**SOEN 423: Distributed Systems**

**Assignment 2 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 CORBA-IIOP protocol.

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

For each project, I included the CenterServer.idl file that defined the corresponding methods for the implementing Servant object that would later be registered to the Naming Context Registry and have it’s method invocations through the set Object Request Broker.

The first project was the Server implementation (i.e. named “CORBA-CenterServer”.). In this project, the **CenterServerImpl** class extended the CenterServerPOA class generated from the IDL. An abstract UserRecord class was created connecting all the common functionality between Employees and Managers. From UserRecord, an EmployeeRecord and ManagerRecord class were derived. Since no database was mentioned to be used for this project, a **Map<Character, List<UserRecord>>** object was used to store all of the said employees, indexed by their the first letter of their last names. Similarly, another **Map<String, ProjectRecord>** was used to store the project information, using the corresponding **ProjectRecord** storage class to do exactly that.

The second and last project was the Client implementation (i.e. named “CORBA-ManagerClient”.). In this project, the client fetches the root naming context registry. The client then creates a CenterServer object for each server location and sets it to the same Object Request Broker object. The client then resolves each remote object from their pre-determined registered names. From this point on, the remote objects are available for invocation.