The architecture of this project centers around utilizing CORBA to facilitate a distributed order management system. The server and client interact remotely via IDL-defined interfaces, providing clear separation and abstraction of services. The IDL file defines a single module, OrderManagement, which exposes procedures necessary to manage orders and enforce user roles, namely Customers and Managers. The IDL definitions include placeOrder, checkOrderStatus, and an Order struct to standardize order details along with a couple additional procedures I added later: managerLogin, and managerDisconnect.

The server implementation extends the generated OrderServicePOA class and manages the logic to store, retrieve, and update order information. I implemented this using a Hashtable to store Order objects keyed by usernames. As stated, this program, for simplicity, assumes that each username can only have one username associated with it. This is why, I decided that if a username already in the system is provided again, it will overwrite the existing order information. This functionally allows users to modify their orders.

The restriction to a single manager at any given time, is addressed by employing a boolean flag indicating the active status of the manager. Manager clients authenticate through the managerLogin method, which checks and toggles this flag accordingly. I added this after compiling the initial IDL file and it was one of the last features I implemented. As such, I ran into some issues with re-compilation and making sure all the class files were up-to-date – briefly getting a BAD_OPERATION error that took me a while to debug.

Both user types are implemented in a single client file, and are differentiated based on user role selection for simplicity. This uses the generated stub classes to communicate with the remote server via the ORB, providing role-specific interactions without maintaining multiple codebases. To ensure robustness, I implemented some basic exception handling for invalid inputs and prompting users for retry.

Throughout development, notable challenges included ensuring synchronized access to shared resources like the Hashtable for orders and managing the state of the manager session. Additionally, I had some difficulties arise from managing CORBA-related runtime exceptions, particularly concerning client-server communication stability. These issues were managed through connection logic and some exception management within both client and server applications.

Overall, the project's design successfully demonstrates a robust implementation of a distributed system leveraging CORBA technology to effectively manage stateful interactions and enforce role-based restrictions, maintaining data consistency and ensuring reliability across client and server communications. This helped familiarize me with CORBA as a smoother alternative to RPC which I have used in another class before this. I enjoyed working on this project more than the previous one as I was able to easily implement this on my mac rather than rely on a Linux machine with limited resources.