Data Requirements – Veterinary Appointment System

Contributors

Ahmet ATAR — 22290230
Yiğit GÜLBEYAZ — 22290725
Onur Yiğit KOCATÜRK — 22290617
Ömerfaruk SARIBAL — 22290042

Project Description

The Veterinary Appointment System is a web-based platform that simplifies how pet owners interact with veterinary clinics. It enables pet owners to register their pets, book appointments with veterinarians, receive medical diagnoses, and view prescriptions. Veterinarians can manage their schedule, update appointment statuses, record medical information, and prescribe treatments.

This system replaces manual processes with a structured, digital workflow. It ensures that all records are securely stored and easily accessible.

Users and Roles

There are two user types in the system:

- **Pet Owner**: Can register, log in, register their pets, and book appointments.
- **Veterinarian**: Can log in, manage their appointments, and record diagnoses and prescriptions.

Users are stored in one unified table with a `role` attribute (`OWNER`, `VET`) that determines permissions.

Main Entities and Their Attributes

We define 6 core entities in the system:

1. **User**

Stores both pet owners and veterinarians.

- `user_id` (PK)
- `name`
- `email` (unique)
- `password`
- `phone`
- `role` (`OWNER` or `VET`)
- `created_at`

2. **Pet**

Each pet is owned by a user and has species-specific data.

- `pet_id` (PK)
- `owner_id` (FK → User)

- `name`
- `species` (e.g. Cat, Dog)
- `breed`
- `gender`
- `birth_date`
- `registered_at`

3. **Appointment**

Each appointment connects one pet with one vet. An appointment may or may not result in a diagnosis.

- `appointment_id` (PK)
- `pet_id` (FK → Pet)
- `vet_id` (FK → User)
- `appointment_time`
- `status` (`PENDING`, `COMPLETED`, `CANCELED`)
- `created_at`

4. **Diagnosis**

Each diagnosis belongs to one completed appointment and contains medical notes and optional prescriptions.

- `diagnosis_id` (PK)
- `appointment_id` (FK → Appointment)
- `description`
- `diagnosed_at`
- `notes`

5. **Prescription**

Prescriptions are tied to diagnoses and may include multiple entries for different medications.

- `prescription_id` (PK)
- `diagnosis_id` (FK → Diagnosis)
- `medicine_name`
- `dosage`
- `instructions`

6. **Specialization**

Veterinarians can have one or more medical specializations (e.g., Dermatology, Surgery). This is useful for appointment filtering.

- `specialization_id` (PK)
- `vet_id` (FK → User)
- `title` (e.g., "Surgery", "Exotic Animals")

Relationships Between Entities

- One **User** (owner) → many **Pets**
- One **User** (vet) → many **Appointments**
- One **Pet** → many **Appointments**
- One **Appointment** → one **Diagnosis** (optional)
- One **Diagnosis** → many **Prescriptions**
- One **Vet** → many **Specializations**

All foreign key relationships will be enforced via JPA annotations and SQL constraints.

System Behavior (Examples)

- A user registers as an OWNER and adds two pets.
- The user browses available vets and books an appointment for one pet.
- The vet logs in, sees their appointment, completes it, and records a diagnosis.
- A prescription is added with medicine along with notes if needed..
- The owner views the visit details and prescription history.

Constraints and Validations

- Email must be unique (`User.email`)
- A vet cannot have two appointments at the same time (`Appointment.appointment_time + vet_id unique`)
- A pet must belong to exactly one user
- A diagnosis can only be added for completed appointments
- A vet's specialization list can’t contain duplicates

Example Queries (Planned)

- List all pets owned by a specific user
- Find upcoming appointments for a given vet
- Show all past diagnoses for a pet
- List prescriptions issued by a certain vet in the last month
- Filter vets by specialization (e.g., all “Surgery” specialists)

Summary

With six interconnected entities and clear relationships between them, this project demonstrates effective use of relational modeling, basic authentication, and real-world data flows. It provides a strong foundation for Spring Boot + JPA-based database implementation and satisfies the academic requirements of a database course, particularly in the areas of:

- Entity and relationship design
- Primary/foreign key usage
- Data integrity via constraints
- Realistic query needs