Report 2

for

Note-Bridge 5

Prepared by Eren Altın, Ahmet Gungor, Marin Nofulla, Sabin Nicolae, Ahmet Hacioglu

07-06-2024

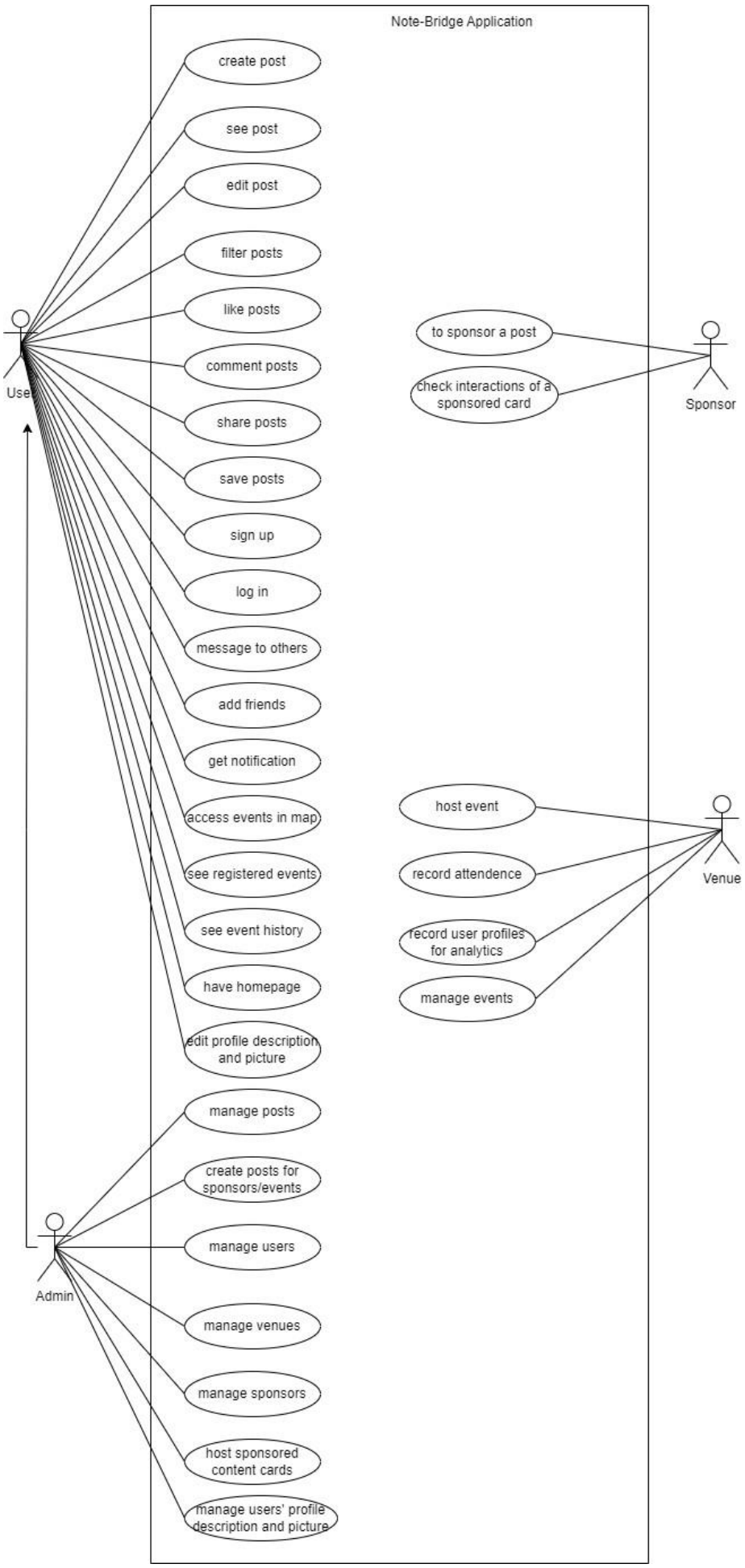
Contents

Use Case Diagram 3

Class Diagram 5

Database Schema 7

Use Case Diagram



The above diagram is the Use Case Diagram for our project. It includes 4 different actors: User, Admin, Sponsor, and Venue with each of them being capable of doing different functionalities.

Users:

The user actor represents the application's broad audience or members. Users may access a wide range of capabilities, with an emphasis on engagement and interaction with information. They can produce, view, modify, and filter postings, which are essential components of any social or content-driven site. In addition to these features, users may interact with postings by like, commenting, sharing, and storing them, increasing user engagement and information distribution. Furthermore, users may sign up and log in, allowing for more customized experiences. They may engage with other users through chat and add friends to their social network within the program. Users also receive alerts, maybe about new posts, comments, or friend activity, which increases their engagement with the app. Users may also access, observe, and evaluate events on a map, browse events for which they have registered, and review their event history, integrating social event features into the program. Users also have a homepage, which is most likely used as a dashboard or starting point for their app activity.

Admin:

Admins are a sub-type of user but has extensive control over the application's content and structure. Admins can regulate postings by evaluating, approving, or deleting them to guarantee content quality and conformity to the platform's guidelines. They also make postings for sponsors or events, either as a service to sponsors or to promote specific events. Admins administer users, which often include managing user accounts, responding to allegations of misconduct, and verifying user compliance with platform standards. They also manage venues and sponsors, which involves adding, updating, and deleting profiles to ensure correct and proper representation in the application. Finally, administrators host sponsored content cards, which entail maintaining and presenting sponsored material in a way that matches with both user engagement goals and the sponsor.

Sponsor:

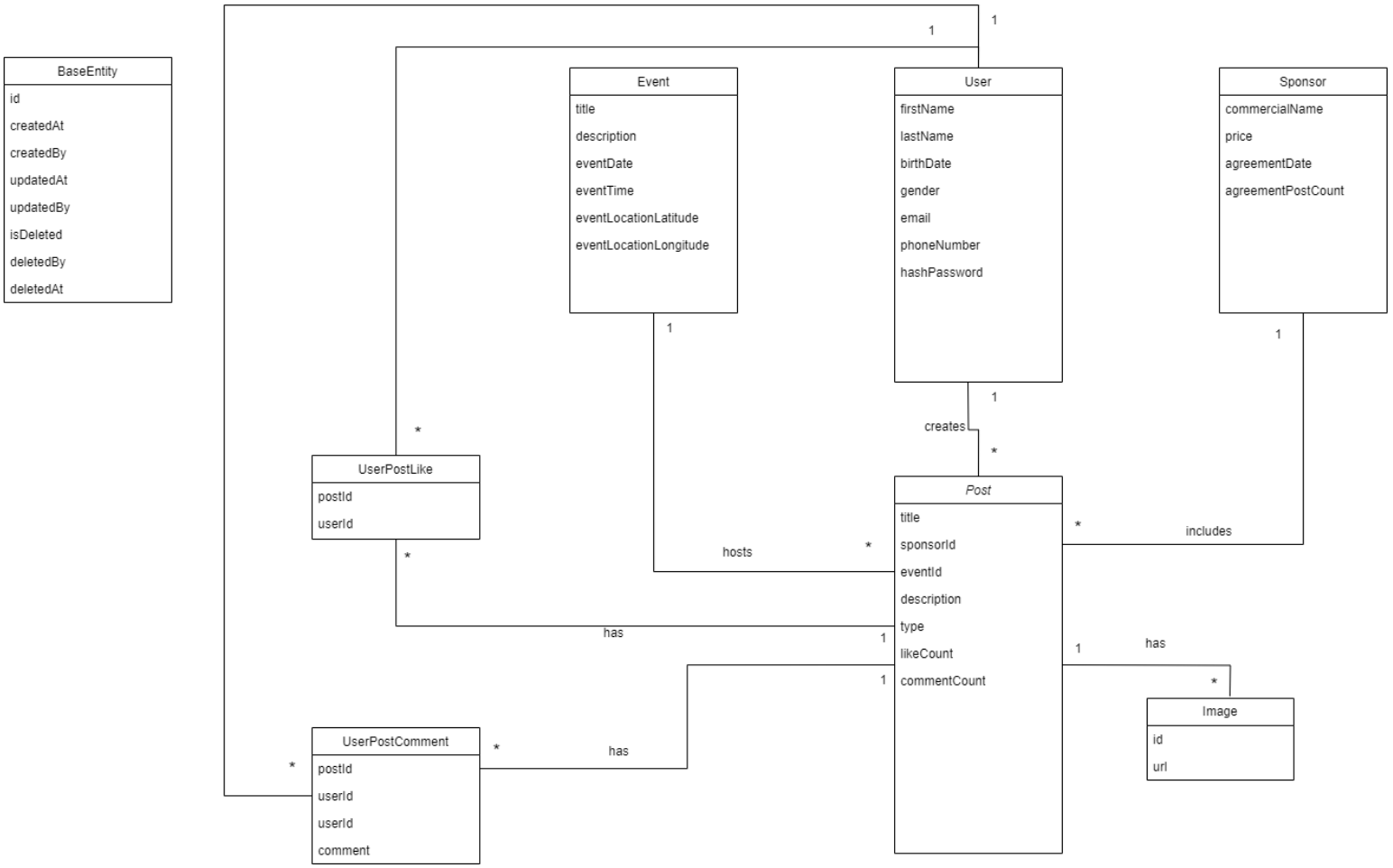
Sponsor actor focuses primarily on marketing and engagement analytics. Sponsors may sponsor posts to promote their products, services or events on the App. They can also monitor sponsored card engagement, which allows them to track how users interact with their sponsored content. This feature is important for sponsors to evaluate the effectiveness of their advertising or promotional strategies on the platform.

Venue:

The Venue actor's skills are focused on event management inside the application. Venues can host events, which often entails giving information on event locations, dates, and specifics. They can also track attendance at these events, which is useful for controlling event capacity and identifying participant demographics. Furthermore, venues may collect user profiles for analytics, giving them insights into the sorts of people that attend

their events, which can help them plan future events and sell more effectively. Venues also manage events, which may involve updating event details, replying to user queries about events, and perhaps managing logistics via the application.

Class Diagram



The above class diagram illustrates the connection between different classes in our application. The multiplicities and verbs showcase each class's functionality and relation to the others.

BaseEntity

- Attributes: id, createdAt, createdBy, updatedAt, updatedBy, isDeleted, deletedBy, deletedAt

- Purpose: Serves as a base class for other entities, providing common attributes for auditing and status tracking.

User

-Attributes: firstName, lastName, birthDate, gender, email, phoneNumber, hashPassword

-Relationships: Creates multiple Posts, has many UserPostLikes and UserPostComments.

Event

-Attributes: title, description, eventDate, eventTime, eventLocationLatitude, eventLocationLongitude

-Relationships: Each event is linked to multiple Posts.

Sponsor

- Attributes: commercialName, price, agreementDate, agreementPostCount

- Relationships: Can sponsor multiple Posts.

Post

-Attributes: title, sponsorId, eventId, description, type, likeCount, commentCount

-Relationships: Created by one User, may be sponsored by one Sponsor, linked to one Event, can include multiple Images, and has many UserPostLikes and UserPostComments.

UserPostLike

-Attributes: postId, userId

-Purpose: Represents the many-to-many relationship between User and Post for likes.

UserPostComment

-Attributes: postId, userId, comment

-Purpose: Represents comments made by users on posts, creating a many-to-many relationship with additional data (the comment itself).

Image

- Attributes: id, url

- Relationships: Many images can be in a post.

Database Schema

```
User(userId, firstName, lastName, birthDate, gender, email, phoneNumber, hashPassword,
    PK(userId))

Sponsor(sponsorId, commercialName, price, agreementDate, agreementPostCount,
      PK(sponsorId))

Event(eventId, title, description, eventDate, eventTime, eventLocationLatitude,
eventLocationLongitude,
     PK(eventId))

Post(postId, baseEntityId, title, sponsorId, eventId, userId, description, type, likeCount,
commentCount,
    PK(postId),
    FK(baseEntityId) REFERENCES BaseEntity(baseEntityId),
    FK(sponsorId) REFERENCES Sponsor(sponsorId),
    FK(eventId) REFERENCES Event(eventId),
    FK(userId) REFERENCES User(userId))

Image(imageId, url, postId,
     PK(imageId),
     FK(postId) REFERENCES Post(postId))

BaseEntity(baseEntityId, createdAt, createdBy, updatedAt, updatedBy, isDeleted, deletedBy,
deletedAt,
        PK(baseEntityId))

UserPostLike(postId, userId,
        PK(postId, userId),
        FK(postId) REF Post
        FK(userId) REF User))

UserPostComment(postId, userId, comment,
        PK(postId, userId),
        FK(postId) REF Post
        FK(userId) REF User))
```

The schema specifies the layout of a relational database intended to serve a social platform, with a focus on user interactions, content management, and event coordination. The database consists of several tables, including User, Sponsor, Event, Post, Image, and BaseEntity. Each table is intended to record a certain sort of data: user profiles, sponsor details, event features, and post dynamics. These tables are linked together via main and foreign keys, allowing the system to manage complicated connections such as postings associated with certain users, events, and sponsors, as well as track user interactions such as comments and likes on posts. In addition, the BaseEntity table has information elements such as creation and update timestamps and deletion flags to enable record management standards across the platform.