Pet Network

I. Content

I. Project Description

A. Overview

My project will be implementing a "Pet Network" that can used by the owner. It's a database-driven platform that allows pet owners to connect, share experiences, look for help and support each other. The system can combine elements of a social network, such as profiles, posts, and messaging, with pet-specific features, like pet profiles, veterinary info, adoption, and events. The database will probably serve as the foundation for user data, pet data, social interactions, and community activities.

B. What this project does

It can create a platform for people who owns pet to:

- Register the pets and owner themselves, also they can share about experience, health info, photos and updates
- Discover local pet events or petsitter
- Access resources like vets, traning services, and adoption centers
- Connect with other pet owners through groups, messaging, and events (one-to-one, one-to-more, and more-to-one).

C. Why is it needed

Nowadays, more and more people choose to have pets instead of children, and existing platforms like Facebook or Instagram are too general for the unique needs of pet communities. Rover’s features are less comprehensive. Therefore, it's a need to make a platform that can let pet owners got a dedicated, organized way that can connect socially while also accessing trusted resources about pet care.

D. What Problem does it solve

There are three main problems that can be solve:

First, fragmented pet communities, currently, pet owners use scattered forums or general social media. The project consolidates pet related networking.

Second, access to pet care information. It can provides structured info on veterinary services, adoption opportunities, and training programs.

Third, the owners can find supports, ask questions, and share tips with individuals.

E. Who does this benefit

5 types of people could have benefit. First, pet owners, the main beneficiaries who use the platform socially and informationally.

Second, veterinarians and trainers. They can gain visibility and interact with clients. Third, adoption centers, they can share pets available for adoption. Also, I think the most database will be used. Forth, event organizers, they could promote local pet-friendly events. Last but not least, pet product sellers and service providers could receive more clients because their target costumer are using the platform.

II. List of Use Cases

A. List of Actors

- Pet Owner

  Creates a profile, register pets, joins groups, and interacts socially.

- Veterinarian

  Maintains a professional profile, shares expertise, and connects with clients.

- Adoption Center Staff

  Registers pets available for adoption, manages inquiries, and posts updates.

- Trainer / Service Provider

  Offers training sessions, creates events, and advertises services.

- Administrator

  Manages the system, ensures security, moderates inappropriate content.

B. List of Use Cases

	Actor	Detail
Create pet profile	Pet Owner	User login → chooses "Add Pet" → fills pet's name, age, breed, vaccination info → pet profile stored in database
Post an update	Pet Owner, Veterinarian, Trainer & Organization	User uploads text / photo → system saves to Posts entity → followers see update in feed
Search for adoption pets (Organization)	Pet Owner	User searches adoption listings by species, age, location → system queries Adoption_Pets entity → returns matching pets
Book veterinary appointment	Pet Owner & Veterinarian	Owner selects vet profile → chooses available slot → system logs appointment in database → vet confirms.
Join event	Pet Owner	User browses events → selects "Join" → event participant record created
Send message	Pet Owner, Veterinarian, Trainer & Organization	User opens chat → enters message → system saves to Messages entity → recipient retrieves

III. Functional Database Requirements

A. List of Main Entities

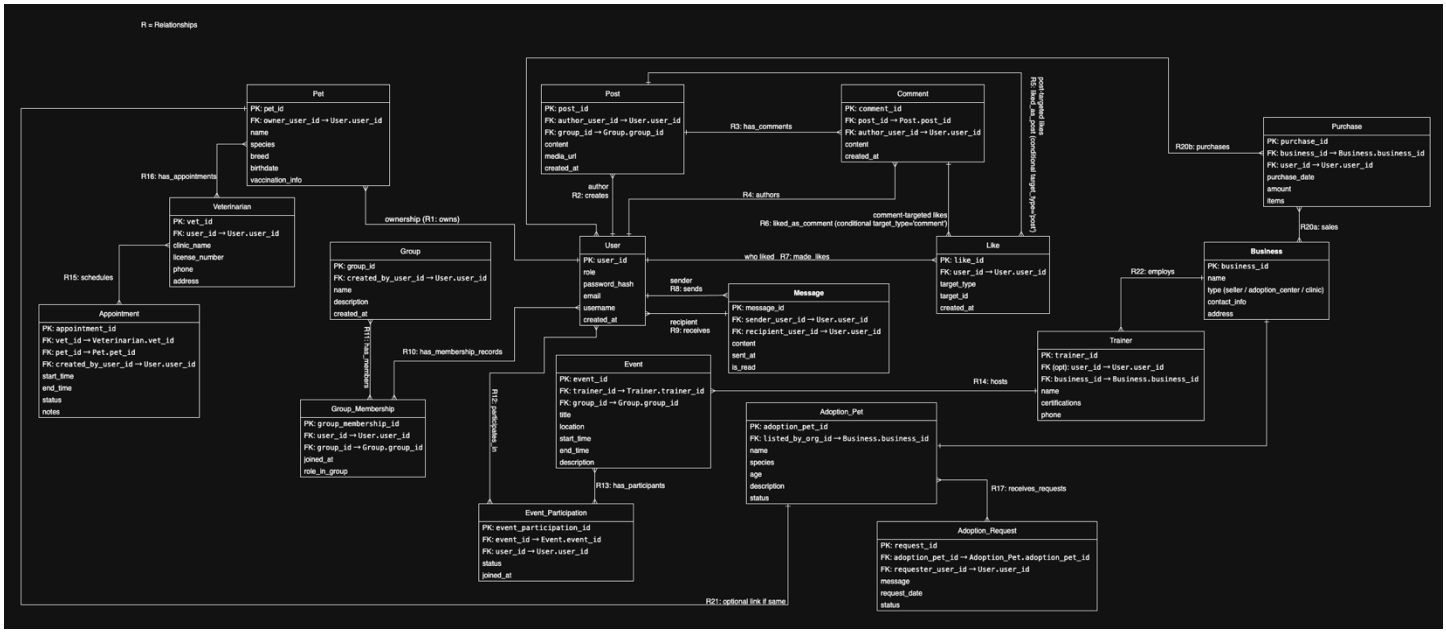
User	stores account details
Pet	stores pet info tied to a user
Post	social updates shared by users
Comment	user responses to posts
Like	reactions to posts or comments
Message	private communication between users
Group	interest-based communities
Group_Membership	links users to groups
Event	local meetups or activities
Event_Participation	users joining events
Veterinarian	registered professionals
Appointment	records pet-vet meetings
Adoption_Pet	pets available for adoption
Adoption_Request	inquiries made by users
Trainer	pet trainers or service providers
Business	pet product/service sellers

B. List of Relationships

User – Pet	"One User" can have "Many Pets"
	"Each Pet" belongs to exactly "One User"
User – Post	"One User" can create "Many Posts"
	"Each Post" belongs to "One User"
Post – Comment	"One Post" can have "Many Comments"
	"Each Comment" belongs to "One Post"
Post – Like	"One Post" can receive "Many Likes"
	"Each Like" belongs to "One Post"
User – Message	"One User" can send "Many Messages"
	"Each Message" belongs to "One Sender" and "One Recipient"
User – Group_Membership – Group	A User can join many Groups
	A Group can have many Users (Many-to-Many via Group_Membership)
User – Event_Participation – Event	A User can join many Events
	An Event can have many Users (Many-to-Many via Event_Participation)

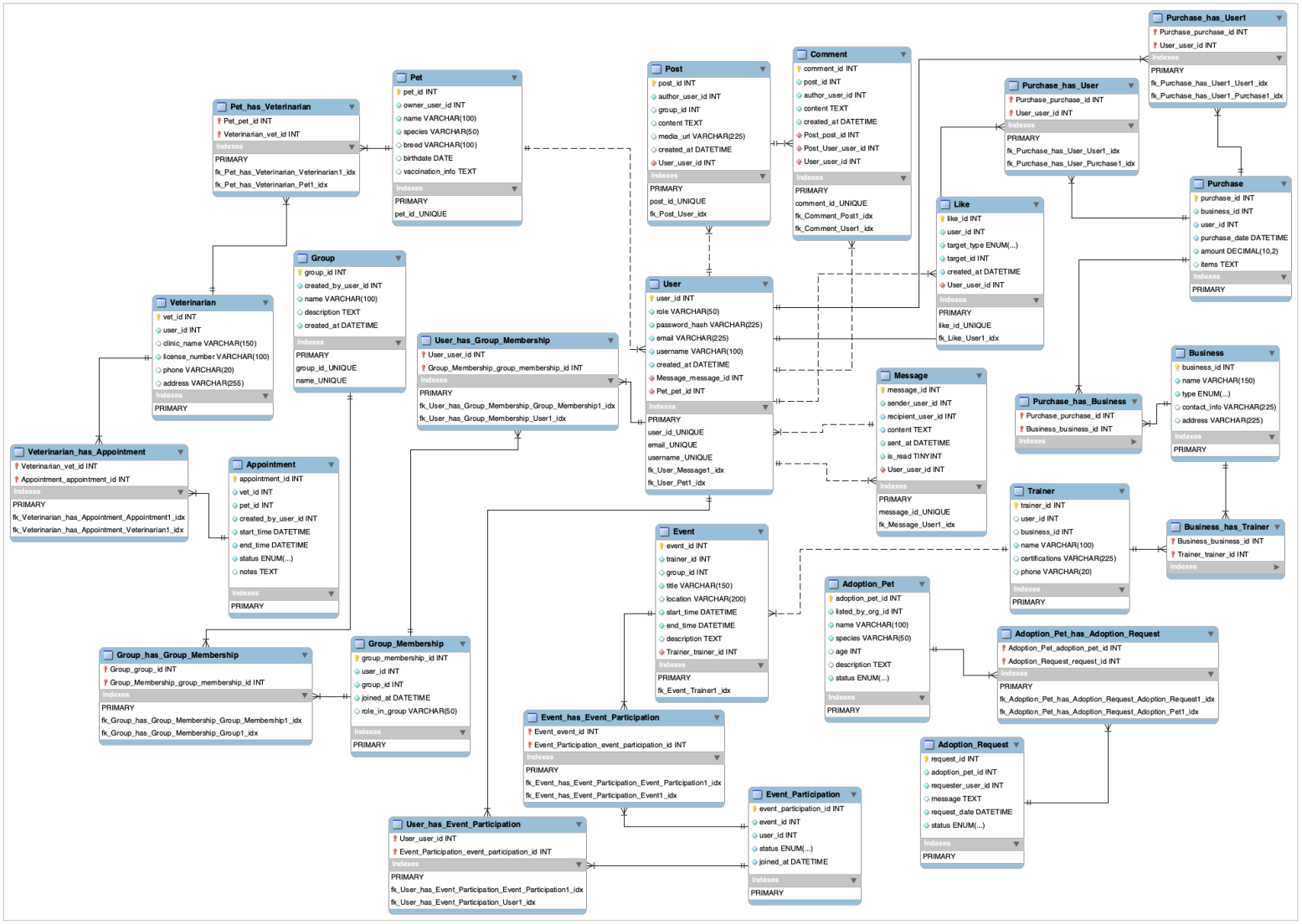
Veterinarian – Appointment – Pet	One Veterinarian can have many Appointments
	One Appointment links one Pet and one Veterinarian
Adoption_Pet – Adoption_Request – User	One Adoption_Pet can have many Requests
	One User can make many Requests
Trainer – Event	One Trainer can host many Events
	Each Event belongs to one Trainer
Business – User	One Business can sell products/services to many Users
	One User can purchase from many Businesses (Many-to-Many)

- **ERD**



IV. Functional Database Requirements

- **EERD**



V. Non-Functional Database Requirements

A. Performance

The platform must support at least 100,000 users including the administrator, and be able to process core queries, such as feed retrieval, and search in 0.20 seconds or less. Database indexes will be created on frequently queried attributes, like "user_id", "pet_id", "event_id").

B. Backup

Daily incremental backups and weekly full backups will be stored in secure cloud storage. Backup and restore testing will be performed monthly to ensure data integrity and disaster recovery.

C. Security

- User passwords stored with hashing and salting.
- Regular security audits and monitoring for suspicious activity.
- Sensitive data, such as medical records, private messages, encrypted at rest.
- Role-based access control (owners, vets, trainers, adoption organizations, admins).

Sample Output

```
=== Insert a test user ===
Saved user id: 12

=== Get user by id ===
UserDto{userId=12, role='normal', username='testuser123', email='testuser@example.com',
createdAt=2025-11-04 14:30:12.0}

=== Update user username ===
UserDto{userId=12, role='normal', username='testuser_updated', email='testuser@example.com',
createdAt=2025-11-04 14:30:12.0}

=== All users (first 10) ===
UserDto{userId=1, role='admin', username='admin1', email='admin@example.com',
createdAt=2024-09-01 09:00:00.0}
UserDto{userId=2, role='vet', username='vetbob', email='vetbob@example.com',
createdAt=2024-10-01 08:00:00.0}
...
=== Join: user with pets (if any) ===
UserWithPet{userId=12, username='testuser_updated', petId=3, petName='Fluffy', species='dog',
breed='Shiba Inu'}
=== Delete test user ===
Deleted. Confirm get() returns null: null
```

```
=== Insert a test pet ===
Saved pet id: 23

=== Get pet by id ===
PetDto{petId=23, ownerUserId=1, name='Fluffy', species='dog', breed='Shiba Inu',
birthdate=2020-05-01}

=== Update pet name ===
PetDto{petId=23, ownerUserId=1, name='Fluffy Jr', species='dog', breed='Shiba Inu',
birthdate=2020-05-01}

=== Find dogs with owners ===
PetWithOwner{petId=23, petName='Fluffy Jr', species='dog', breed='Shiba Inu', ownerId=1,
ownerUsername='petlover'}

=== Delete test pet ===
Deleted. Confirm get() returns null: null
```

```
=== Insert a test appointment ===
Saved appointment id: 34

=== Get appointment by id ===
AppointmentDto{appointmentId=34, vetId=1, petId=1, createdByUserId=1, startTime=2025-11-04
15:30:00.0, endTime=2025-11-04 16:30:00.0, status='scheduled'}

=== Update appointment status ===
AppointmentDto{appointmentId=34, vetId=1, petId=1, createdByUserId=1, startTime=2025-11-04
15:30:00.0, endTime=2025-11-04 16:30:00.0, status='completed'}

=== Get appointments for vet 1 (join) ===
AppointmentWithDetails{appointmentId=34, start=2025-11-04 15:30:00.0, end=2025-11-04 16:30:00.0,
status='completed', petId=1, petName='Fluffy', ownerId=1, ownerName='petlover', vetId=1,
clinicName='Happy Pets Clinic'}

=== Delete appointment ===
Deleted. Confirm get() returns null: null
```