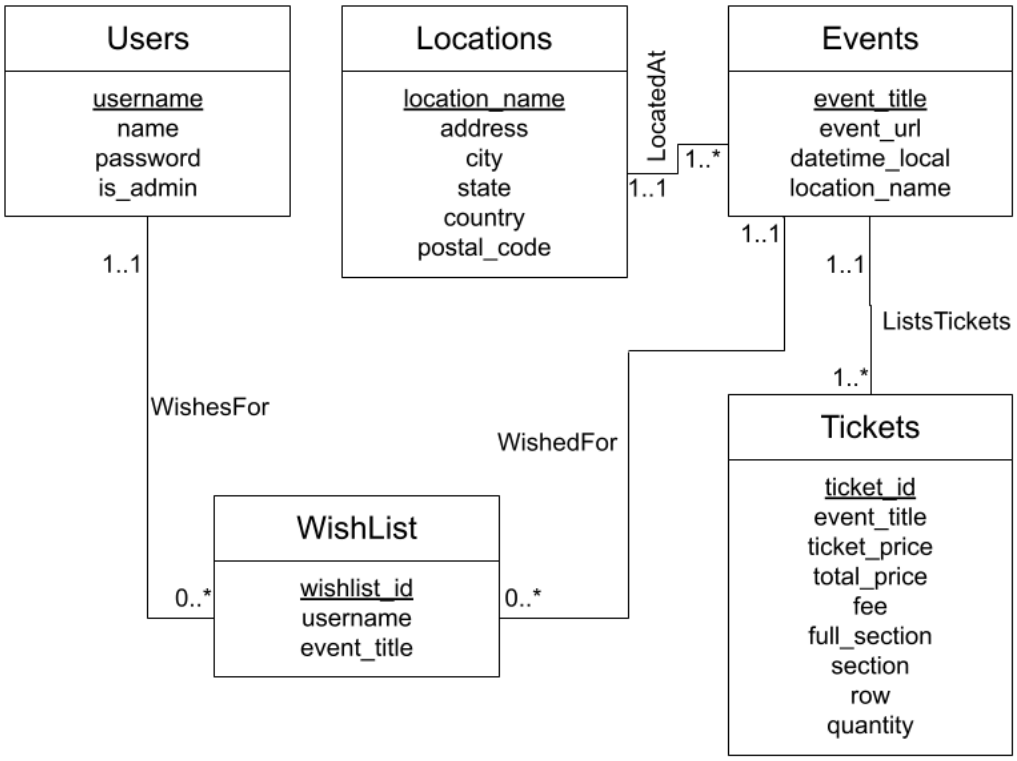

1. **UML Diagram**

2. **Explanation**

Entities:

1. User:
   Users is an entity because it captures user-specific attributes like name, username and password.
2. Events:
   Events is modeled as an entity because an event contains multiple attributes related to its nature (title, time, location, and tickets).
3. Locations:
   Locations is an entity to model different places where events are held, and these are independent of other entities.
4. Tickets:
   Tickets is treated as a separate entity because tickets are sold for different events and have detailed attributes that would make this entity large if stored as part of another table. A ticket has multiple attributes like section, row, fee, quantity, etc.
5. WishList:

Wishlist is modeled as an entity to link users and events, which would otherwise require storing lists within either Users or Events, violating normalization rules.

Explanation of cardinality:

- Users to Wishlist:
  A user can add 0 or many events to their wishlist. This is a **one-to-many relationship**. A wishlist_id is unique to one user_id. This assumption is made based on customer requirements as it depends on the user to wishlist 0 or multiple events.

- Events to Wishlist:
  An event can be associated with many users' wishlist. Additionally, the same user can have multiple events in his wishlist. Thus, the cardinality between events and wishlist is that a single wishlist can have 0 or many events and an event can occur in 1 wishlist. This is a **one-to-many relationship**. This assumption is made based on application constraints, as we allow the same event to be in multiple wishlists.

- Events to Tickets:
  An event can have multiple tickets associated with it. An event can have either 1 or many tickets, and one ticket is related to one event. Thus, we have a **one-to-many relationship** between them. This assumption is made based on application constraints where we assign one event_id to multiple tickets.

- Events to Location:
  An event can have only one location. The same location can be associated with 1 or more events. Therefore, we have a **one-to-many relationship.** This assumption is made based on application constraints where one location can host many events.

- Users to Events:
  Many users can be connected to several different events. Moreover, a single event can be connected to many different users. Therefore we have a **many-to-many relationship** between the events and users.

3. UML Requirements

1. Our 5 entities are Users, Locations, Events, Tickets, and WishList
2. We have a many-to-many relationship between Users and Events through the WishList table because many users wish for many events, and many events are wished for by many users. We have a 1-to-many relationship between Locations and Events because one location may host many events, but each event is hosted at one location.

4. Normalization Proof:

Functional Dependencies:

- username → name, password, is_admin
- location_name → address, city, state, country, postal_code
- event_title → event_url, datetime_local, location_name
- ticket_id → event_title, ticket_price, total_price, fee, full_section, section, row, quantity
- event_title, username → wishlist_id

Left Side Only	Middle	Right Side Only	None
username ticket_id	location_name event_title	name, password, is_admin, address, city, state, country, postal_code, event_url, datetime_local, ticket_price, total_price, fee, full_section, section, row, quantity, wishlist_id	

username, ticket_id → event_title, location_name, … (RHS), so it is a minimal superkey. Therefore, no singleton LHS key is a superkey.

<u>BCNF Proof for Each Relation</u>:

Users

R(username, password, is_admin)
FD: username → password, is_admin
username+: {username, password, is_admin} is R, so username is a superkey.
No violating FD's, so BCNF

Locations

R(location_name, address, city, state, country, postal_code)

FD: location_name → address, city, state, country, postal_code
location_name+: {location_name, address, city, state, country, postal_code} is R, so location_name is a superkey.
No violating FD's, so BCNF

Events

R(event_title, event_url, datetime_local, location_name)
FD: event_title → event_url, datetime_local, location_name
event_title+: {event_title, event_url, datetime_local, location_name} is R, so event_title is a superkey.
No violating FD's, so BCNF

Tickets

R(ticket_id, event_title, ticket_price, total_price, fee, full_section, section, row, quantity)
FD: ticket_id → event_title, ticket_price, total_price, fee, full_section, section, row, quantity
ticket_id+: {ticket_id, event_title, ticket_price, total_price, fee, full_section, section, row, quantity} is R, so ticket_id is a superkey.
No violating FD's, so BCNF

WishList

R(wishlist_id, event_title, username)
FD: event_title, username → wishlist_id
event_title, username +: {event_title, username, wishlist_id}, so event_title, username is a superkey.
No violating FD's, so BCNF

5. Logical Design:

Users(
username:VARCHAR(100) [PK],
name:VARCHAR(100),
password:VARCHAR(100),
is_admin:BOOLEAN
)

Events(
event_title:VARCHAR(250) [PK],
event_url:VARCHAR(250),
datetime_local:VARCHAR(100),

location_name:VARCHAR(250) [FK to Locations]
)
Locations(
location_name:VARCHAR(250) [PK],
address:VARCHAR(250),
city:VARCHAR(100),
state:VARCHAR(100),
country:VARCHAR(100),
postal_code:VARCHAR(10)
)
Tickets(
ticket_id:INT [PK],
event_title:VARCHAR(250) [FK to Events],
ticket_price:FLOAT(2),
total_price:FLOAT(2),
fee:FLOAT(2),
full_section:VARCHAR(100),
section:VARCHAR(100),
row:VARCHAR(100),
quantity:INT
)
WishList(
wishlist_id:INT [PK],
user_name:INT [FK to Users],
event_title:VARCHAR(250) [FK to Events],
)