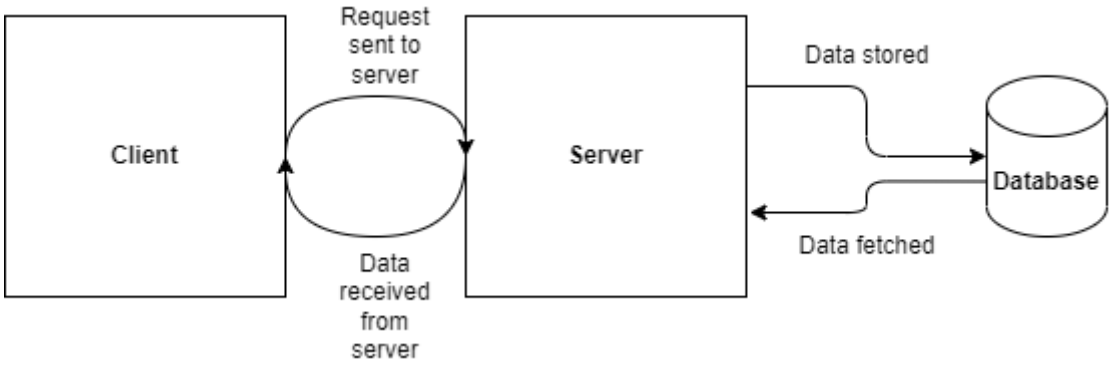
CPRE-550 Project 2

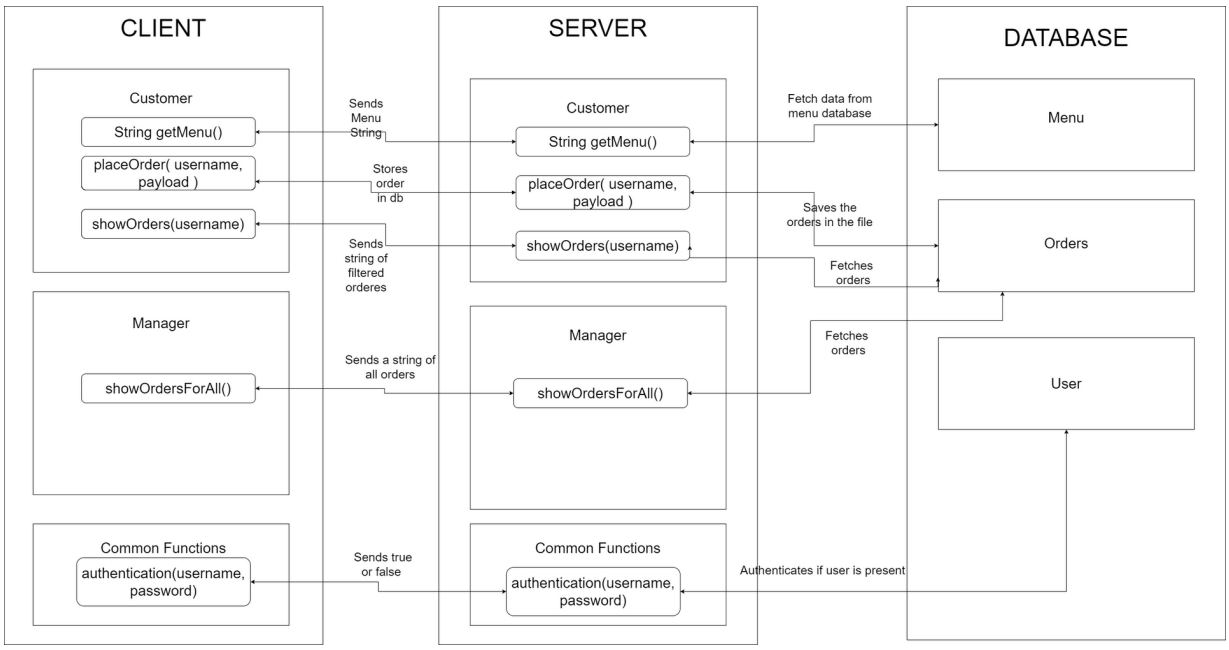
Project 2 (Common Object Request Broker Architecture)

Peeyush Gupta - peeyush@iastate.edu

- **System Design**
  - **Objective:** The objective of the project is to create an Application which can serve to both Customers and Manager.
  - **Proposed Solution:** The project is based on Common Object Request Broker standard. Different functions can be implemented on the server. These functions then can be used by Client to do the required transactions.
    - **System Diagram and its Components:** The System has primarily three components *Server*, *Client* and *Database*.

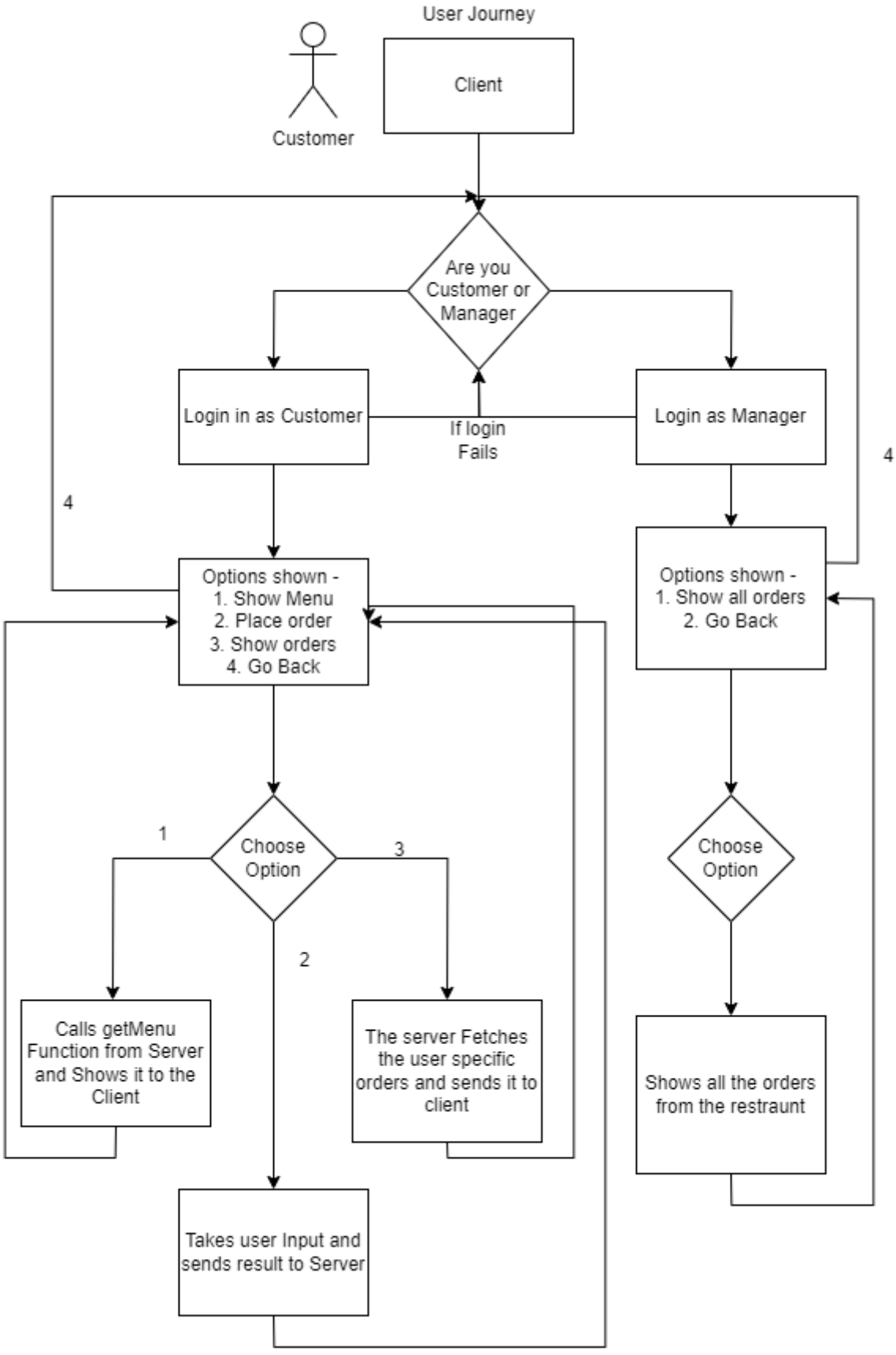


The client will act as a user interface, which will ask questions from clients. Based on the input of the client, appropriate result will be show. The server will process the requests of clients. Some requests may ask for data stored in database. The data is processed according to the need of user and sent back to the client
The database stores the values sent by client to the servers like orders and menu etc. This helps us retain the value even if the server craches or turns off unexpectedly.



- **System Functions:** All the functions can be categorised into three categores based on the which type of user they serve namley Customer, Manager and common Functions.
  - Authentication Function: This function validates weather the user has entered correct username and password. The password is Case Type checked. This user data is stored in Users Files. If the client has provided correct values, then it returns true boolean value else it returns false boolean value.
  - Get Menu Function: This function fetches the complete menu of the restruant. The menu is stored in Menu file, in key-value pairs, where key is the name of food item and its value is the cost of the item. It returns the whole menu in form of a string, which is displayed by client.
  - Place Order Function: This function is used by customers to place the order. It requires username, item key values and the quantity of each item. The user is not prompted to put his username because a variable is used to store the value during the authentication. It sends a message back showing if the order was successfully placed if the order is successfully stored in the orders file. If some error occurs in the process, it shows the appropraite error message.
  - Show orders : This function is also used by customers to see their orders. It takes username from client. The server fetches all the orders from Orders file and filters the data on basis of the username. Then the filtered data is sent to client in form of string.
  - Show all Orders: This function simply takes all the orders stored in orders file and sends in form of a string back to the client.

- **The Customer Journey**: The figure below shows what a Client journey looks like throughout the process. What prompts and options are visible to the user.



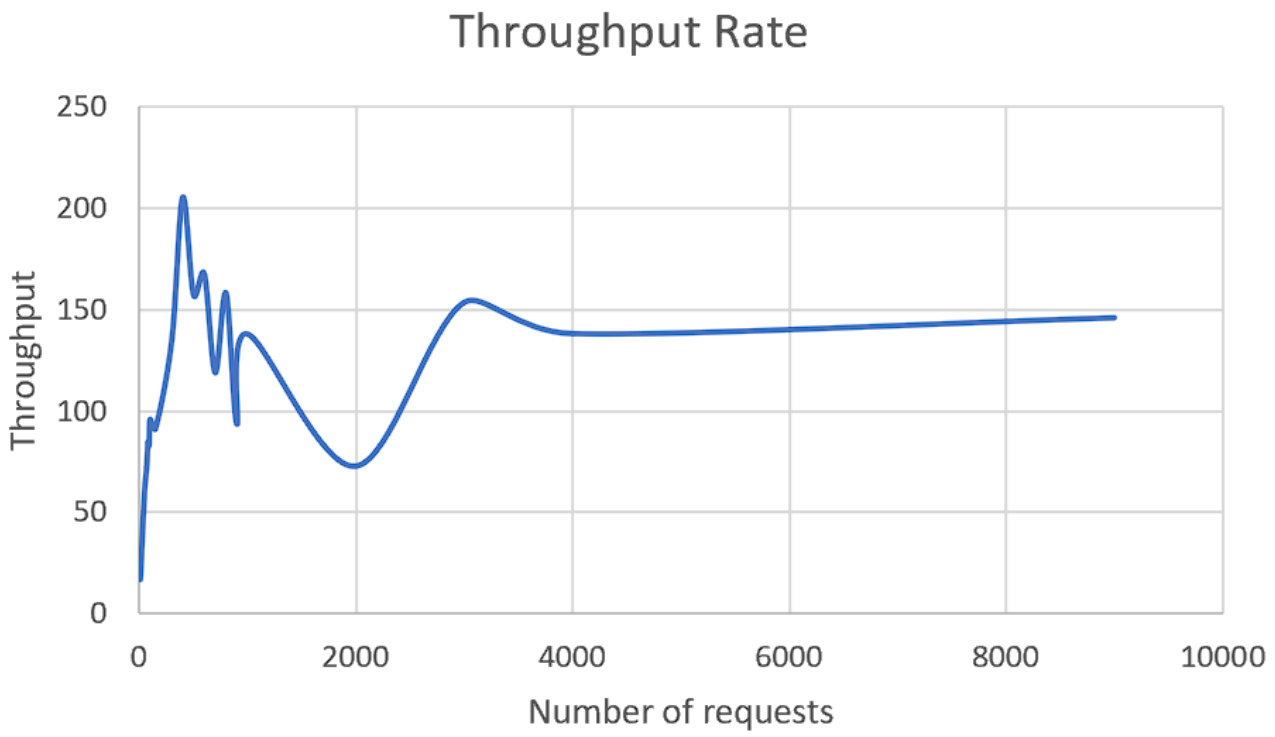
- **Perfomance**
  - **Response Time:** Response time refers to the time taken by system to respond to clients request. To measure response time, we can calculate the time difference between when a function from server is invoked and when server responds to the client.

Method	Time Taken (in seconds)

Method	Time Taken (in seconds)
Authentication for customer	0.008
Authentication for Manager	0.002
Placing order	0.008
Get Order status customer	0.003
Get Order status manager	0.003

- **Throughout Rate:** It is the number of requests a server can handle in the given time frame. To measure the throughput, I am calling multiple requests from a client to check for the throughput rate.

Requests	Time Taken (in seconds)	Throughput
10	0.601	16.63
20	0.714	28.14
50	0.81	61.74
100	1.045	95.69
400	1.934	206.82
600	3.2	187.5
900	3.544	253.95

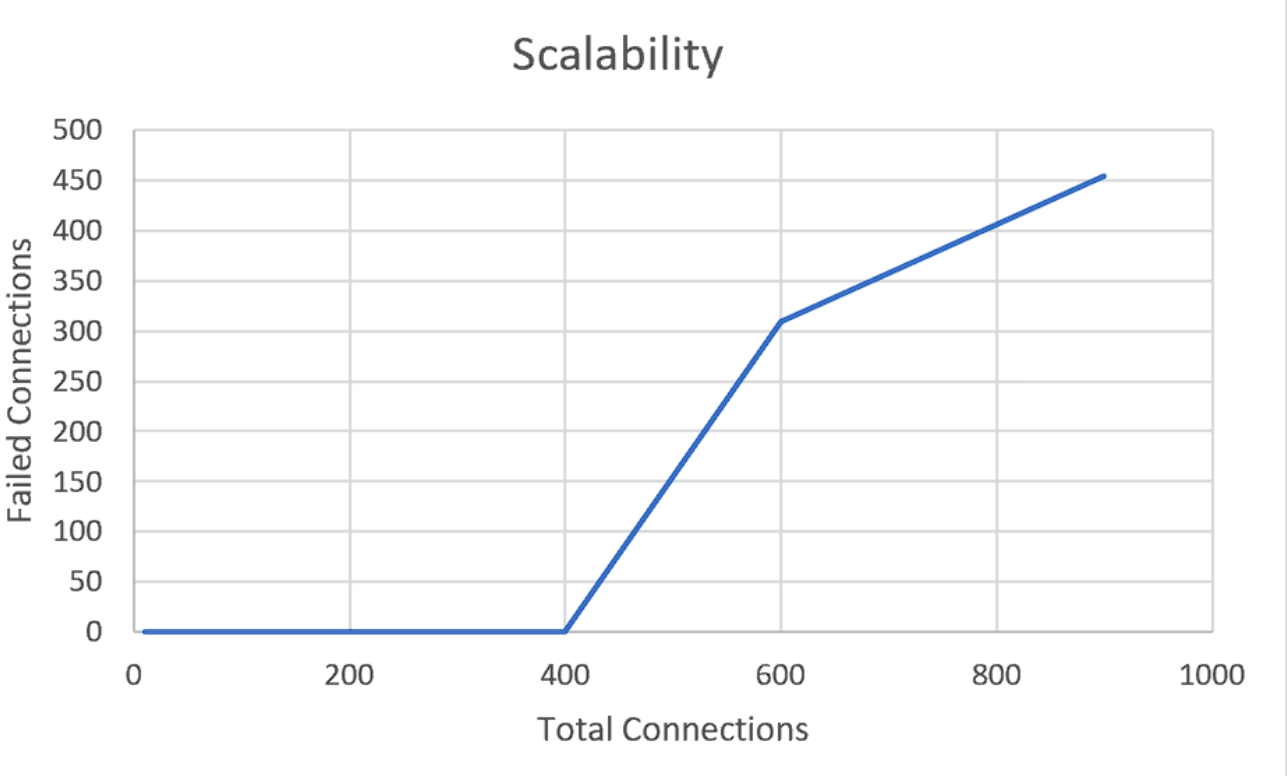


From the graph above it can be seen that throughput averages out to 150 requests per second.

- **Scalability:** It refers to the ability of the system to handle increasing number of clients and requests without any effect in performance. Scalability is calculated using multi-threading in

java. I am running various threads and each thread creates a new Client connection and does one function call to the server. Experimental results are documented below in table.

Clients	Time Taken (in seconds)	Failed Connections
10	0.571	0
20	0.765	0
50	0.867	0
100	1.112	0
400	2.237	0
600	3.193	310
900	3.742	454



From above graph it can be seen that connections attemp fail in between 400 to 600 connections. It can be deduced that server can handle upto 400 clients before any issues or errors start to occur.

- **Error Occured:** It refers to how many errors or requests were failed in a particular amount of time. For calculation of errors, I am catching all the errors and Exception thrown by server and counting them. The Experimental results are documented below.

Requests	Time Taken (in seconds)	Errors
0	0.601	0
0	0.714	0
0	0.81	0
00	1.045	0

Requests	Time Taken (in seconds)	Errors
00	1.934	0
00	3.2	64
00	3.544	569