given port (port 1099 if no port was provided). A storage class called Employee was created so that the data for this assignment may be organized logically. Since no database was mentioned to be used for this project, a HashMap<long, Employee> object was used to store all of the said employees, indexed by their respective employee IDs. Similarly, another HashMap<long, Project> was used to store the project information, using the corresponding Project storage class to do exactly that.

The third and last project was the Client implementation (i.e. named “ManagerClient”.). In this project, the client fetches the object registry at the provided host and on the provided port. Using the established connection through the registry object, the client can stream the exported implemented objects from the server via a stub (which contains only the methods from its defining remote interface.). Casting this stub to the provided object type name’s remote interface type, the client may now use the object as a normal Java object, however it’s methods would be invoked remotely.