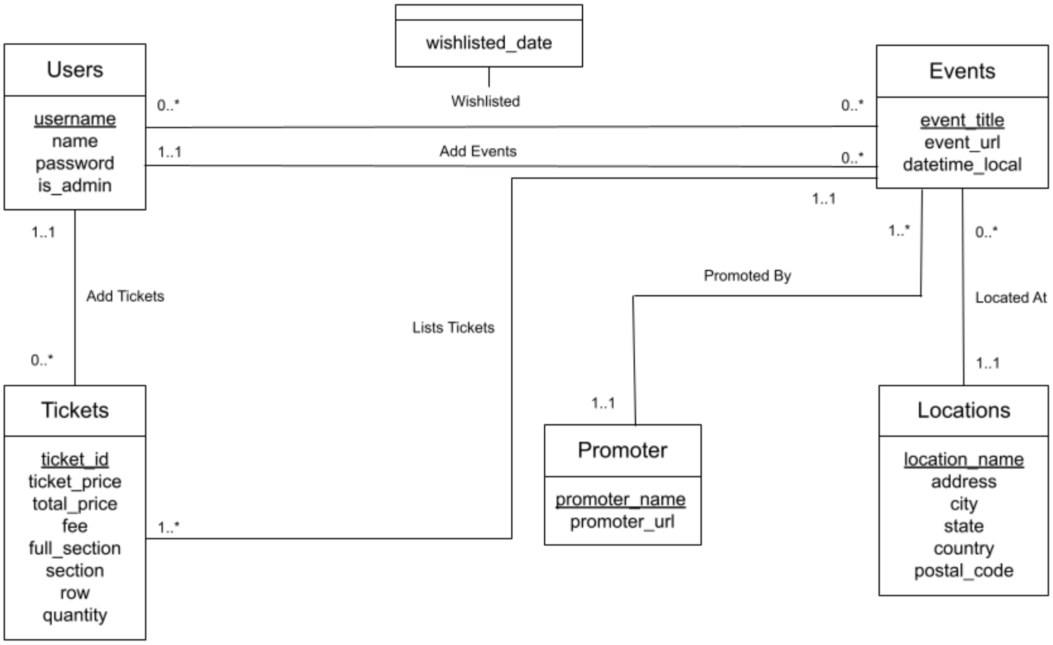

1. **UML Diagram**

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. <u>Promoter:</u>
   Promoter is modeled as an entity, which indicates the actual platform or seller selling the tickets. It contains attributes such as platform name and website. A promoter may or may not be related to a particular event, so it can be considered as an entity.

   Relation Tables:
1. Wishlist:
   There is a many-to-many relationship between users and events. Thus, we create a relation table which can be uniquely identified by username and event_title.

Explanation of cardinality:

- <u>Users to Events:</u>
  A user can add 0 or many events to their wishlist. This is a **many-to-many relationship**. An event can be wishlisted by 0 or many users. This assumption is made based on customer requirements as it depends on the user to wishlist 0 or multiple events.
  Additionally, admin users can add new events. Admin user can add 0 or more events and an event is associated with at most one admin user. (Some events were added automatically.) Thus, this is a **one-to-many relationship** based on application constraints.

- <u>Users to Tickets:</u>
  Additionally, admin users can add new tickets to the data. An admin user can add 0 or more tickets to the table and a ticket is associated with at most one admin user. (Some tickets were added automatically.) Thus, this is a **one-to-many relationship** based on application constraints.

- <u>Events to Tickets:</u>
  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.

- <u>Events to Location:</u>

An event can have only one location. The same location can be associated with 0 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.

- Events to Promoters:
  A promoter can sell tickets for 1 or more events. An event is associated with exactly one promoter. Some examples for promoters are StubHub, ETickets, Sports Illustrated Tickets, Tickets.com, FeverUp, Vivid Seats. Thus, it is an application based constraint showing **many-to-one relationship**.

3. UML Requirements

1. Our 5 entities are Users, Locations, Events, Tickets, and Promoter
2. We have a many-to-many relationship between Users and Events through the WishList relation 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, promoter_name, username
- ticket_id → event_title, ticket_price, total_price, fee, full_section, section, row, quantity, username
- promoter_name → promoter_url
- username, event_title → wishlist_date

Left Side Only	Middle	Right Side Only	None

username ticket_id	location_name event_title promoter_name	name, password, is_admin, address, city, state, country, postal_code, event_url, datetime_local, ticket_price, wishlist_date, total_price, fee, full_section, section, row, quantity, promoter_url	

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

BCNF Proof for Each Relation:

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

Promoter

R(promoter_name, promoter_url)
FD: promoter_name → promoter_url

promoter_name +: {promoter_url}, so promoter_name is a superkey.
No violating FD’s, so BCNF

Wishlist

R(event_title, username, wishlist_date)
FD: event_title, username → wishlist_date
event_title, username +: {event_title, username, wishlist_date}, 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],
        promoter_name:VARCHAR(250) [FK to Promoter],
        username:VARCHAR(100) [FK to Users]
        )
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,
        username:VARCHAR(100) [FK to Users]
        )
Promoter(
         promoter_name:VARCHAR(250) [PK]
         promoter_url:VARCHAR(250)
         )
WishList(
        user_name:VARCHAR(100) [FK to Users][PK],
        event_title:VARCHAR(250) [FK to Events][PK],
        wishlist_date:VARCHAR(100)
        )
```