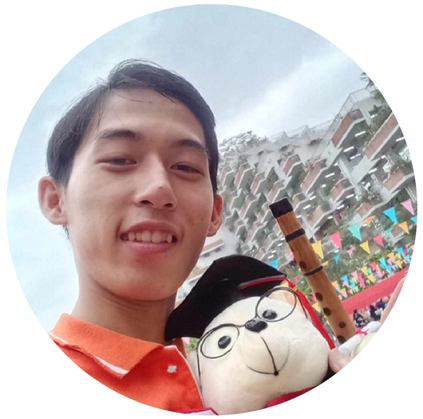
JULY 22, 2023

TABLE RESERVATION SYSTEM

A Table Reservation Software by Group 6
SWD392 - Software Architecture and Design

OUR TEAMATES

The team developer table reservation system is a web application that allows users to reserve tables at a restaurant or other dining establishments. It facilitates easy booking, manages reservations, and provides real-time availability status. The system enhances the overall dining experience for customers and helps restaurants streamline their table management processes.



ĐẶNG LÊ MINH QUANG
SE161711

TRƯƠNG LÊ MINH
SE160798

ĐOÀN GIA BẢO
SE161370

NGÔ THỊ KIM LOAN
SE150874

NHÂM HÁN ĐẠT
SE161324

NGUYỄN HOÀNG VIỆT
SE160991

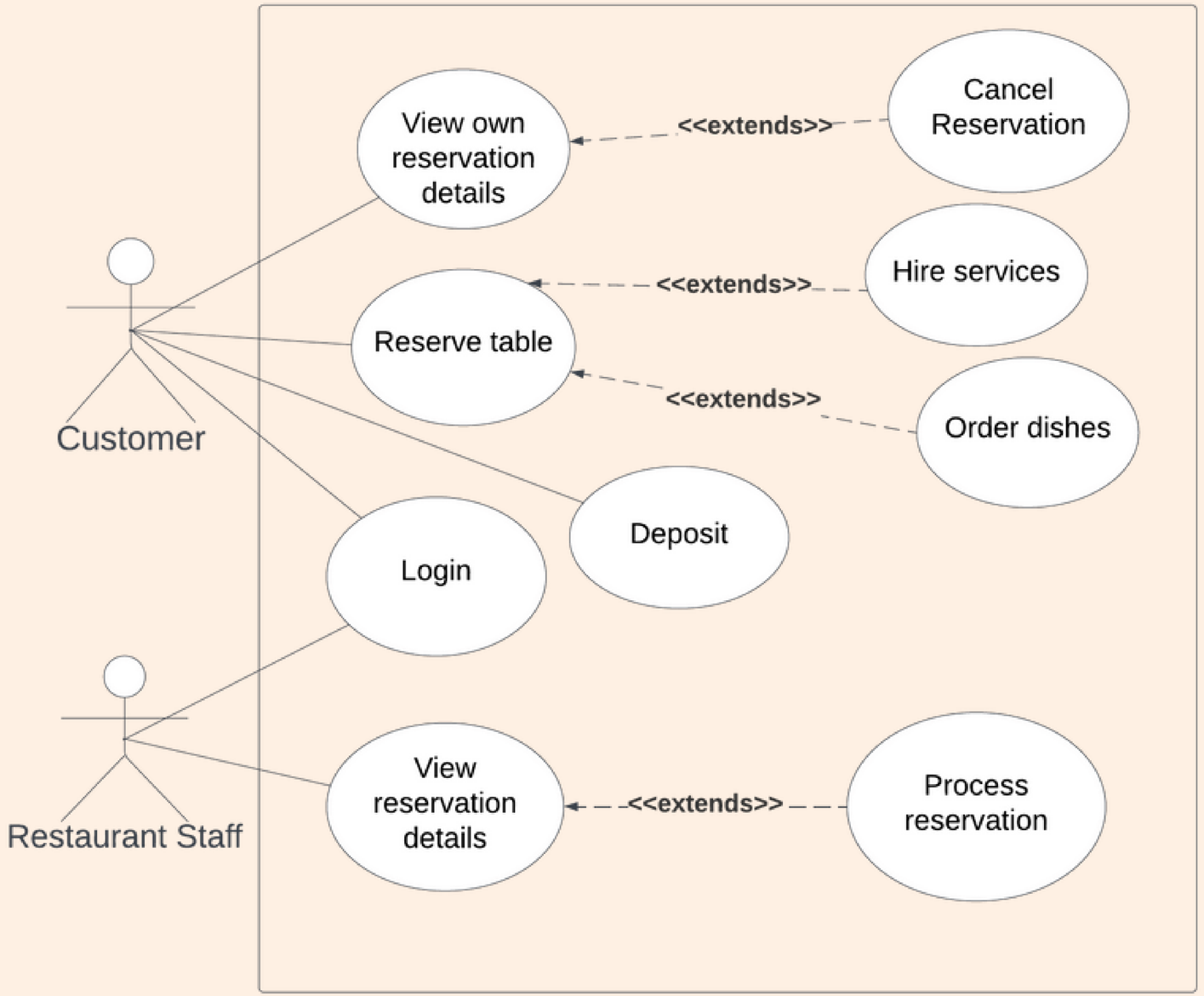
TABLE OF CONTENTS

PROBLEM DESCRIPTION

- Allow customers to reserve tables based on their preferred date, time, number of people, and cuisine type
- Display the restaurant’s menu, venues, location, and contact details to help customers make an informed decision.
- Allow customers to pre-order dishes and extra services for their reservation.
- Allow customers to view their reservation details and a cancellation link.
- Allow customers to cancel their reservation online up to a certain time before their booking.
- Update the restaurant’s inventory of tables in real-time and prevent overbooking.
- Allow the restaurant staff to manage the upcoming reservations and any cancellations made by the customers.

USE CASES

USE CASE MODEL



Actors

CUSTOMER

- Logins to the software
  - Reserves tables matching preferred check-in time, number of guests, restaurant's venues
- Pre-orders dishes, extra services and receives price estimation
- Views own reservation details
- Deposit to complete reserving tables of more than 10 people
- Cancels own reservation

RESTAURANT STAFF

- Logins to the software
- Views reservations of all status including reserved, pending, cancelled...
- Approve/Reject reservations that need processing (reservations of more than 10 people)

Use Case Model

LOGIN

Summary: Customer logins to the software
Actor: Customer

Main Sequence:

1. Clicks "Login" button
2. Enters username, password
3. System displays message and grant access if logins successfully

RESERVE TABLE

Summary: Customer reserves a table
Actor: Customer

Main Sequence:

1. Clicks "Make A Reservation"
2. Enters reservation information such as date, time, number of people
3. Customer can order dishes and services
4. System receives information details and displays reservation status

VIEW RESERVATION

Summary: Customer views own reservations of all status
Actor: Customer

Main Sequence:

1. Clicks "My Reservations"
2. System displays customer's reservations details categorized by status
3. Customer can cancel reservation or deposit to complete reservation of more than 10 people

DEPOSIT

Summary: Customer deposits to complete a reservation of more than 10 people
Actor: Customer
Main Sequence:

1. Views reservation of "Pending deposit" status
2. Clicks "Deposit"
3. System directs customer to payment and redirect to the software afterwards
4. System notifies customer and changes status of reservation to "Reserved"

Use Case Model

LOGIN

Summary: Restaurant staff logins to the software
Actor: Restaurant Staff

Main Sequence:

1. Clicks "Login" button
2. Enter username, password
3. System displays message and grant access if logins successfully

VIEW RESERVATIONS

Summary: Restaurant staff views reservations of all status
Actor: Restaurant Staff

Main Sequence:

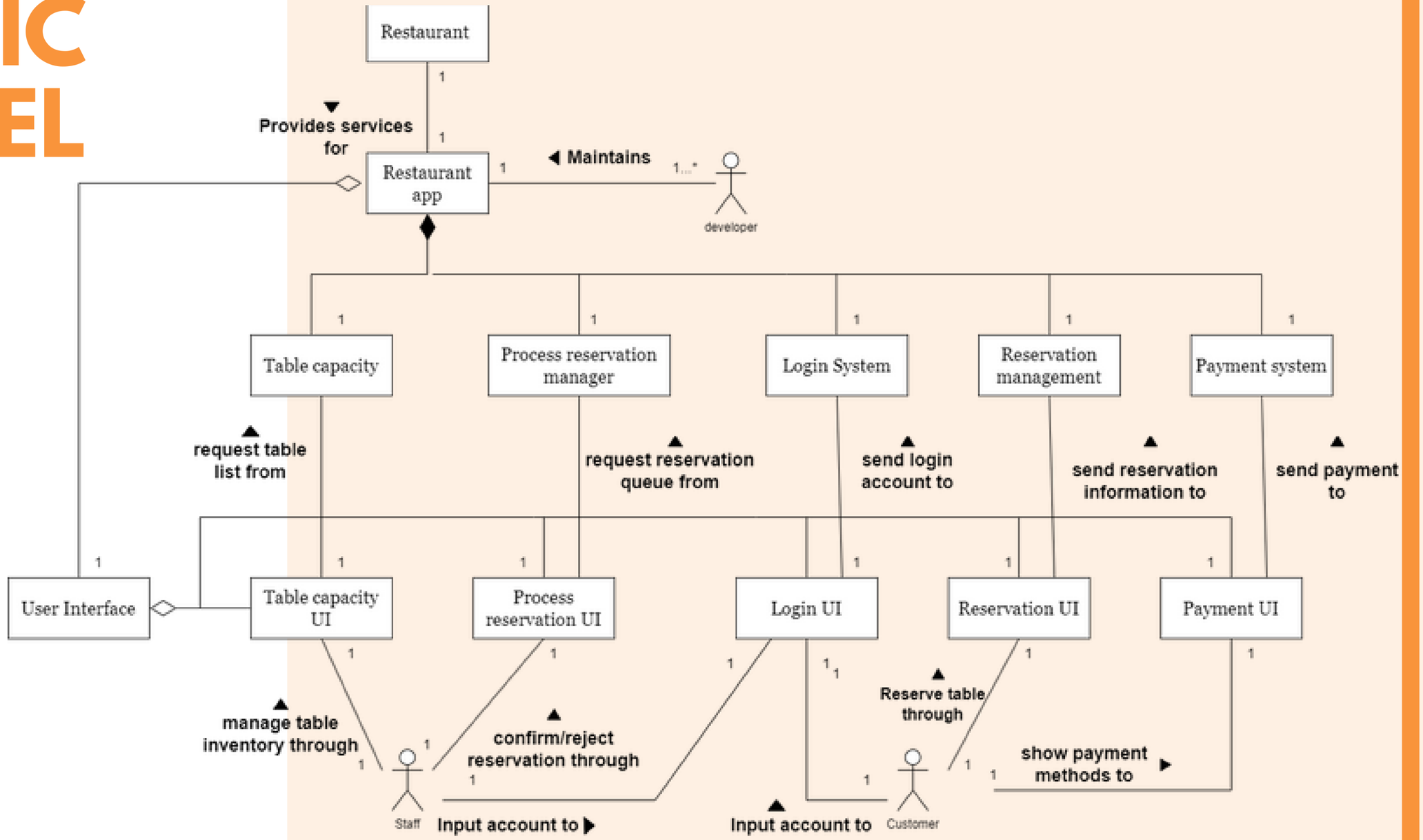
1. Clicks "Manage Reservation"
2. System displays all customers' reservations details categorized by status

PROCESS RESERVATIONS

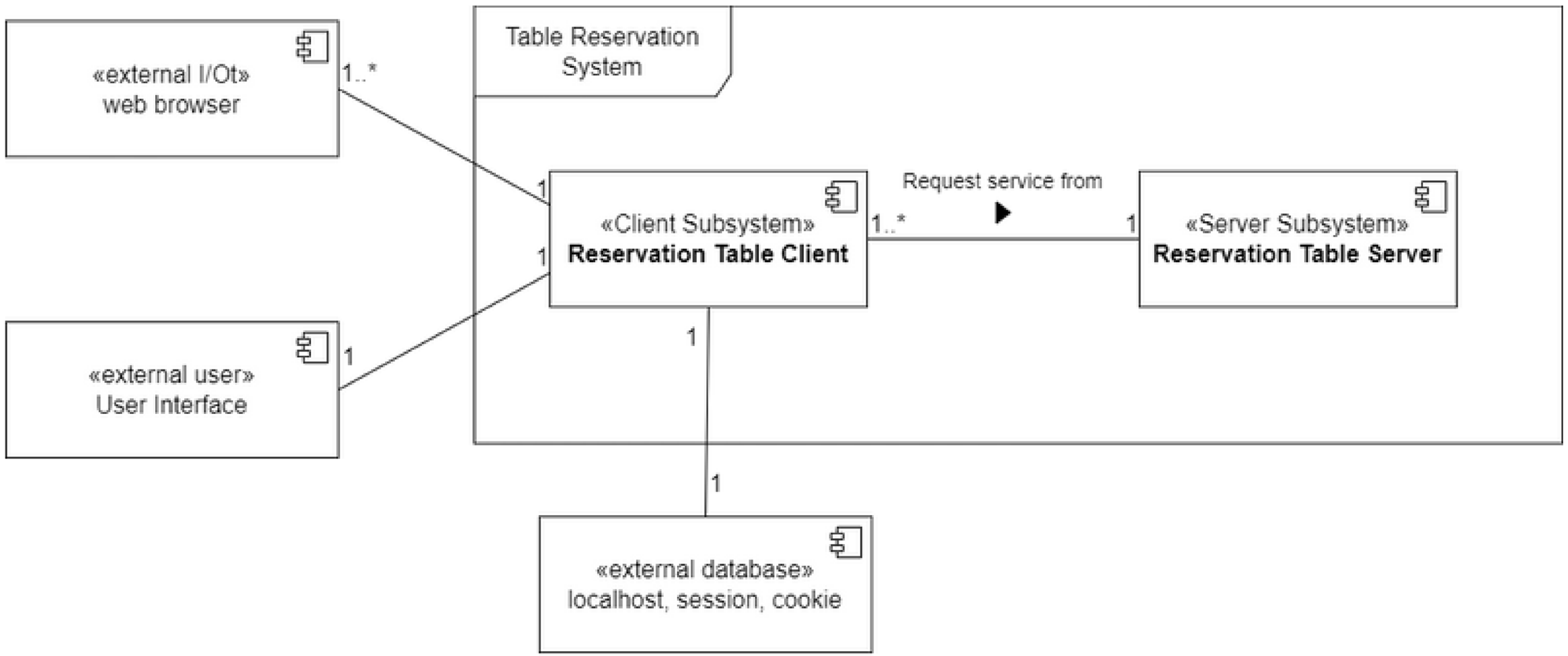
Summary: Restaurant staff approves/rejects reservation of more than 10 people
Actor: Restaurant Staff
Main Sequence:

1. Clicks "Manage Reservation"
2. Switch to "Pending processing" reservations tab
3. View reservations details (including dishes and services if has) and clicks "Approve"/"Reject"
4. System sends deposit request to customer

STATIC MODEL



OBJECT STRUCTURING



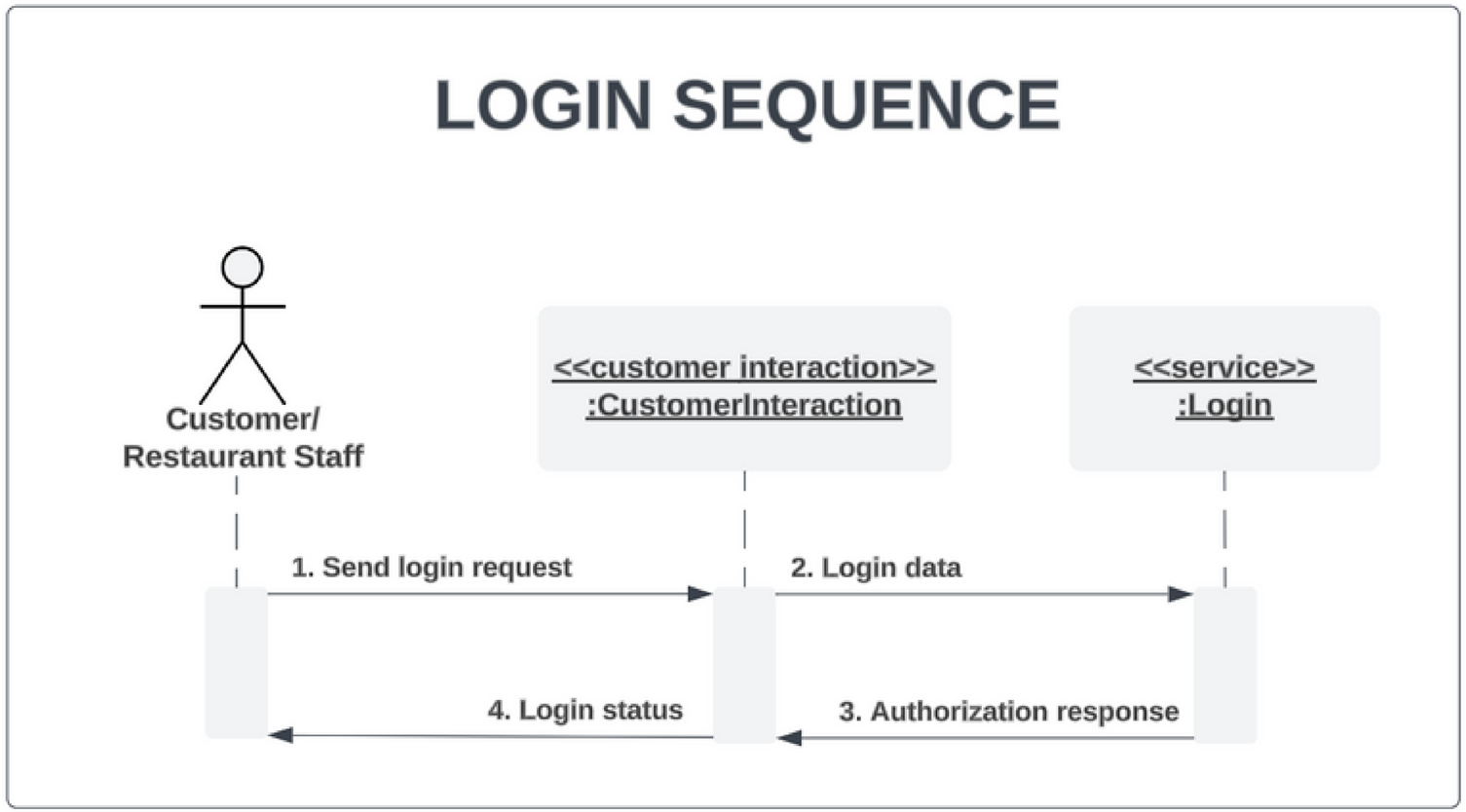
DYNAMIC MODEL

THE DYNAMIC MODEL DEPICTS THE INTERACTION AMONG THE OBJECTS THAT PARTICIPATE IN EACH USE CASE.

DYNAMIC MODEL

THE DYNAMIC MODEL DEPICTS THE INTERACTION AMONG THE OBJECTS THAT PARTICIPATE IN EACH USE CASE.

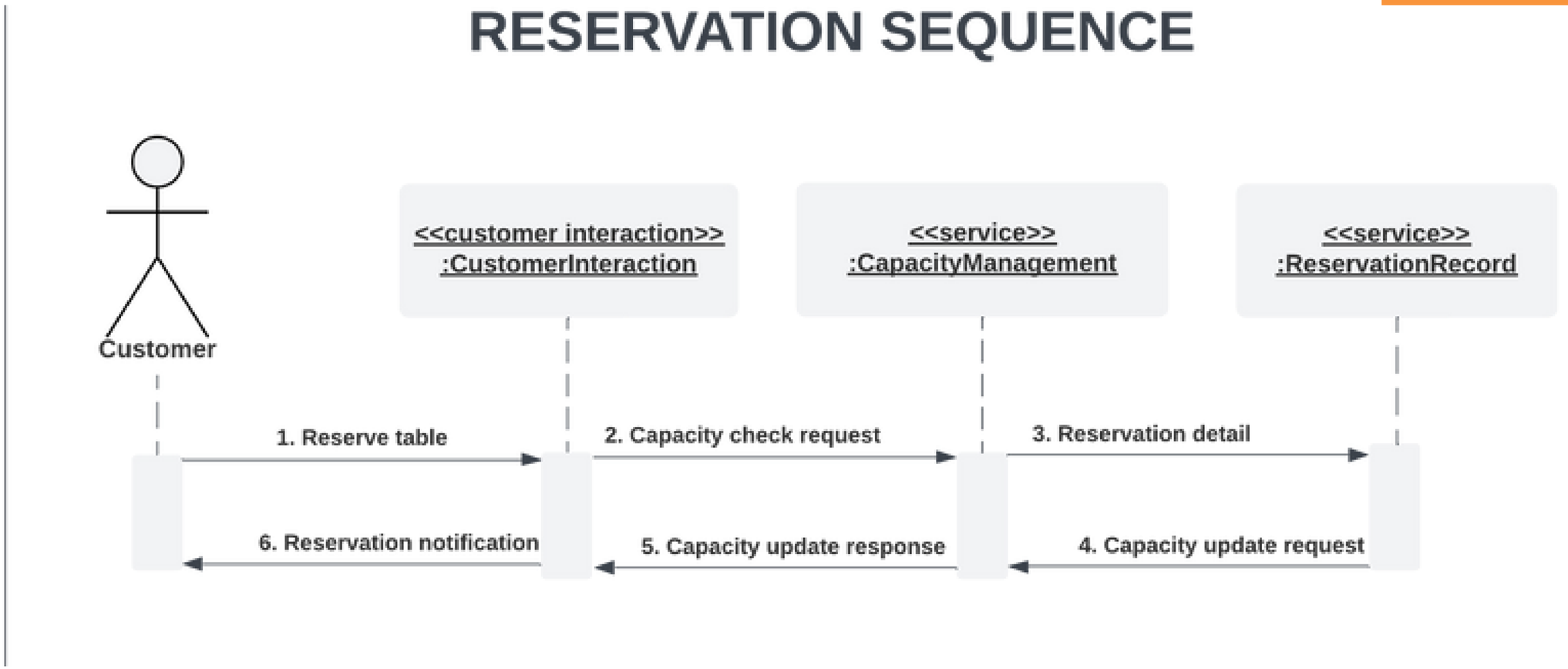
LOGIN



DYNAMIC MODEL

THE DYNAMIC MODEL DEPICTS THE INTERACTION AMONG THE OBJECTS THAT PARTICIPATE IN EACH USE CASE.

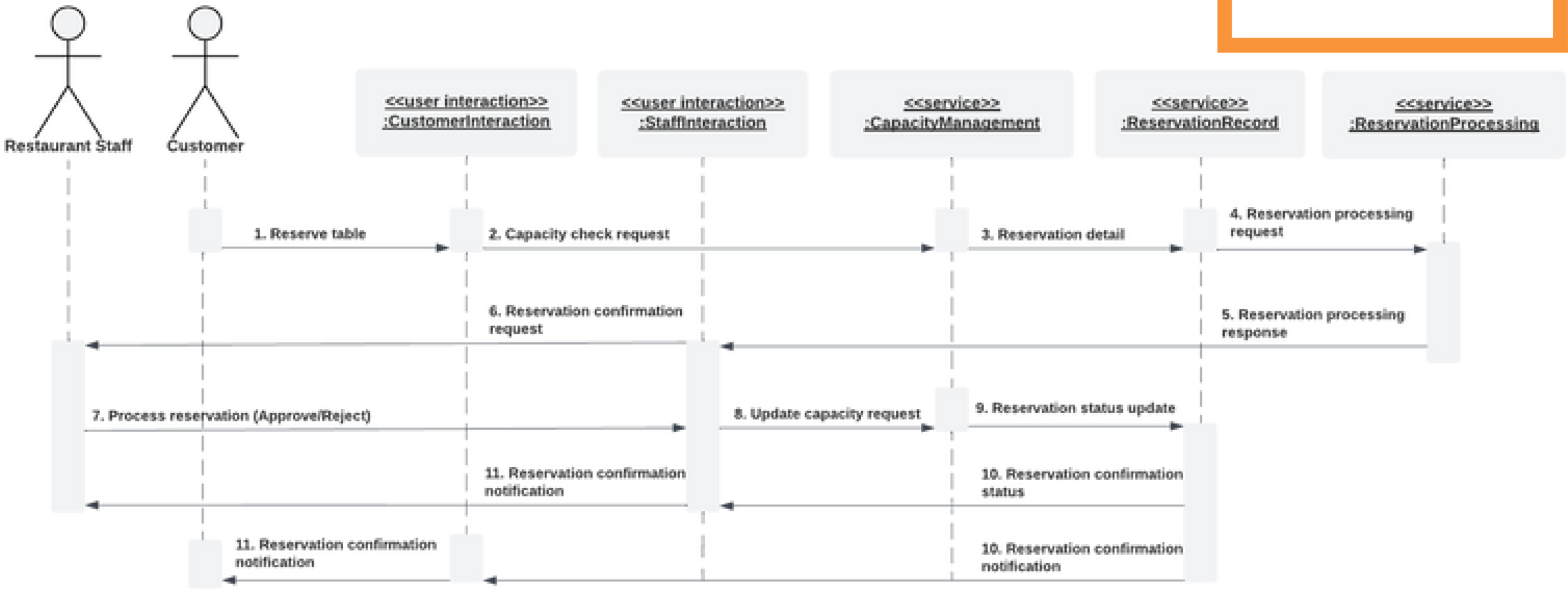
RESERVATION



DYNAMIC MODEL

THE DYNAMIC MODEL DEPICTS THE INTERACTION AMONG THE OBJECTS THAT PARTICIPATE IN EACH USE CASE.

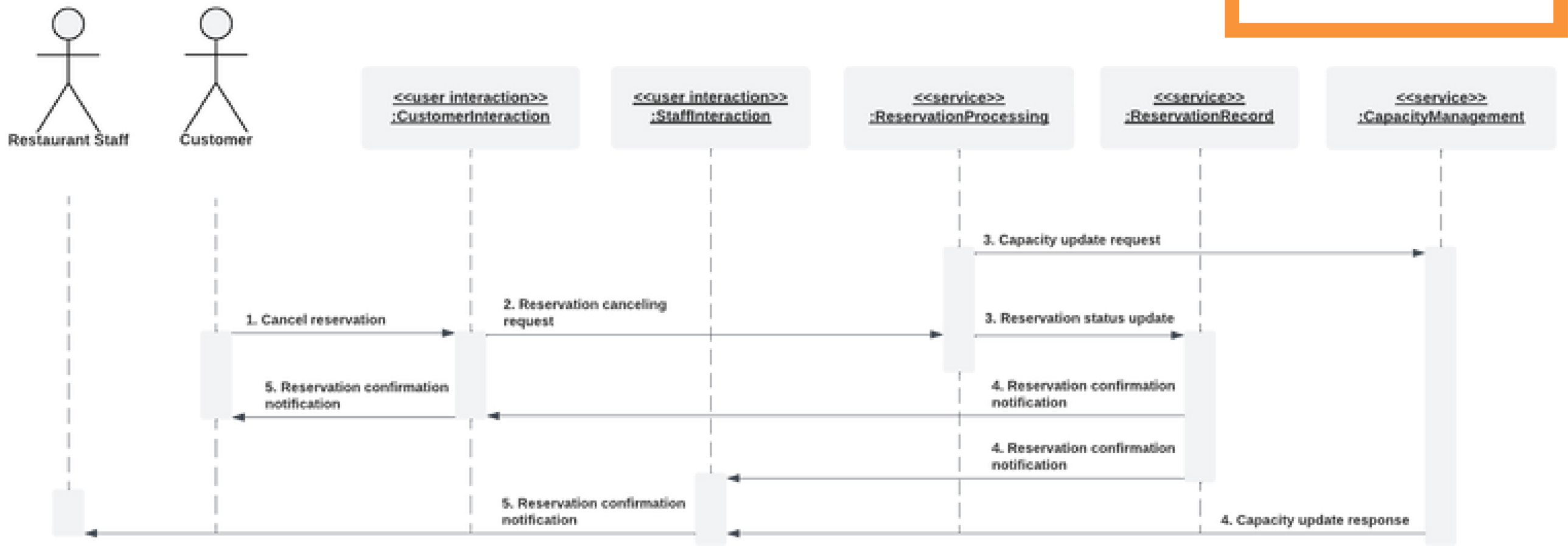
RESERVATION ALTERNATIVE



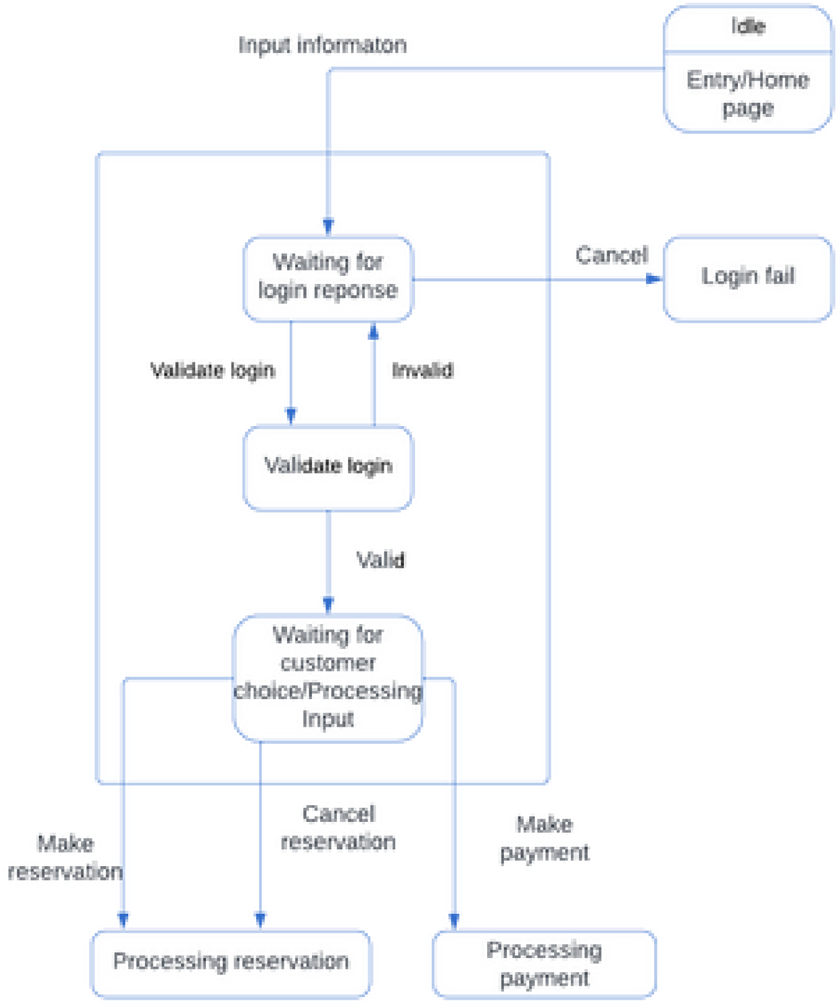
DYNAMIC MODEL

THE DYNAMIC MODEL DEPICTS THE INTERACTION AMONG THE OBJECTS THAT PARTICIPATE IN EACH USE CASE.

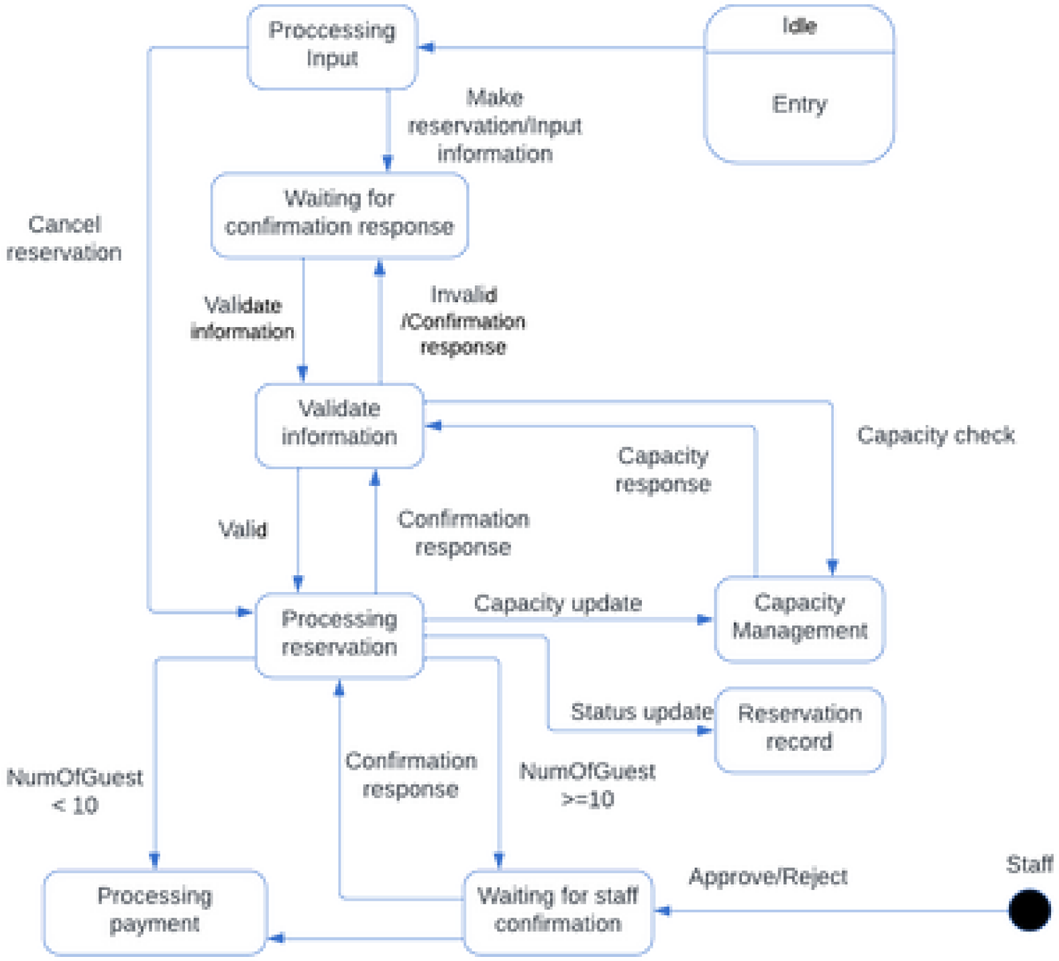
RESERVATION CANCELING



STATE CHART



State chart for Table reservation: Login



State chart for Table reservation: Make reservation, Cancel reservation

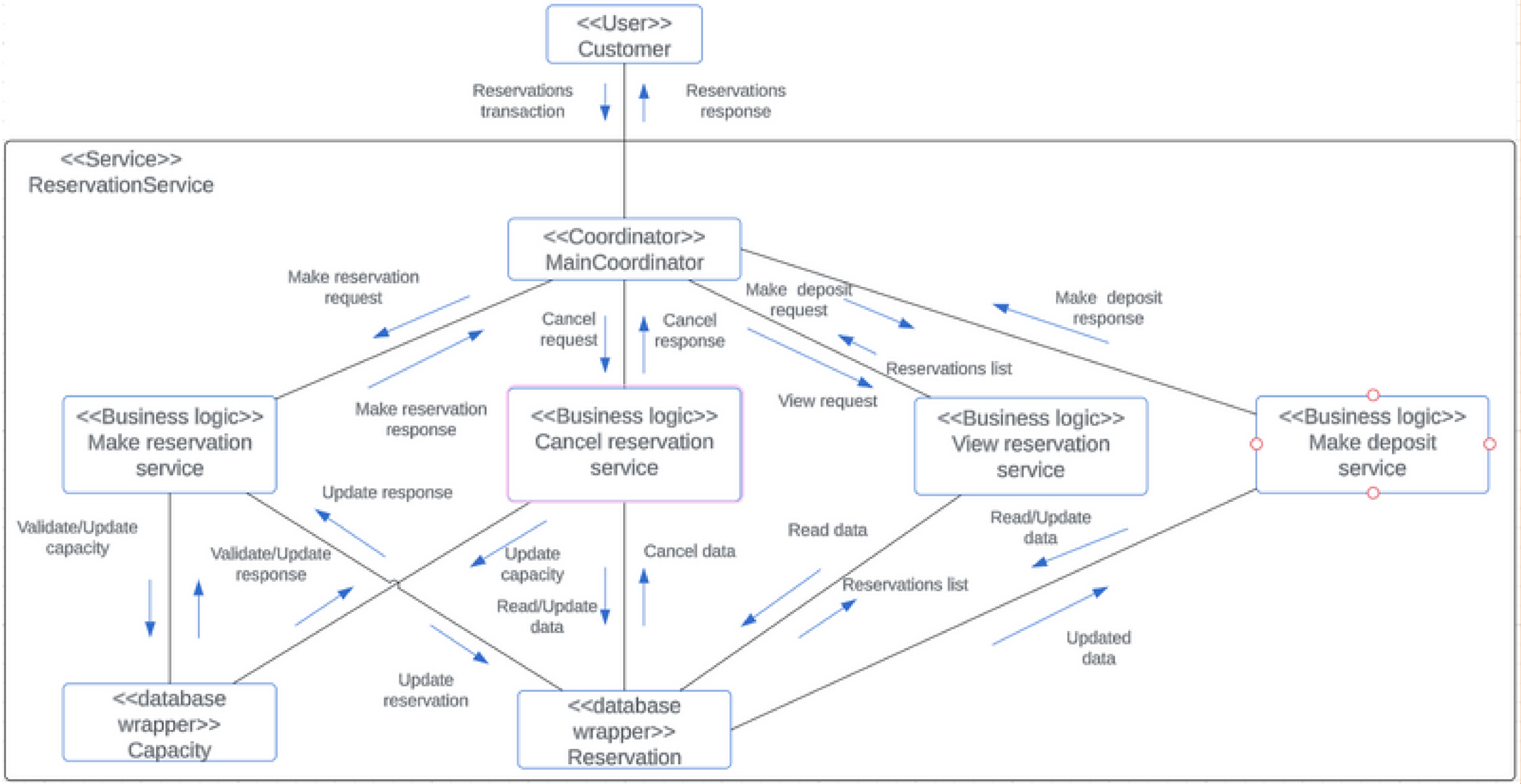
TABLE RESERVATION SYSTEM DESIGN

THE ANALYSIS MODEL OF THE TABLE RESERVATION SYSTEM IS MAPPED TO A DESIGN MODEL. THE STEPS IN THIS PROCESS ARE AS FOLLOWS:

1. Integrate the communication model. Develop integrated communication diagrams

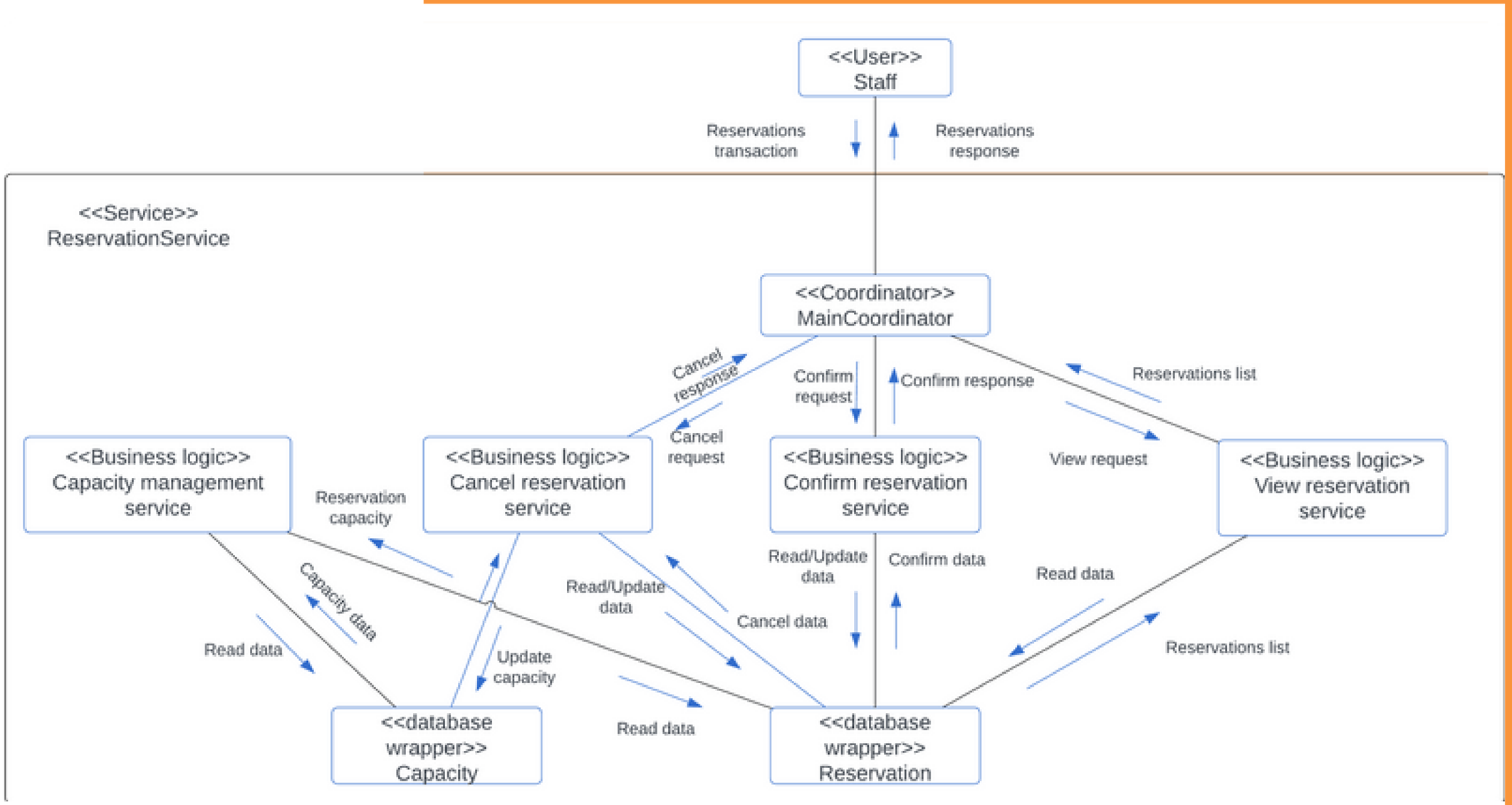
2. Develop the detailed software design.

COMMUNICATION MODEL



Communication diagram for Table Reservation system User Customer

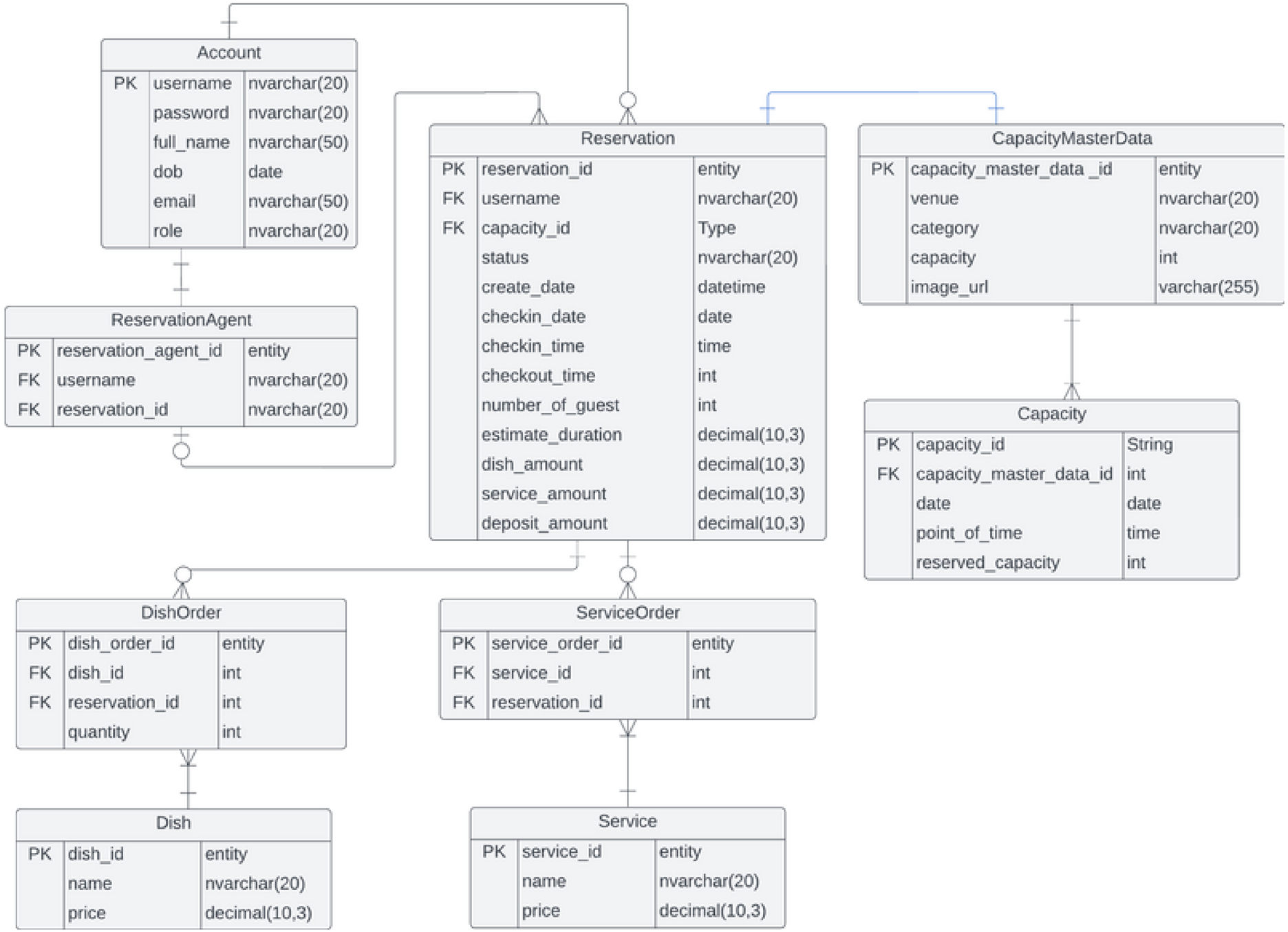
COMMUNICATION MODEL



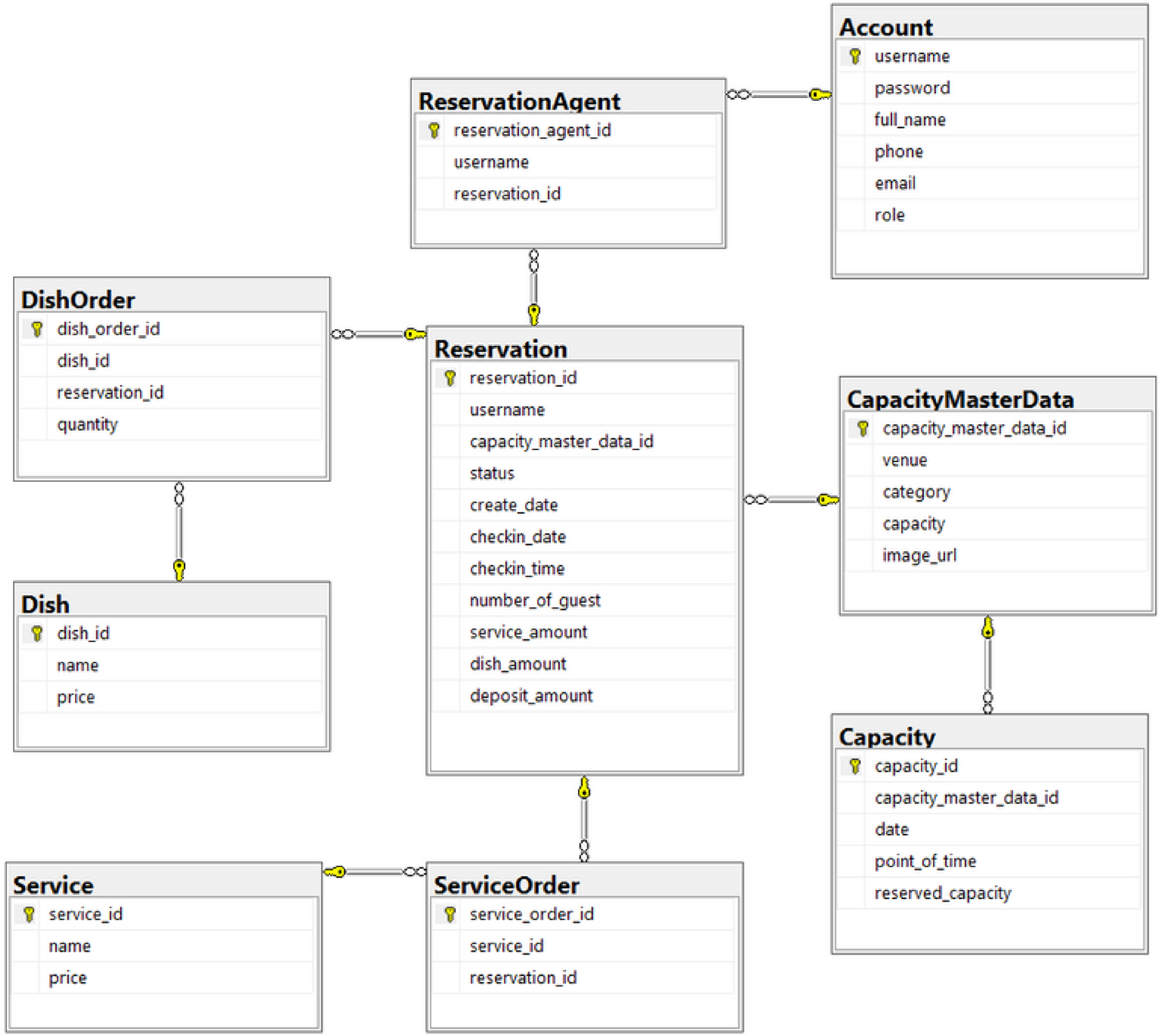
Communication diagram for Table Reservation system User Staff

RELATIONAL DATABASE DESIGN

RELATIONAL DATABASE DESIGN



RELATIONAL DATABASE DESIGN



DETAILED DESIGN

Detailed Design

RESERVE TABLE (customer)

wait (making reservation)

if reservation is validated

then if reservation quantity is smaller than 10

Write reservation to database with status "Reserved"

else write reservation to database with status "Pending processing"

Send customer to select dishes and services view

If customer want to select dish and services

Then Wait customer select;

Add dishes and services to database mapping with the reservation

else skip

STAFF CONFIRM (staff)

wait new "pending processing" reservation

if restaurant services are available to reserve the reservation

Update reservation into database with status "Pending deposit"

else update reservation into database with status "cancel"

Technology

Java

_ Strong Tech for developing Web-based Application
_ Open source with a strong community
_ An object-oriented

Springboot

_ Provides Embedded HTTP servers like tomcat

Microsoft SQL server

_ Highly data secure
_ Excellent Data Restoration and Recovery Mechanism

ReactJS

_ Building user interfaces based on components

TESTING

TEST CASES

LOGIN

Summary: Check for login function
Actor: All
Case 1: Test login successfully
Description: send username and password to login, if successfully return response with status 200
Expected 01: 200 status
Case 2: Test login fail
Description: send username and password to login, if fail return response with status 401 (Unauthorized)
Expected 02: 401 status

MAKE RESERVATIONS

Summary: Test about customer create a reservation
Actor: Customer

Case: Create reservation with full data, if create successfully return status 200 with reservation_id
Expected:
200 status
reservation_id

PROCESS RESERVATIONS

Summary: Test about staff confirm and cancel reservation
Actor: Restaurant staff
Case 1: Change status of reservation base on request, if request is cancel -> change reservation into "canceled"
Expected 01: 200 status
Case 2: Change status of reservation base on request
if request is approve -> change reservation status into "pending deposit"
Expected 02: 200 status

MANAGE CAPACITY

Summary: Test staff get restaurant capacity base on reservation
Actor: Restaurant staff
Case: Request server to get capacity base on checkin date, check-in time and customer venue
Expected: 200 status with capacity in body

DEMO TEST CASES

```java
@Test
public void testStatusAccountStaff() throws Exception {
    setUp();
    String[] usernames = {"staff", "Customer1","staff_manager"};
    for (String user: usernames) {
        String uri = "/account/" + user;
        MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.get(uri)
                .accept(MediaType.APPLICATION_JSON_VALUE)).andReturn();

        int status = mvcResult.getResponse().getStatus();
        Assertions.assertEquals(200, status);
        String content = mvcResult.getResponse().getContentAsString();
        Account account = mapFromJson(content, clazz:Account.class);
        Assertions.assertEquals(account.getUsername(), user);
    }
}
//----------------------------Test Reservation-------------------------------------------------------
List<Reservation> reservations = new ArrayList<>();
@Test
public void testAddReservation() throws Exception {
    setUp();
    Reservation reservation = new Reservation();
   reservation.setUsername(username:"customer 11");
   reservation.setStatus(status:"reserved");
   reservation.setNumberOfGuest(numberOfGuest:15);
   reservation.setCapacityMasterDataId(capacityMasterDataId:3);
   reservation.setCreateDate(new Date(2023-06-03));
   reservation.setCheckinDate(new Date(2023-06-10));
   reservation.setCheckinTime(new Time(16,00,00));
    String uri = "/create-reservation";
        MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.post(uri)
                .contentType(MediaType.APPLICATION_JSON_VALUE).content(mapToJson(reservation))).andReturn();

        int status = mvcResult.getResponse().getStatus();
        Assertions.assertEquals(200, status);
        if (status == 200) {
            String content = mvcResult.getResponse().getContentAsString();
            Reservation returnReservation = mapFromJson(content, clazz:Reservation.class);
            if (returnReservation != null && returnReservation.getReservationId() != 0){
                Assertions.assertEquals("returnReservation","returnReservation");
            } else Assertions.assertEquals("returnReservation", "null");
            reservations.add(returnReservation);
            testDeleteReservation();
        }

}
```

```java
@Test
public void testDeleteReservation() throws Exception {
    System.out.println("This is size()" + reservations.size());
    String uri = "/";
    for (Reservation reservation :
            reservations) {
        MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.post(uri + reservation.getReservationId() +"/cancel")
                ).andReturn();
        System.out.println(reservation);
        int status = mvcResult.getResponse().getStatus();
        Assertions.assertEquals(200, status);
    }

}
@Test
public void testViewDishes() throws Exception {
    setUp();
        String uri = "/dishes";
        MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.get(uri)
                .accept(MediaType.APPLICATION_JSON_VALUE)).andReturn();

        int status = mvcResult.getResponse().getStatus();
        Assertions.assertEquals(200, status);
        String content = mvcResult.getResponse().getContentAsString();
}
```

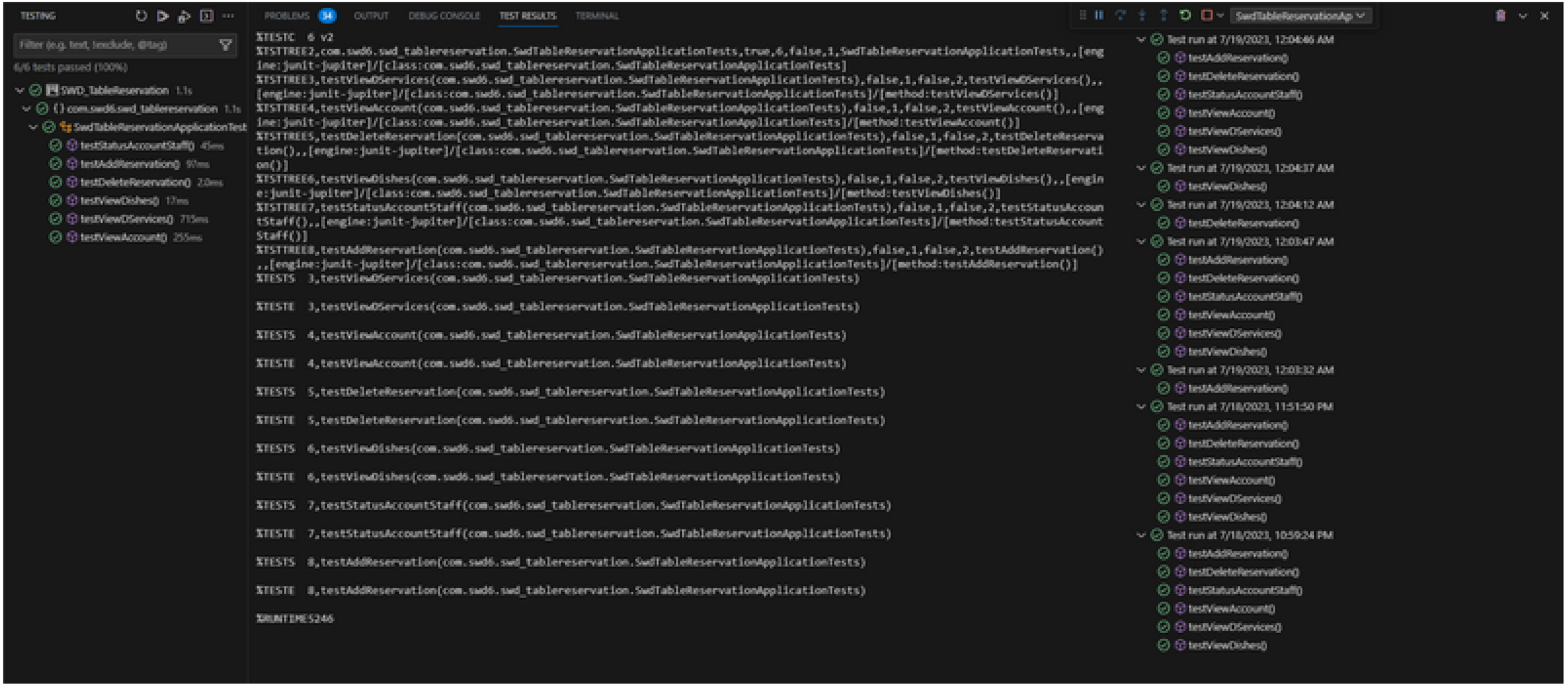
```java
@Test
public void testViewDServices() throws Exception {
    setUp();
    String uri = "/services";
    MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.get(uri)
            .accept(MediaType.APPLICATION_JSON_VALUE)).andReturn();

    int status = mvcResult.getResponse().getStatus();
    Assertions.assertEquals(200, status);
    String content = mvcResult.getResponse().getContentAsString();
}
@Test
public void testViewAccount() throws Exception {
    setUp();
    String[] usernames = {"Customer2", "Customer1", "Customer3", "Customer6"};
    for (String user : usernames) {
        String uri = "/reservations/username?username=" + user;
        MvcResult mvcResult = mvc.perform(MockMvcRequestBuilders.get(uri)
                .accept(MediaType.APPLICATION_JSON_VALUE)).andReturn();

        int status = mvcResult.getResponse().getStatus();
        Assertions.assertEquals(200, status);
        String content = mvcResult.getResponse().getContentAsString();
        if (content.length() > 0) Assertions.assertEquals("testViewAccount", "testViewAccount");
        else Assertions.assertEquals("testViewAccount", "\"\"");
    }
}

}
```

DEMO TEST CASES



DEMO APPLICATION


THANK
YOU