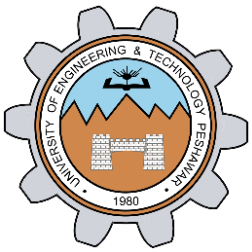
Project Milestone 2



Spring 2025

Group Members

Name	Registration Number
Hassan Zaib Jadoon	22PWCSE2144
Ahsan Raza	22PWCSE2099
Mutahhar Fayyaz	22PWCSE2176

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Summeyea Salahuddin

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Normalized Relational Schema - Event Booking System

Converted Relational Schema (from Conceptual Schema)

Based on the Entity Relationship Diagram (ERD), the conceptual schema has been converted into a relational schema and normalized up to **Third Normal Form (3NF)**. Each relation is designed to eliminate redundancy and maintain data integrity.

Relational Schema

1. **Admin**
   (admin_id, name, email, password, created_at)
   - Stores information about event administrators
   - Each admin can manage multiple events
2. **Event**
   (event_id, admin_id, title, description, date, time, venue, available_seats, price, created_at)
   - Represents individual events created by admins
   - admin_id is a foreign key referencing the Admin table
   - available_seats reflects remaining capacity
3. **Customer**
   (customer_id, name, email, phone)
   - Contains customer information for bookings
   - Supports booking without requiring full user registration
4. **Booking**
   (booking_id, event_id, customer_id, status, booking_date)
   - Links customers to events via bookings
   - event_id references the Event table
   - customer_id references the Customer table
   - status indicates current booking state (e.g., pending, confirmed

Normalization to 3NF

To ensure the database is efficient, consistent, and free from redundancy, the schema was normalized to the **Third Normal Form (3NF)**. Below is the step-by-step normalization process based on the initial structure of event and booking data.

Step 1: Unnormalized Form (UNF)

In the unnormalized form, multiple data points such as seat numbers and customer information were stored together in a single record, resulting in redundancy and multivalued fields.

Example: BookingDetails (event_id, event_title, event_date, event_time, venue, admin_name, admin_email, customer_name, customer_email, seat_number1, seat_number2, ..., status)

Step 2: First Normal Form (1NF)

In 1NF, all attributes must hold only **atomic** values, and repeating groups must be eliminated.

Revised Tables in 1NF:

- **users(user_id, name, email, password, role)**
- **events(event_id, title, date, time, venue, price, admin_id)**
- **seats(seat_id, event_id, seat_number, row, section, status, price)**
- **bookings(booking_id, event_id, user_id, status, total_amount, selected_seats)**

Note: The selected_seats field in bookings still violates 1NF as it stores multiple values in JSON format.

Step 3: Second Normal Form (2NF)

In 2NF, partial dependencies are removed. All non-key attributes must depend on the entire primary key. A new relation is introduced to handle the many-to-many relationship between bookings and seats.

Revised Tables in 2NF:

- **users(id, name, email, password, role)**
- **events(id, title, description, date, time, venue, price, capacity, user_id)**
- **seats(id, event_id, seat_number, row, section, status, price)**
- **bookings(id, event_id, user_id, ticket_number, qr_code, status, total_amount, booking_date)**
- **booking_seats(booking_id, seat_id)**

The booking_seats table replaces the JSON-based selected_seats field to ensure atomic values and proper foreign key relationships.

Step 4: Third Normal Form (3NF)

In 3NF, transitive dependencies are removed. All non-prime attributes must depend only on the primary key.

Final Relations in 3NF

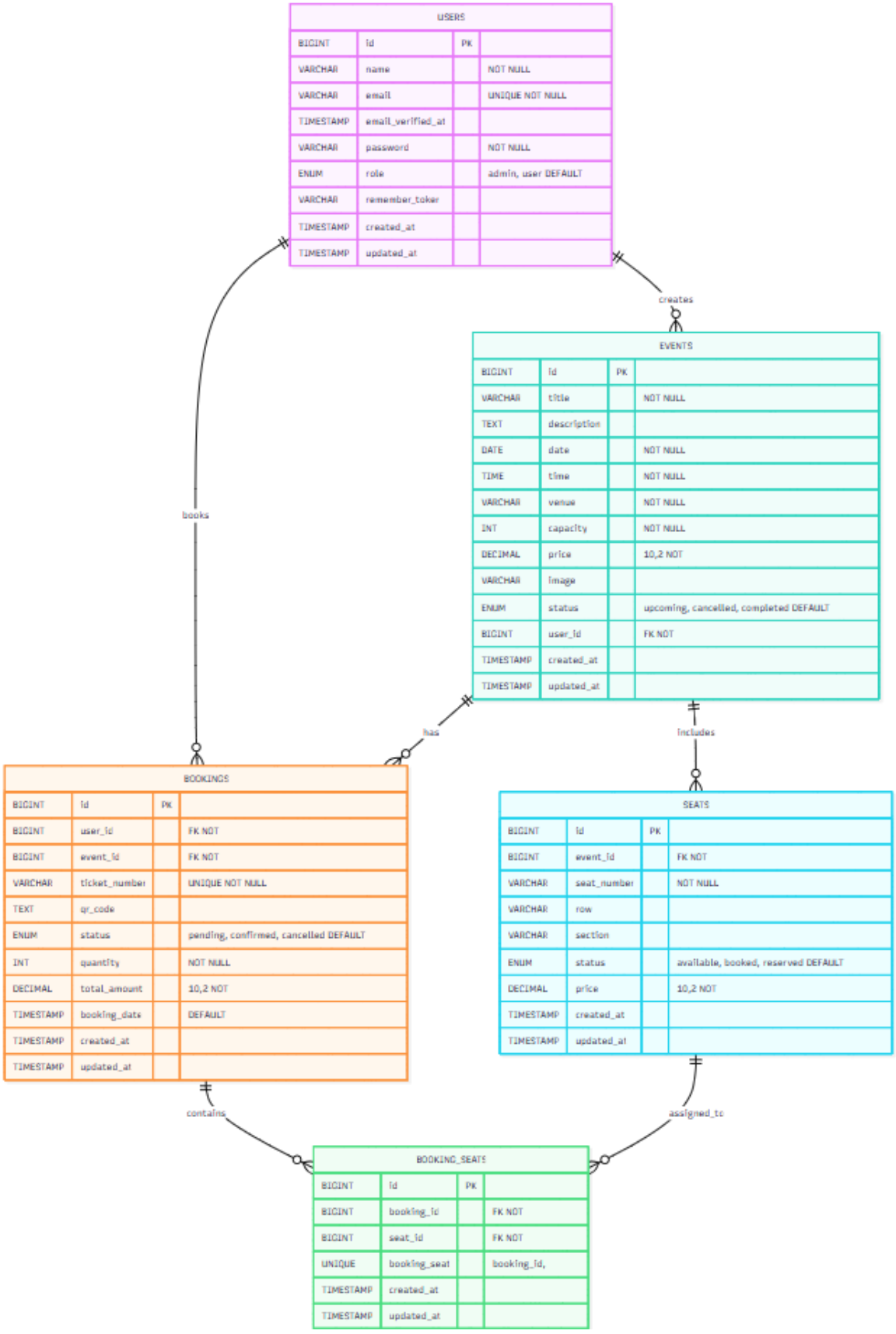
1. **users**
   (id, name, email, password, role, email_verified_at, remember_token, created_at, updated_at)
2. **events**
   (id, title, description, date, time, venue, capacity, price, image, status, user_id, created_at, updated_at)
3. **seats**
   (id, event_id, seat_number, row, section, status, price, created_at, updated_at)
4. **bookings**
   (id, user_id, event_id, ticket_number, qr_code, status, quantity, total_amount, booking_date, created_at, updated_at)

5. **booking_seats** *(new table added for full normalization)*
   (booking_id, seat_id)

The database design has been successfully normalized to **Third Normal Form (3NF)** to ensure:

- Elimination of redundancy
- Improved data consistency and integrity
- Scalability and flexibility in querying booking and seating data

Figure:


USERS
BIGINT id PK
VARCHAR name NOT NULL
VARCHAR email UNIQUE NOT NULL
TIMESTAMP email_verified_at
VARCHAR password NOT NULL
ENUM role admin, user DEFAULT
VARCHAR remember_token
TIMESTAMP created_at
TIMESTAMP updated_at
creates
books
EVENTS
BIGINT id PK
VARCHAR title NOT NULL
TEXT description
DATE date NOT NULL
TIME time NOT NULL
VARCHAR venue NOT NULL
INT capacity NOT NULL
DECIMAL price 10,2 NOT
VARCHAR image
ENUM status upcoming, cancelled, completed DEFAULT
BIGINT user_id FK NOT
TIMESTAMP created_at
TIMESTAMP updated_at
has
includes
BOOKINGS
BIGINT id PK
BIGINT user_id FK NOT
BIGINT event_id FK NOT
VARCHAR ticket_number UNIQUE NOT NULL
TEXT qr_code
ENUM status pending, confirmed, cancelled DEFAULT
INT quantity NOT NULL
DECIMAL total_amount 10,2 NOT
TIMESTAMP booking_date DEFAULT
TIMESTAMP created_at
TIMESTAMP updated_at
SEATS
BIGINT id PK
BIGINT event_id FK NOT
VARCHAR seat_number NOT NULL
VARCHAR row
VARCHAR section
ENUM status available, booked, reserved DEFAULT
DECIMAL price 10,2 NOT
TIMESTAMP created_at
TIMESTAMP updated_at
contains
assigned_to
BOOKING_SEATS
BIGINT id PK
BIGINT booking_id FK NOT
BIGINT seat_id FK NOT
UNIQUE booking_seat booking_id,
TIMESTAMP created_at
TIMESTAMP updated_at