SOFTWARE DOCUMENT

# INTRODUCTION TO RESERVATION RESTAURANT ARCHITECTURE SYSTEM

# *Team Members*

**Nguyen Thanh Dang Khoa**

SE150842

**Bui Duc Phong**

SE150977

**Su Thanh Tra**

SE150546

**Nguyen Thanh Trung**

SE160650

**Ho Hoang Hiep**

SE150764

**Dang Van Thien**

SE160654

# INTRODUCE

03

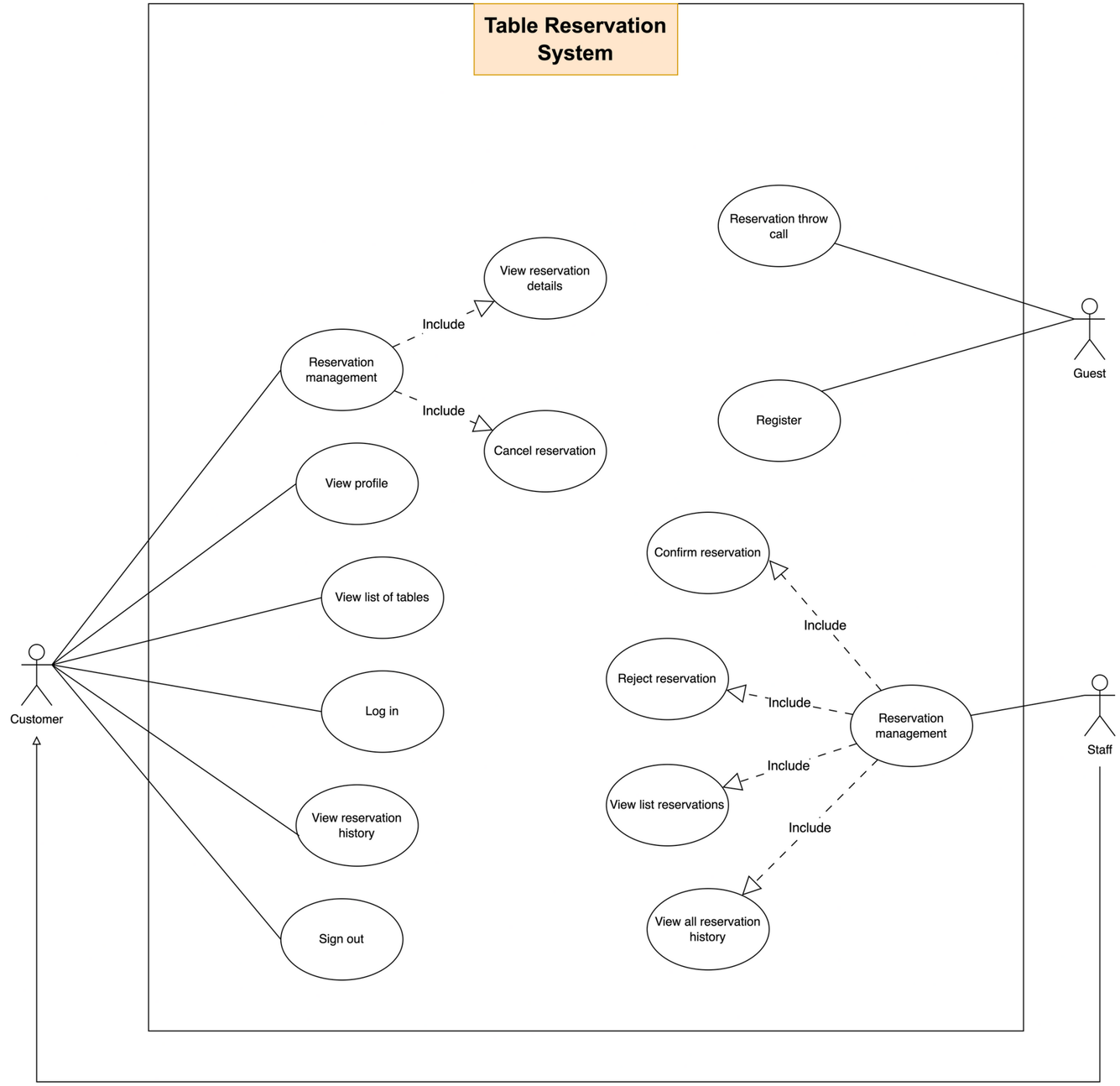
## TABLE RESERVATION SYSTEM

Table reservation system is born to help customers and restaurant staff reserve tables and manage their reservations in a convenient but also effective manner

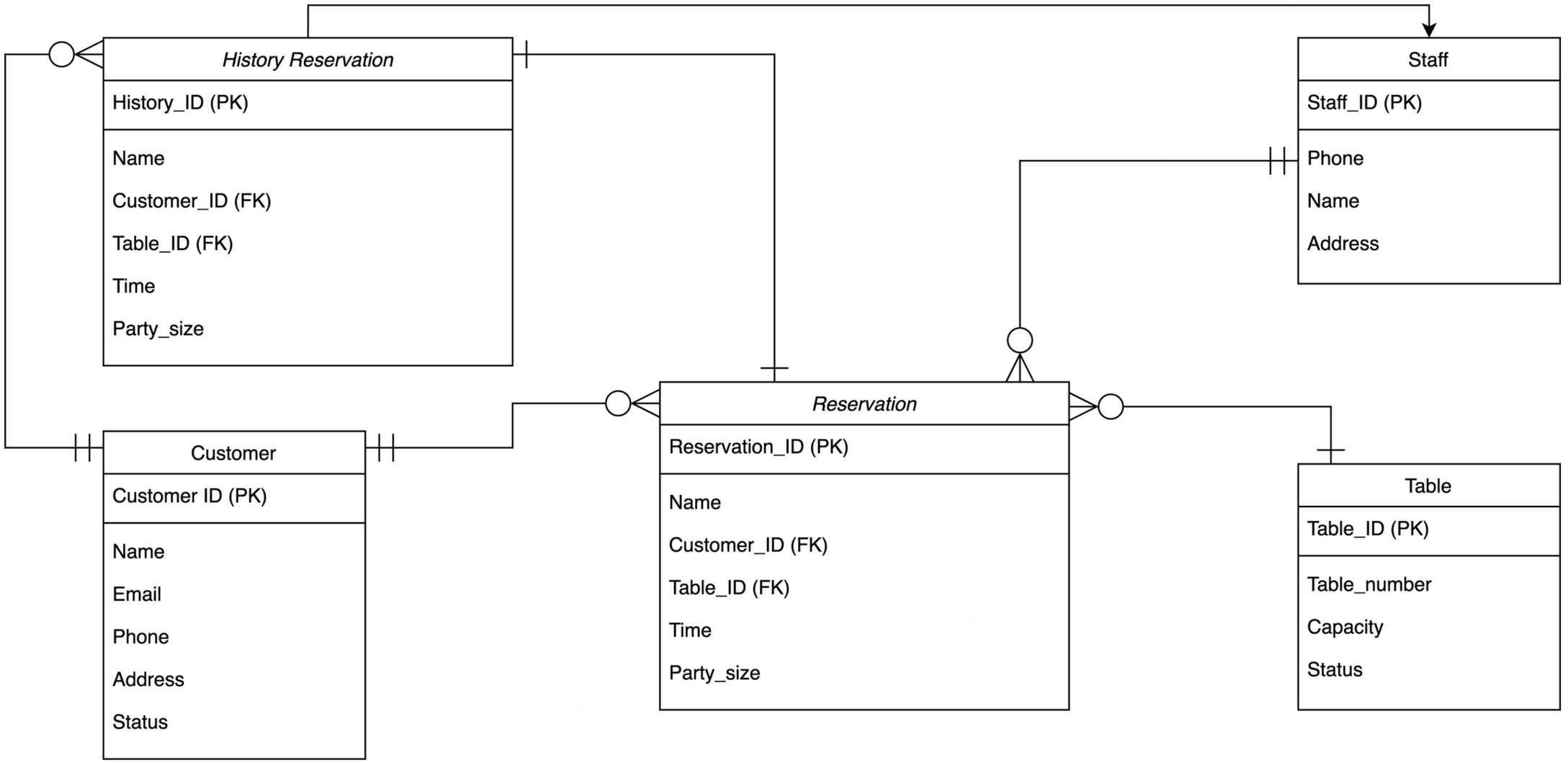
# Requirement

- As guest, I can ceate account
- As guest, I can call to create a reservation
- As user, I can login
- As user, I can view my profile
- As user, I can choose time and number of people in a reservation
- As user, I can view my reservation history
- As user, I can create a reservation once at time an wait staff call to comfirm
- As staff, I can create a reservation
- As staff, I can confirm or reject a reservation throw call
- As staff, I can view all reservation history
- A reservation can have 5 statuses "Pending, Accepted, Rejected, Completed, Cancelled"
- A reservation must be made at least 30 minutes earlier, at most 1 minute later

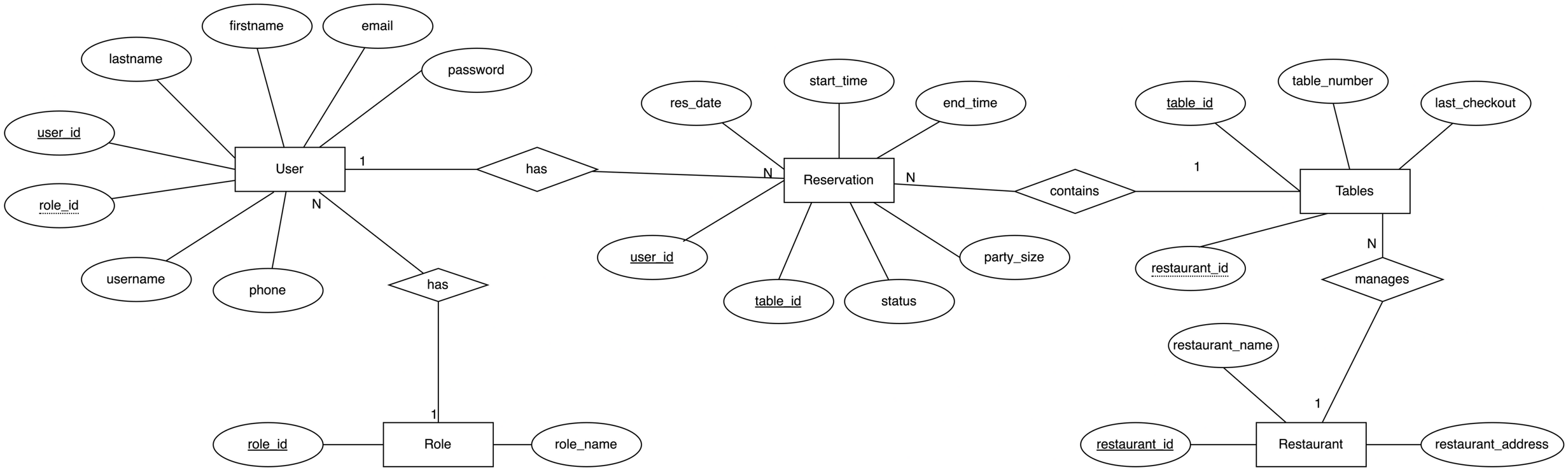
# Usecase Diagram



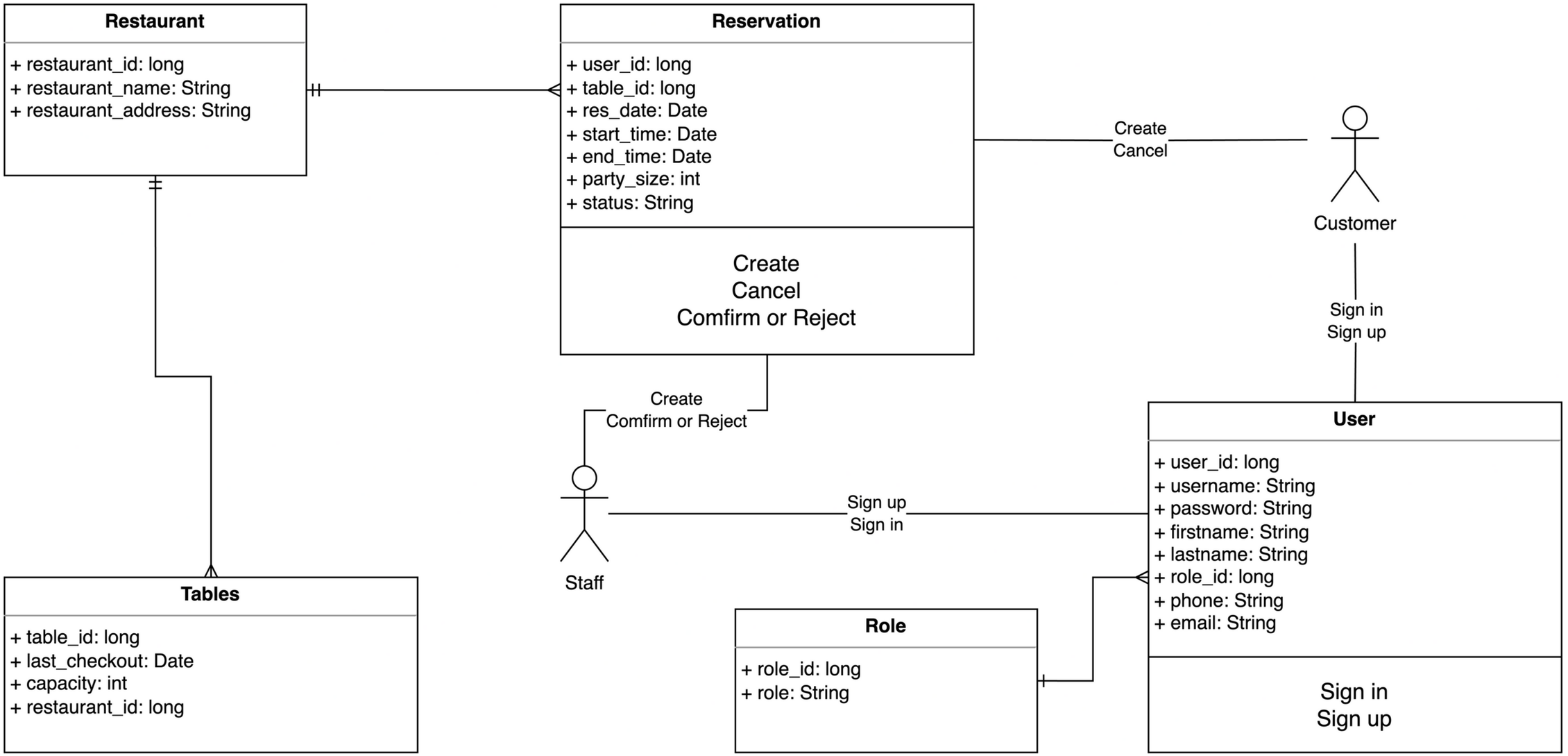
# Relational Database



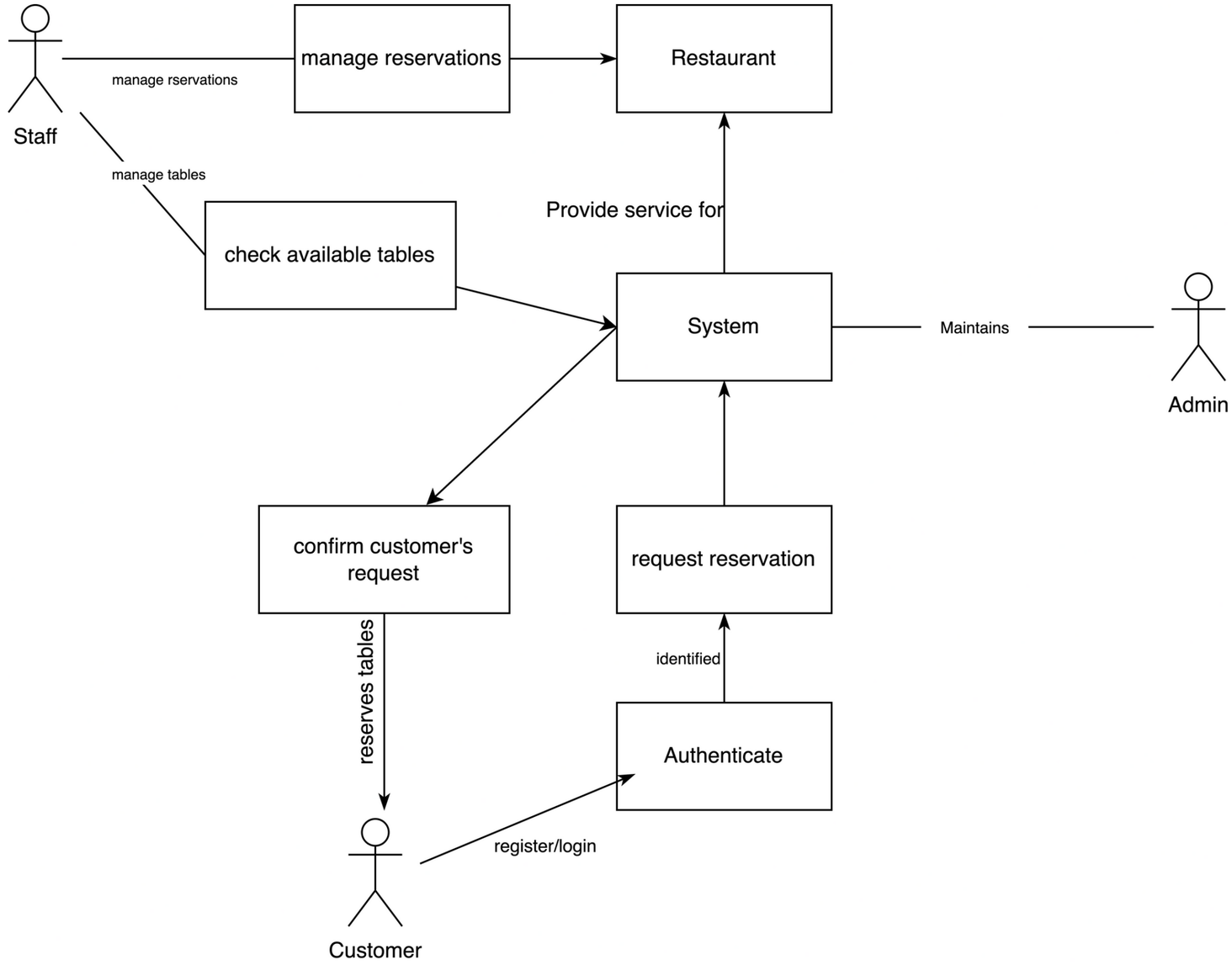
# ERD


firstname
email
lastname
password
user_id
role_id
User
1
N
has
username
phone
has
1
role_id
Role
role_name
res_date
start_time
end_time
N
Reservation
N
contains
user_id
table_id
status
party_size
table_id
table_number
last_checkout
1
Tables
N
restaurant_id
manages
restaurant_name
1
restaurant_id
Restaurant
restaurant_address

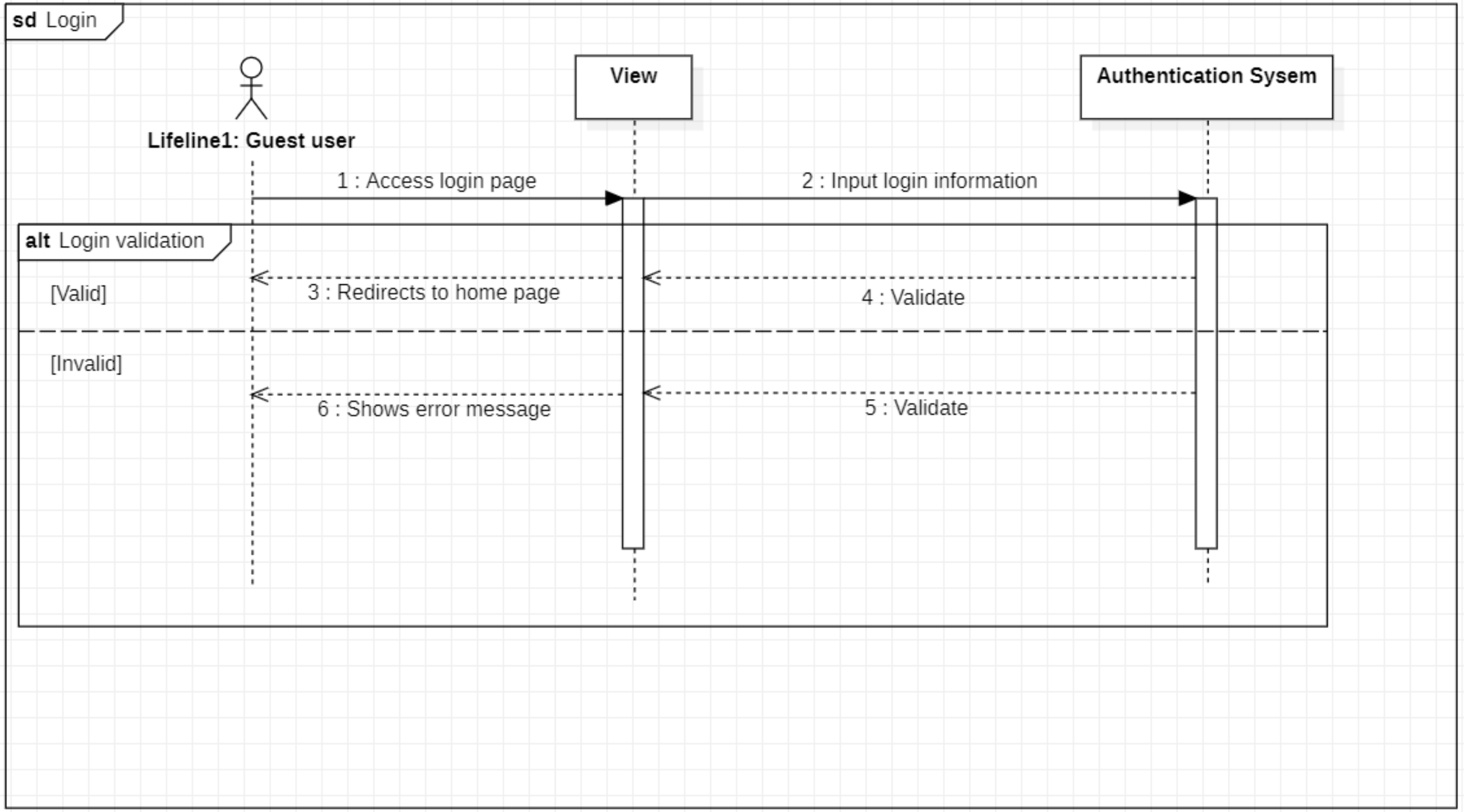
# Class diagram



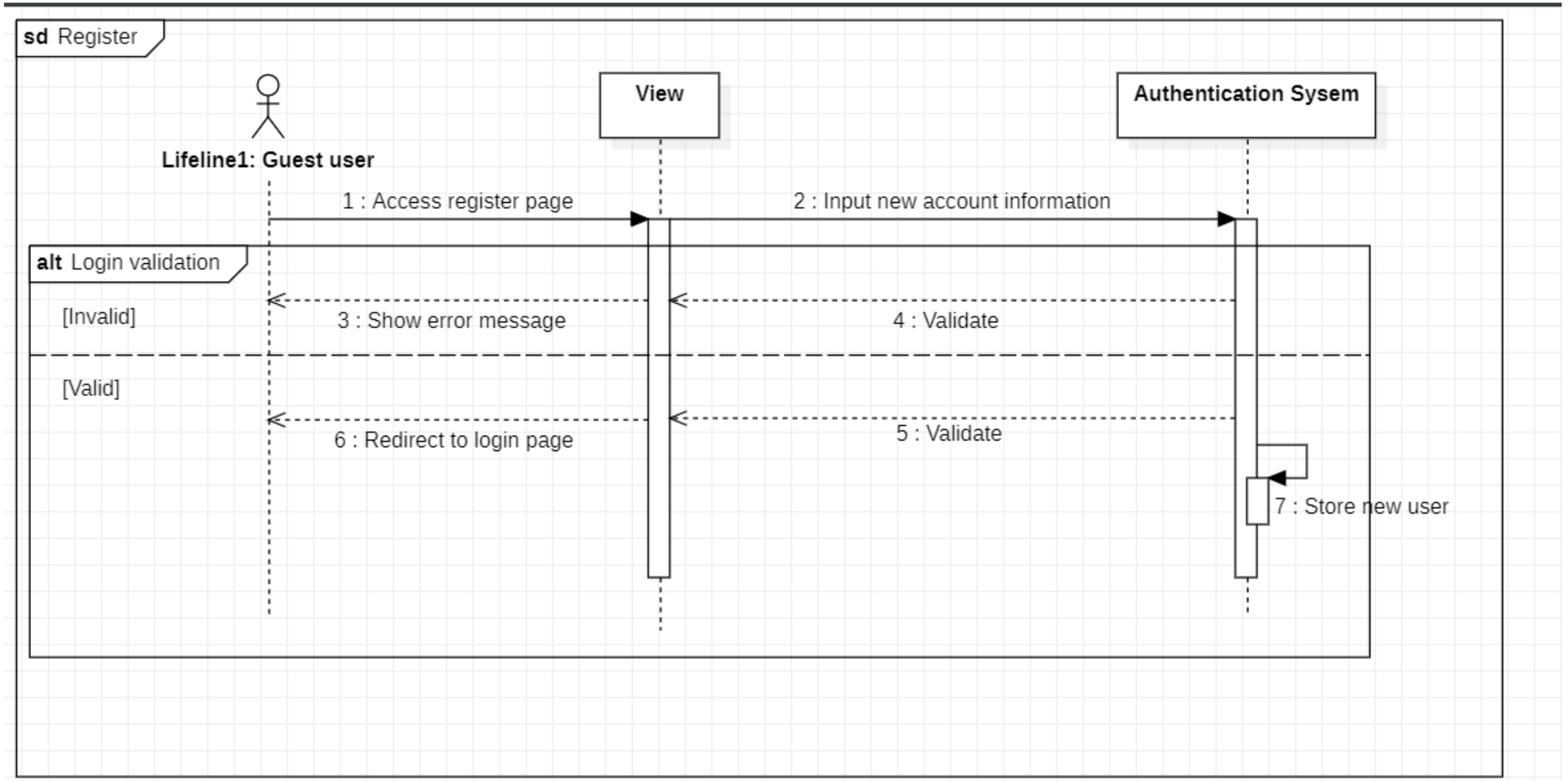
# Static Modeling



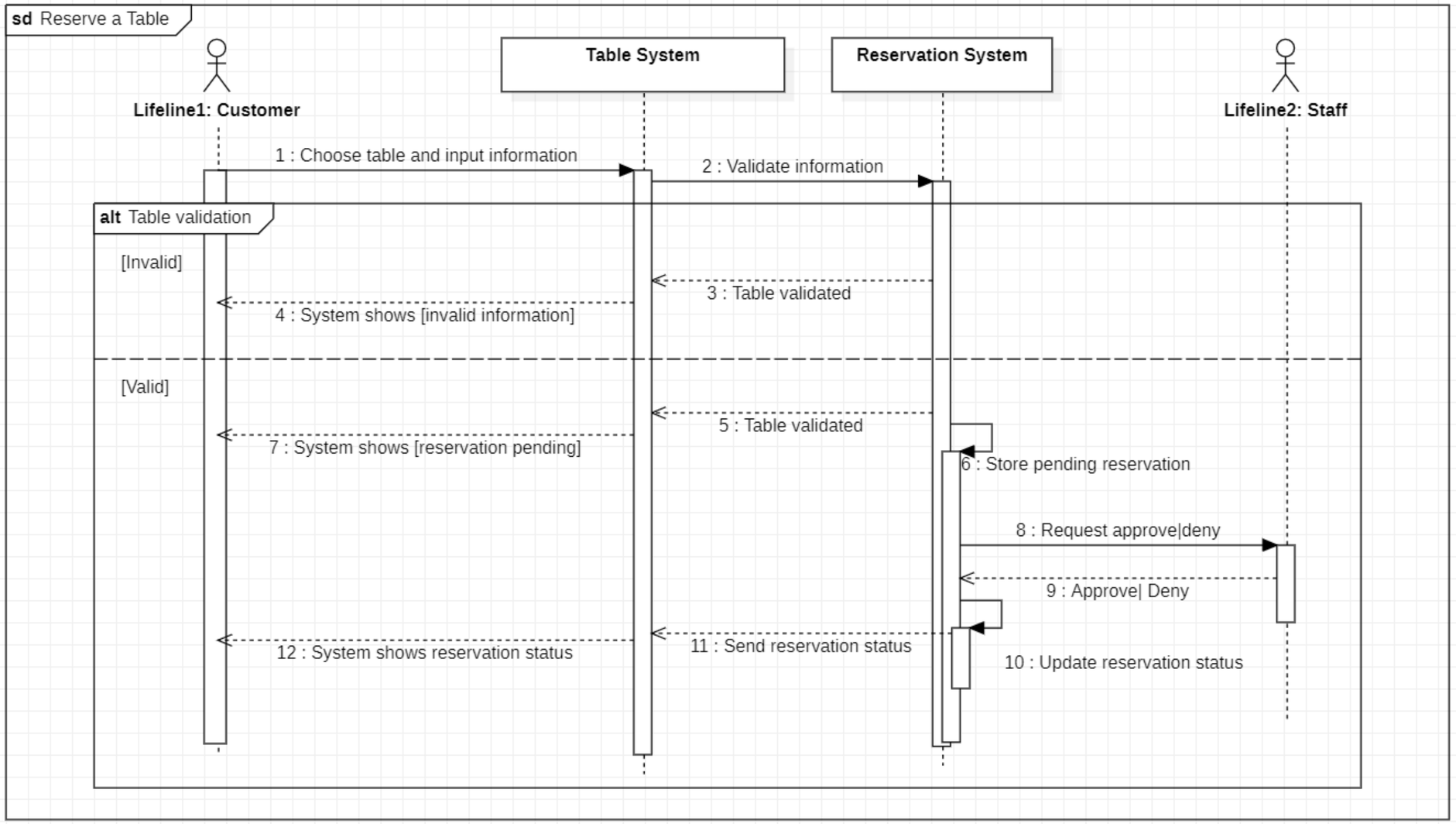
# Sequence Diagram - Login



# Sequence Diagram - Register



# Sequence Diagram - Reserve a Table



# USECASE MODEL

**Use case name:** Login

**Summary:** Allows the user to login and use the system

**Actor:** Guest (unlogged in) users

**Dependency:** none

**Precondition:** User is at the login page

**Main sequence:**

1. User enters username and password

Username must have 3 characters or more

Password must have 8 characters or more

Username and passsword must match already existing usernames and passwords in the system

2. User clicks "LOGIN"

3. The authentication system validates user information

4. User logs in successfully and is redirected to the home page

**Alternative sequence:**

**Step 3:** if the information is invalid, the system displays an error message and asks the user to input again. There is no "reset password" feature

**Postcondition:** User is at the home page

# USECASE MODEL

**Use Case Name:** Register

**Summary:** Allows the user to create a new customer account to use the system

**Actor:** Guest (unlogged in) users

**Precondition:** User is at the register page

**Main Sequence:**

1. User enters usernam, password, confirm password, firstname, lastname, email and phone.

Username must have 3 characters or more

Password must have 8 characters or more

Phone must be in number type and must have 10 numbers or more

Username and phone must not match already existing usernames and phones in the system

2. User clicks "Register"

3. The authentication system validates user information

4. User creates a new account successfully and is redirected to the login page

**Alternative Sequence:**

Step 3: if the information is invalid, the system displays an error message and asks the user to input again

**Postcondition:**

User is at the login page

# USECASE MODEL

**Use Case Name:** Make a Reservation

**Summary:** This use case allows a logged in user to make a reservation for the restaurant.

**Actor:** Logged in users

**Precondition:** The user must be logged in into the system and is at the home page

**Main Sequence:**

1.User chooses an available table

2.User enters

3.User enters reservation time, and number of people.

Reservation time must be at least 30 minutes earlier than the current time, and at most a week later

4.The system validates the information

**Alternative Sequence:**

Step 3: if the information is invalid, the system displays an error message and asks the user to input again

**Postcondition:**

A reservation is successfully made and user can see reservation status pending message

# USECASE MODEL

**Use Case Name:** Approve/Deny Reservation

**Summary:** Allows staff to approve or deny a pending reservation

**Actor:** Staff users

**Precondition:** Staff must be logged in into the system and is viewing the reservation's details

**Main Sequence:**

1.Staff clicks on "Approve" or "Reject" button

2.System updates reservation's status.

**Alternative Sequence:**

If the system does not find any existing reservations associated with the user, it displays a message indicating that no reservations were found.

**Postcondition:**

Reservation's status is changed from "pending" to "accepted" or "rejected"

# USECASE MODEL

**Use Case Name:** Cancel Reservation

**Summary:** This use case describes the process of a user and  astaff canceling an existing reservation.

**Actor:** User, staff

**Precondition:** The user and staff must have a reservation that they want to cancel. The user should be logged in the system.

**Main Sequence:**

1. The system displays a list of the user's existing reservations.
2. The user and staff selects the reservation they wish to cancel from the list.
3. The user and staff confirms their intent to cancel the reservation by selecting the "Cancel" option.
4. The system updates reservation's status

**Alternative Sequence:**

If the system does not find any existing reservations associated with the user, it displays a message indicating that no reservations were found.

**Postcondition:**

The reservation is successfully canceled, and the user receives a cancellation confirmation.the reservation status is updated  to "Canceleled" in the system.

# API

## Reservation Reservation API

- GET /reservation
- PUT /reservation
- POST /reservation
- GET /reservation/{id}
- DELETE /reservation/{id}

## User User API

- GET /user
- PUT /user
- POST /user
- GET /user/{id}
- DELETE /user/{id}

## Restaurant Restaurant API

- GET /restaurant
- PUT /restaurant
- POST /restaurant
- GET /restaurant/{id}
- DELETE /restaurant/{id}

## Table Table API

- GET /tables
- PUT /tables
- POST /tables
- GET /tables/{id}
- DELETE /tables/{id}

# TRADE OFF

## Front End


## Back End



# TRADE OFF

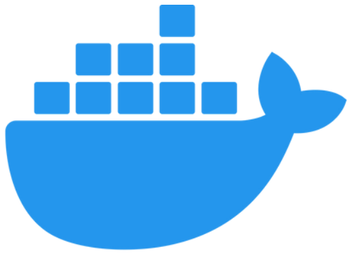
## DevOPs

NGINX®

GitLab

DigitalOcean

GitHub Actions



docker®

# THANKS FOR WATCHING