Photo Studio Administration System

1. Purpose

The Photo Studio Administration System aims to provide a centralized, easy-to-use platform for managing all aspects of a photography business, including studio bookings, customer management, package listings, staff schedules, and payment tracking. The system will streamline daily operations, improve customer service, and ensure accurate record-keeping.

Customers can browse available photography packages, check available slots, and make online bookings. Administrators can manage customer records, booking schedules, payments, package details, and promotional offers in real time.

The platform will be responsive and accessible across desktops, tablets, and smartphones, enabling both customers and staff to access it anywhere. Role-based authentication will ensure secure access to different modules for administrators, staff, and customers.

2. Scope

The Photo Studio Administration System will provide:

- Customer-Facing Features

* Online booking for photoshoots, events, and studio sessions.
* Viewing available packages with pricing and sample images.
* Checking real-time slot availability.
* Online payment integration.
* Email/SMS notifications for booking confirmations and reminders.

- Admin/Staff Features

* Add, update, or delete photography packages.
* Manage booking schedules, staff assignments, and customer data.
* Track payments, generate invoices, and apply discounts.
* Upload and manage sample images for portfolio display.
* Manage promotional offers and seasonal discounts.

- System Integration

  * Integration with payment gateways for secure transactions.
  * Email/SMS services for automated communication.
  * Secure REST APIs for frontend-backend communication.

3. Non-Functional Requirements

Performance

- Load booking and package listing pages within 2 seconds on standard broadband.
- Handle up to 500 concurrent users without significant performance loss.

Scalability

- Support growing numbers of packages, customers, and bookings.
- Horizontal and vertical scaling capability.

Security

- HTTPS/TLS encryption for all data transfers.
- Role-based authentication (Admin, Staff, Customer) with JWT tokens.
- Input validation to prevent SQL injection, XSS, and CSRF.

Usability

- Clean, intuitive navigation with a sidebar for module access.
- Consistent design with clear icons and labels for all pages.
- Responsive layout for desktop, tablet, and mobile devices.

Availability

- Minimum uptime: 99.5% annually.
- Automated backup of booking data every 24 hours.

Maintainability

- Modular code structure for easy updates to packages, booking rules, and UI.
- Clear documentation for backend API endpoints and frontend components.

4. System Architecture

Tech Stack:

- Frontend:

    React.js with modern UI libraries (e.g., Material UI / TailwindCSS).

- Backend:

    Spring Boot for REST APIs, business logic, and authentication.

- Database:

    MySQL for storing bookings, services, staff, and customer records.

5. Database Design

Entities & Attributes:

1. User
   - user_id (PK)
   - name
   - email
   - phone_number
   - password
   - role (Admin, Staff, Customer)
   - created_at

2. Service
   - service_id (PK)
   - service_name (Wedding, Portrait, Event, Editing, etc.)
   - description
   - price
   - duration
   - status (Active/Inactive)

3. Booking
   - booking_id (PK)
   - user_id (FK)

- service_id (FK)
- booking_date
- session_time
- status (Pending, Confirmed, Completed, Cancelled)
- payment_status (Paid/Unpaid)

4. Staff
- staff_id (PK)
- name
- role
- contact_info
- assigned_sessions

5. PhotoDelivery
- delivery_id (PK)
- booking_id (FK)
- delivery_link
- delivery_date
- status (Pending, Delivered)

6. Implementation Details

Backend (Spring Boot)

- Built using Spring Boot to handle all business logic, booking management, authentication, and secure API communication with the frontend.
- Uses Spring Data JPA with Hibernate ORM to interact with the MySQL database for efficient CRUD operations.
- Implements Spring Security with JWT (JSON Web Tokens) for role-based authentication (Admin, Staff, Customer) and secure API access.
- Provides RESTful endpoints for booking operations, service management, staff assignments, and photo delivery tracking.

Frontend (React.js)

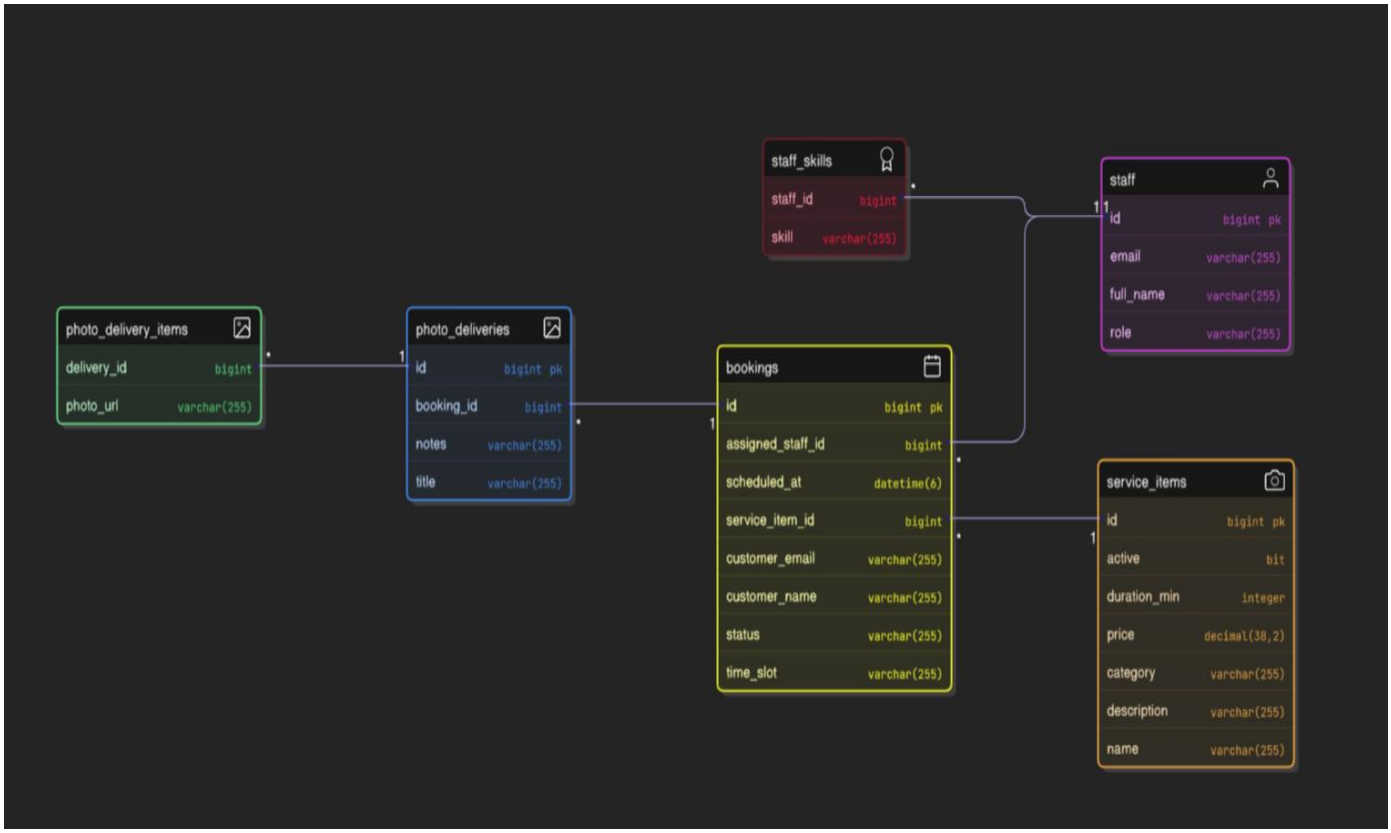
- Developed using React.js for a responsive and interactive user interface.
- Utilizes TailwindCSS for consistent UI design and responsive layouts.

- Integrates Axios for secure API requests to the backend.
- Includes features like booking calendars, package management, staff dashboards, and automated notifications.

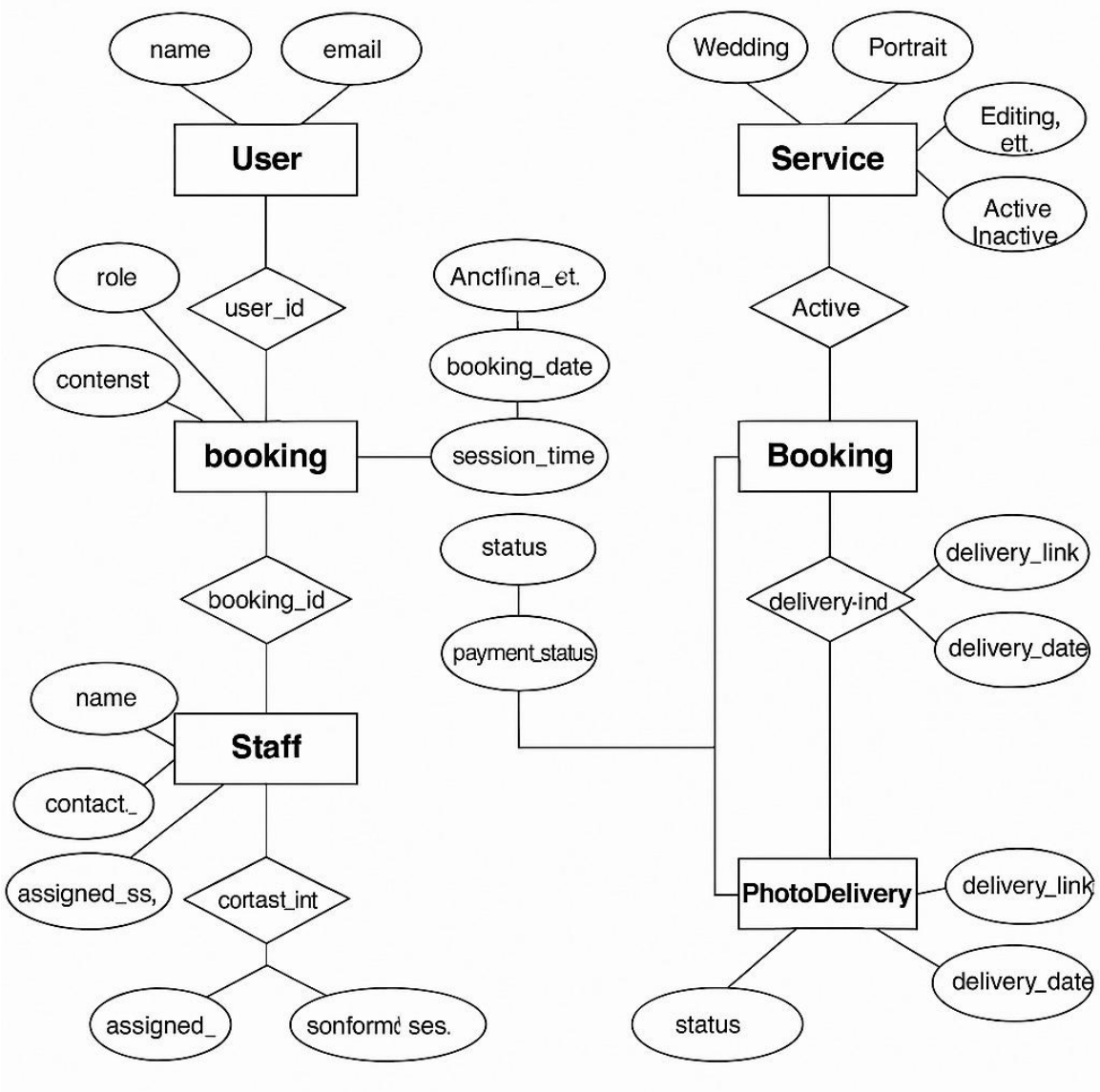
Database (MySQL)

- MySQL is used to store user accounts, services, bookings, staff, and photo delivery data.
- Database schema is normalized to avoid redundancy and ensure data consistency.
- Indexed search fields for fast retrieval of booking and service information.

7.Design Diagrams



UML Diagram



ER Diagram

8. Deployment

Backend Deployment:

- Local Deployment: Run the Spring Boot application using Maven/Gradle on Tomcat embedded server (port 8081 by default).
- MySQL database hosted locally or via Docker container.
- Cloud Deployment: Deploy backend to AWS EC2, Azure App Service, or similar.

- Use AWS RDS or Azure Database for MySQL for cloud-hosted database.
- Configure environment variables for database credentials, JWT secret, and API keys.

Frontend Deployment:

- Local Deployment: Run React app using npm run dev.
- Production Deployment: Build optimized files (npm run build) and deploy via Vercel, Netlify, or AWS S3 + CloudFront.
- Configure API base URLs to point to the deployed backend.

Database Deployment:

- Initialize MySQL schema using provided SQL scripts.
- Enable automated daily backups and failover support for high availability.

Additional Deployment Considerations:

- Enable HTTPS using SSL/TLS certificates for secure communication.
- Implement CI/CD pipelines (GitHub Actions, GitLab CI, Jenkins) for automated builds, tests, and deployments.
- Maintain version control via Git with structured branching strategy.

9. Glossary of Terms

Term / Acronym	Definition
CRUD	Create, Read, Update, Delete – basic database operations.
JWT	JSON Web Token – a secure, compact way of transmitting authentication data between client and server.
REST API	Representational State Transfer – API communication style that uses HTTP methods like GET, POST, PUT, DELETE.
REST	Representational State Transfer
RDS	Relational Database Service