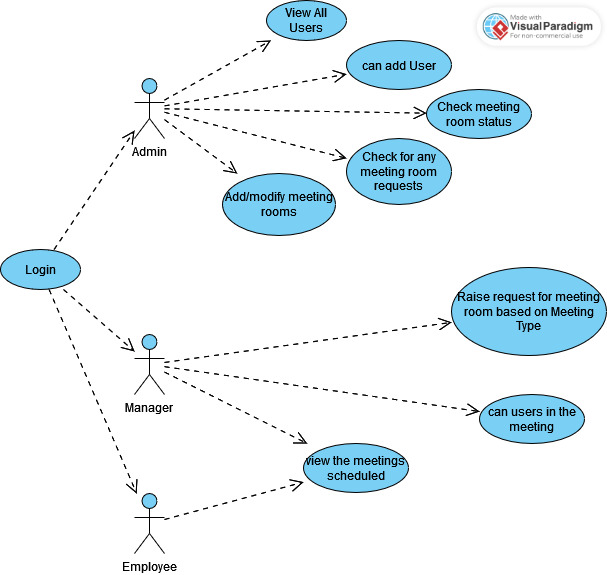
Automated Meeting Room Booking System

The Automated Meeting Room Booking System simplifies booking meeting rooms by letting users—Admins, Managers, and Employees—reserve rooms and manage schedules easily. The system provides a login interface for different roles, each with specific features like viewing room availability, making booking requests, and managing room details. It ensures a smooth booking experience by automating room management and handling user permissions efficiently.

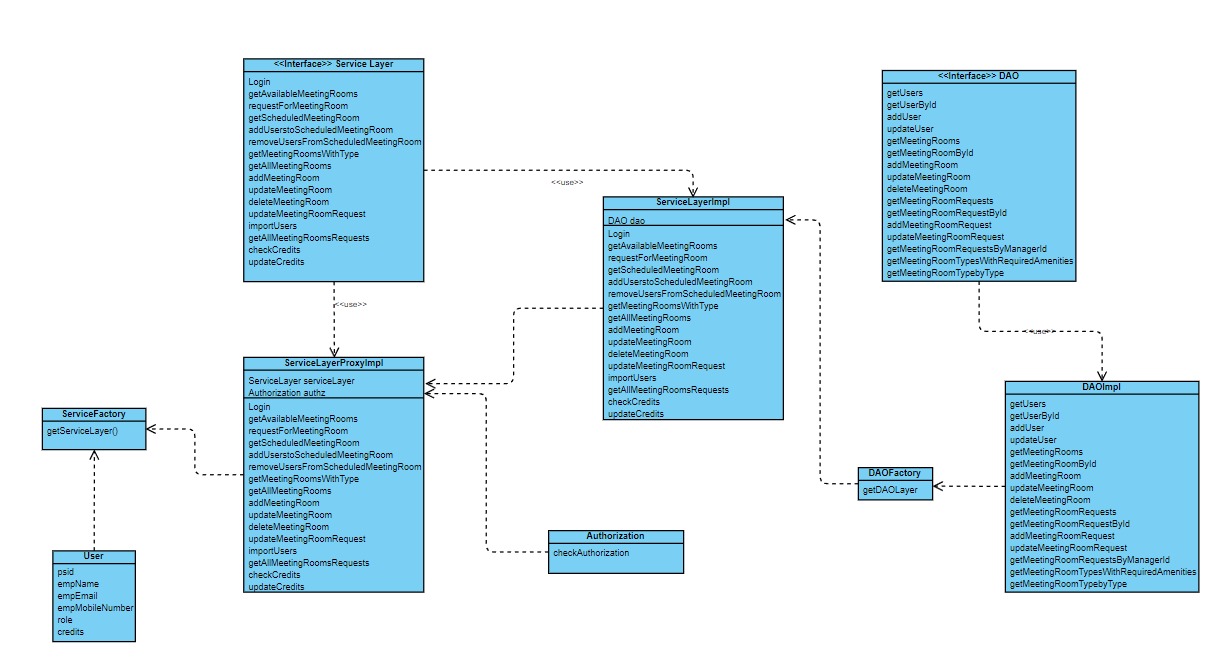
The Automated Meeting Room Booking System features an intuitive frontend designed for seamless interactions across different user roles—Admin, Manager, and Employee. It begins with a unified login screen where users input their credentials to access role-specific dashboards. The Admin dashboard allows comprehensive management, including viewing all users, adding new users, checking room availability, managing room requests, and modifying room details. Managers can create meeting requests, select amenities, add attendees, and view schedules with filters. Employees have a streamlined dashboard to view their assigned meetings with details like title, date, time, room, and approval status. The system's backend database efficiently supports these functionalities with well-structured tables. The \*\*Users\*\* table handles user information and roles, while \*\*Credentials\*\* manage passwords. \*\*RoomRequests\*\* tracks bookings and their statuses, and \*\*Attendees\*\* links users to bookings. \*\*Rooms\*\* details room information, \*\*Amenities\*\* lists available features, and \*\*RoomAmenitiesReference\*\* associates rooms with amenities. \*\*MeetingType\*\* and \*\*MeetingTypeRequiredAmenities\*\* define meeting types and their required amenities, ensuring effective room management and booking processes.

Project incorporates key design patterns to improve modularity and security. We’ve used the **Factory Pattern** in the DAO and Service layers to centralize object creation, allowing for easier dependency management and promoting loose coupling between components. Additionally, the **Proxy Pattern** is implemented between the UI and Service layers. When the UI layer requests access to the Service layer, a Service Proxy layer intervenes, handling authentication and authorization. Only upon successful validation is the actual Service object returned to the Proxy, ensuring secure and controlled access to critical functionalities.

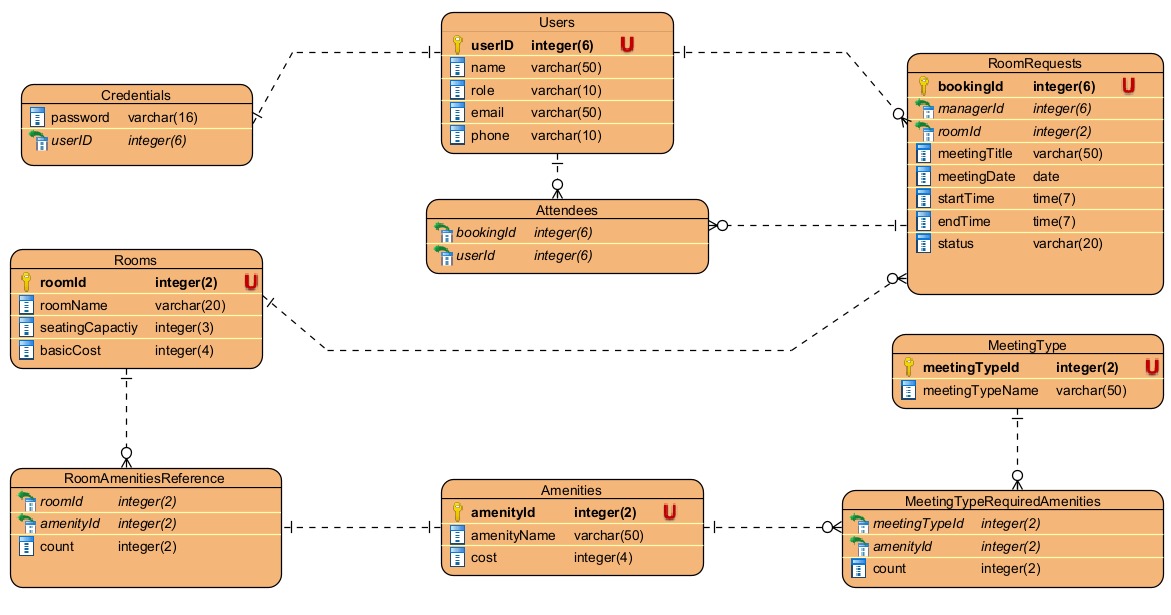
1. Use case Diagram



1. Class Diagram



1. ER diagram



1. Data Flow Diagram

