# Phase III: Software Design and Modeling

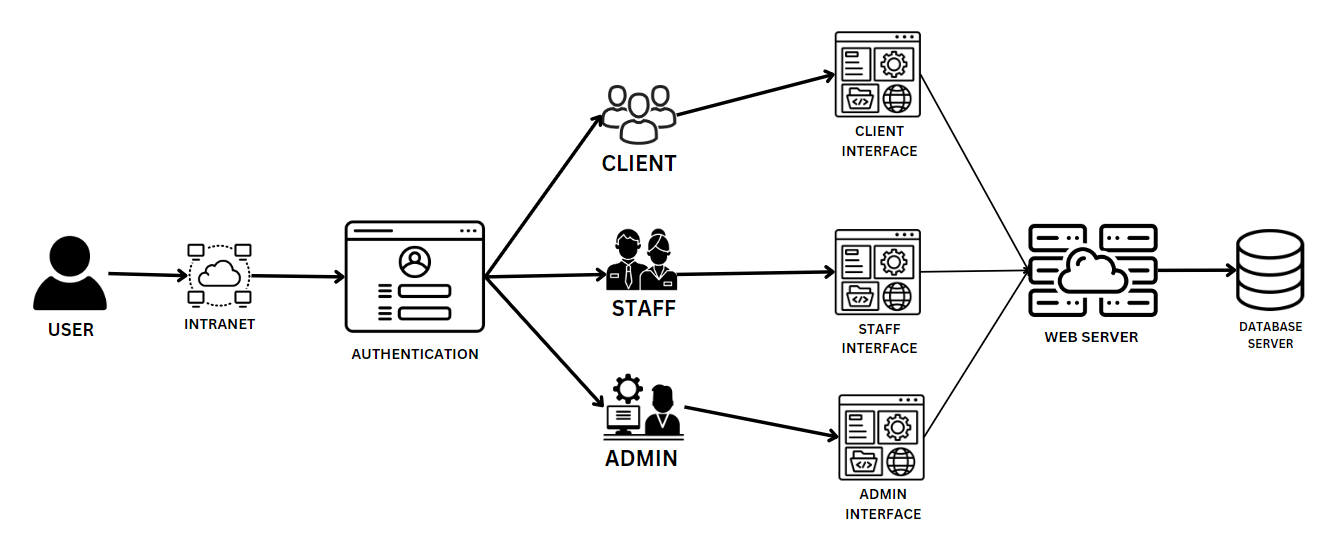
***Deadline: April 1st, 2024, 23:59***

*Dear Students,*

*In Phase III of our project, we will focus on software design and modeling. Below are the requirements for this phase, explained in an easy-to-understand format:*

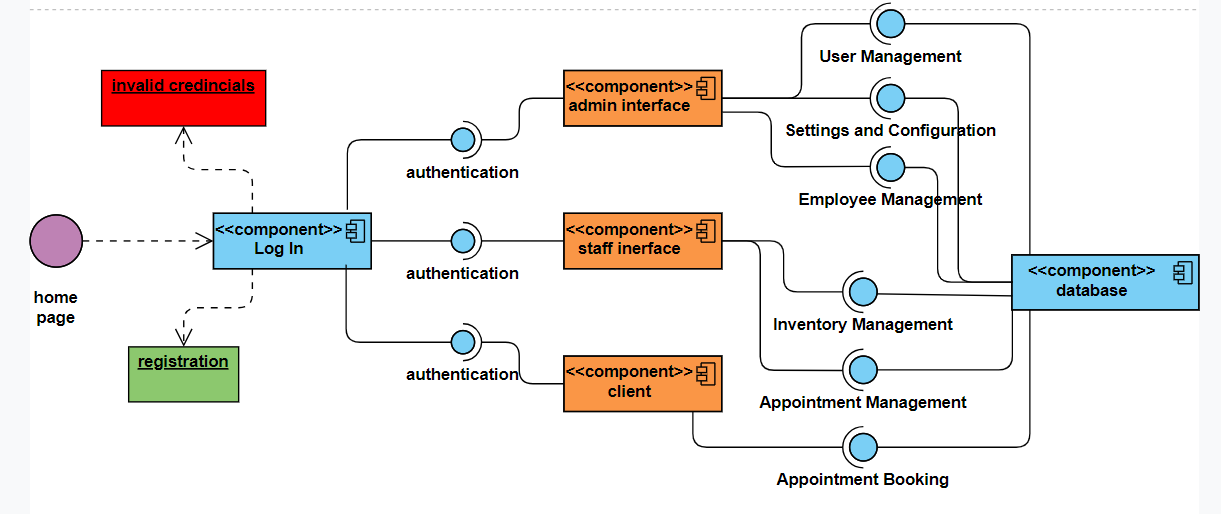
# Software Design and Modeling System Architecture:

*Explain how different parts of the system work together. Think of it as describing the big picture of your application - what it does and how it does it.*

**

# Component Diagram:

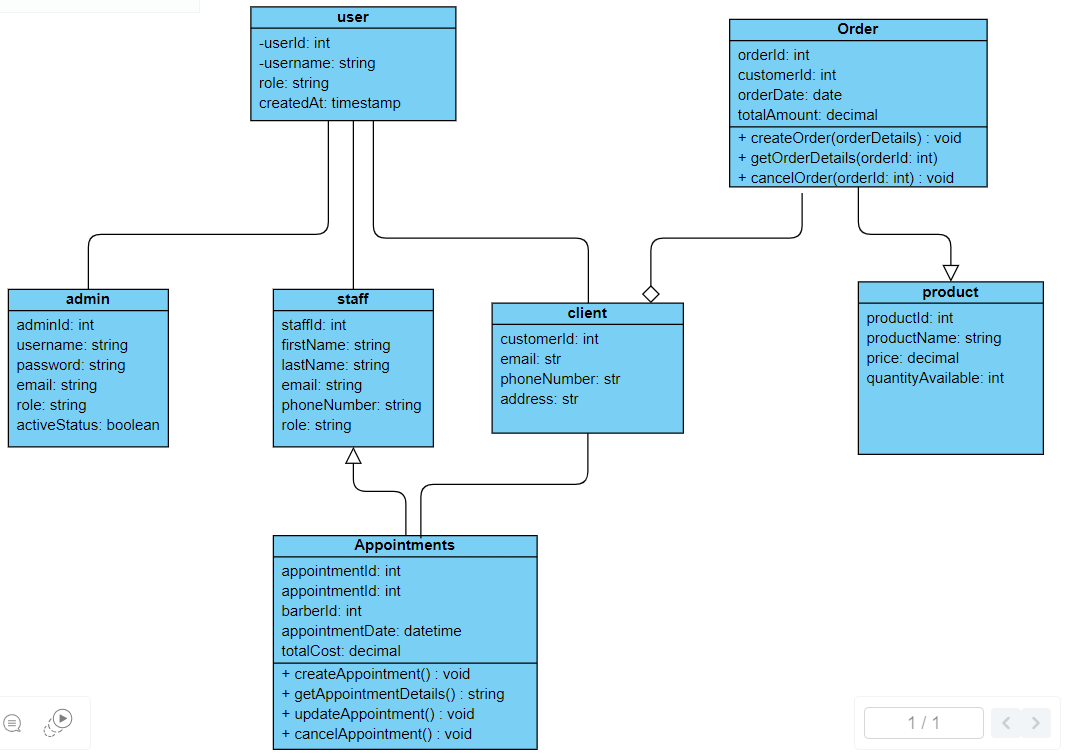
*Draw a picture showing the different parts (components) of your application and how they interact with each other. For example, if your application has a login feature, a component diagram would show how the login component talks to other parts of the system.*

**

# Detailed Design

**Class Diagram:**

*Think of a class diagram as a family tree for your application. It shows the different types of "things" in your application (called classes) and how they relate to each other. For example, if your application deals with cars, a class diagram would show that a Car class might have attributes like color and model, and methods like drive() and park().*



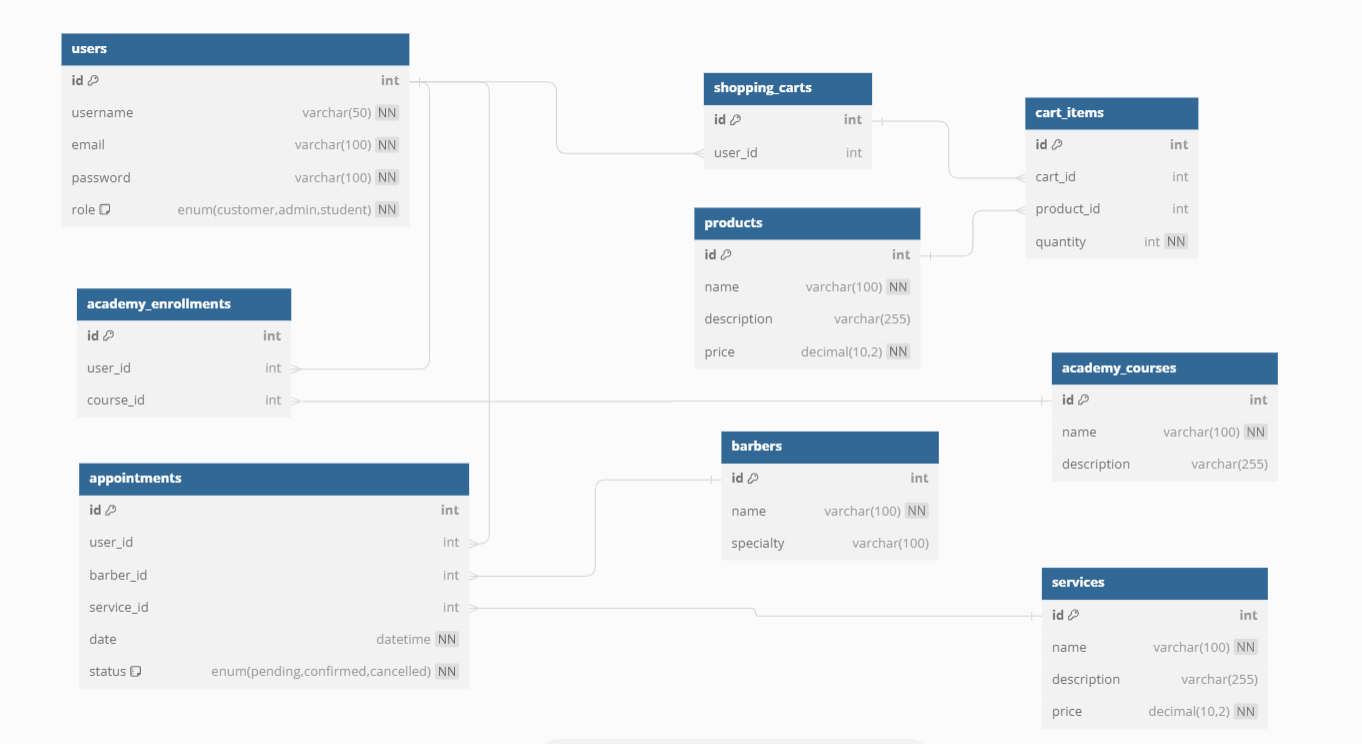
# Sequence Diagrams:

# Sequence diagrams illustrate the chronological flow of interactions within your application, akin to step-by-step guidelines depicting how various components communicate and collaborate. For instance, a sequence diagram for a barbershop booking system would outline actions like selecting a barber, scheduling an appointment, and confirming the booking.

# 

# Database Design:

*Explain how you've organized the data in your application. This includes things like what tables you have in your database, how they're related to each other, and how you've made sure your data is organized eficiently.*



1. *Users Table:*

*In the barber shop management system database, we have organized the data into several tables to efficiently manage various aspects of the business. Here's how the data is organized:*

*1.Users Table: This table stores information about users of the system, including customers and barbers. Each user has a unique ID, username, role, and creation timestamp. This table helps manage authentication and authorization within the system. 2. Customers Table: This table contains details about customers who visit the barber shop. It includes information such as customer ID, first name, last name, email, and phone number. This table allows the system to manage customer records and track their appointments and preferences.*

*3. Barbers Table: This table stores information about barbers employed at the barber shop. It includes details such as barber ID, full name, specialization, and experience level. This table helps manage barber records and track their availability and schedules. 4. Appointments Table: This table manages appointments scheduled by customers with barbers. It includes details such as appointment ID, customer ID, barber ID, appointment date, and total cost. This table facilitates appointment scheduling and helps track customer bookings and barber availability.*

*5. Services Table: This table contains information about services offered by the barber shop, such as haircut, shaving, styling, etc. It includes details such as service ID, name, description, duration, and price. This table helps manage service offerings and pricing within the system.*

*6.Inventory Table: If the barber shop sells products, this table manages inventory details. It includes information such as product ID, name, description, quantity available, and price. This table helps track inventory levels and manage stock for products sold at the shop.*

*7. Transactions Table: This table records transactions, including purchases made by customers and sales of products/services. It includes details such as transaction ID, customer ID, transaction type, amount, and timestamp. This table helps track financial transactions and monitor revenue generated by the barber shop.*

*Relationships between tables:*

*- Users Table is related to Customers Table and Barbers Table through foreign keys, allowing associations between users and their respective roles (customers or barbers). - Appointments Table is related to Customers Table and Barbers Table, as appointments are scheduled between customers and barbers.*

*- Services Table may be related to Appointments Table if services are associated with appointments (e.g., specific services booked during appointments).*

*- Inventory Table may be related to Transactions Table if products are sold, as transactions involve inventory updates.*

*- Transactions Table may be related to Customers Table if transactions are associated with customer purchases. Efficient organization:*

*- Tables are normalized to minimize data redundancy and improve data integrity. - Indexes are used to optimize query performance, especially for frequently accessed columns.*

*- Foreign key constraints are enforced to maintain data consistency and integrity. - Properly structured relationships between tables facilitate efficient data retrieval and manipulation operations.*

*- Regular database maintenance tasks, such as indexing, optimizing queries, and monitoring performance, ensure data organization remains efficient over time.*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), Username (type: string), password (type: string), role (type:strin)*

*- Relationship: One-to-One with Customers Table and Barbers Table.*

*2. Customers Table:*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), First name (type: string), Last name (type: string), Email (type: string), Phone number (type: string)*

*- Relationship: One-to-Many with Appointments Table, One-to-One with Users Table.*

*3. Barbers Table:*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), Full name (type: string), Specialization (type: string), Experience level (type: string)*

*- Relationship: One-to-Many with Appointments Table, One-to-One with Users Table.*

*4. Appointments Table:*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), Customer ID (type: integer, foreign key), Barber ID (type: integer, foreign key), Appointment date (type: datetime), Total cost (type: numeric)*

*- Relationship: Many-to-One with Customers Table and Barbers Table.*

*5. Services Table:*

*- Primary Key: ID*

*-Attributes: ID (type: integer, primary key), Name (type: string), Description (type: string), Duration (type: string), Price (type: numeric)*

*- Relationship: Many-to-Many with Appointments Table.*

*6.Inventory Table:*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), Product name (type: string), Description (type: string), Quantity available (type: integer), Price (type: numeric)*

*- Relationship: One-to-Many with Transactions Table.*

*7. Transactions Table:*

*- Primary Key: ID*

*- Attributes: ID (type: integer, primary key), Customer ID (type: integer, foreign key), Transaction type (type: string), Amount (type: numeric), Timestamp (type: datetime)*

*-Relationship: Many-to-One with Customers Table, One-to-Many with Inventory Table.*

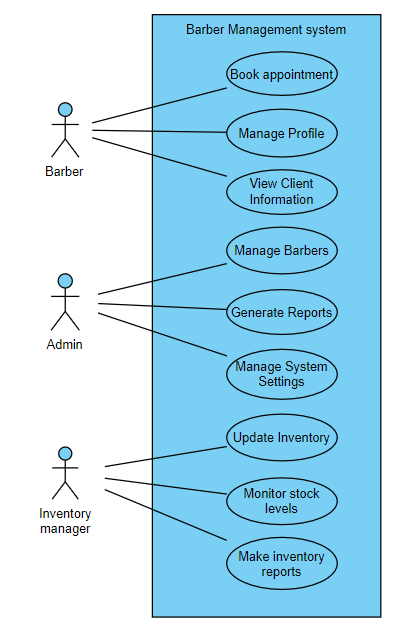
*These descriptions outline the structure and relationships of the tables within the barbershop management system database.*

# Modeling

Use Case Diagram:

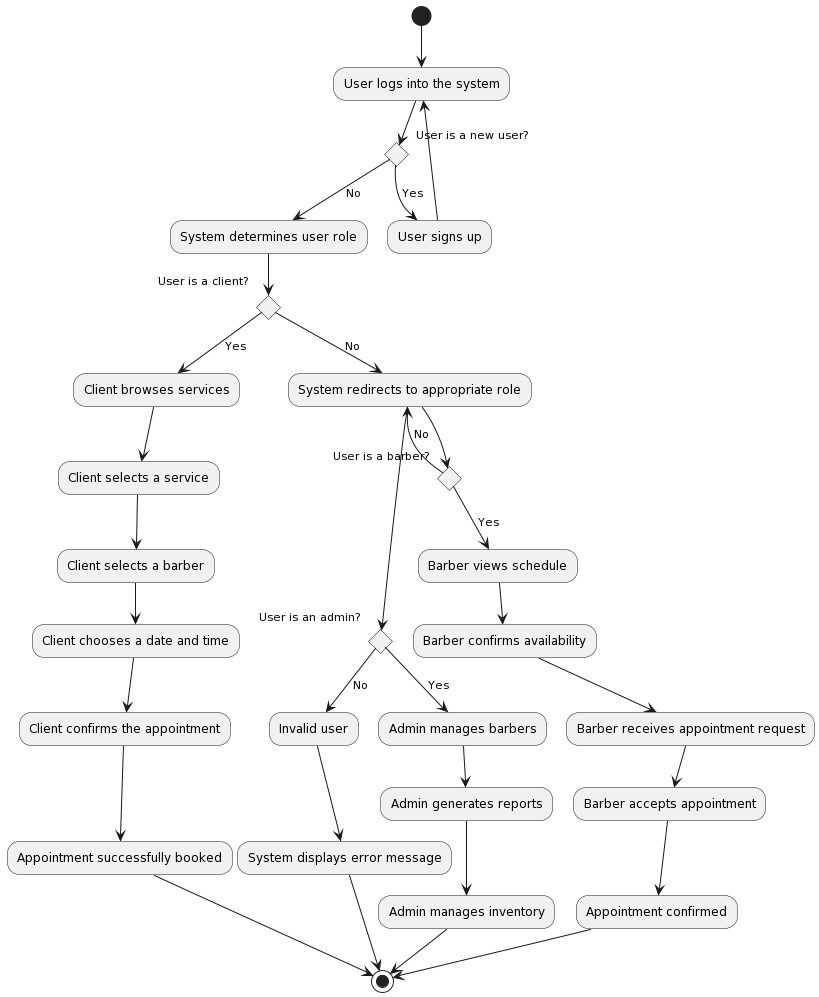
A use case diagram shows the different ways people (or other systems) can use your application. It's like a map of all the different things your application can do. For

example, a use case diagram for a music streaming app might show that users can search for songs, create playlists, and listen to music.



**Activity Diagrams:**

Activity diagrams show the ﬂow of activities in your application. They're like ﬂowcharts that show the steps involved in completing a task. For example, an activity diagram for booking a ﬂight might show the steps involved, like searching for ﬂights, selecting one and entering passenger information.



State Diagrams:

State diagrams show the different states that an object in your application can be in, and how it transitions between those states. They're like maps of all the possible "statuses" your application can be in. For example, a state diagram for a light switch might show that the switch can be in the "on" or "off" state, and how it transitions

between them.

