**Pet Next Door Project**

**Software Requirements Specification**

**Version 1.0**

**WebApp Requirements Specification**

**Scope**

The Pet Next Door web application is designed for pet lovers, pet owners, and individuals interested in adopting or fostering pets. Users can browse profiles of pets available for adoption, connect with nearby pet owners, and share updates about their pets. Animal shelters and adoption agencies can also use the platform to list pets and manage adoption requests. The web app serves as a community-driven space where people can find pets, exchange care tips, and build meaningful connections around their love for animals.

The scope of the app pertains to all features necessary in the webapp for our local proximity based pet app for pet owners. The requirements primarily focuses on the necessary set of criteria necessary for this webapp to provide basic functionality and utility to our intended consumers. These requirements are agnostic to frameworks, languages, and tools, and are merely intended to provide explicit information of the specifications required for this web app.

1. **Functional Requirements**
   1. **User Profile Creation**
      1. The system shall allow the user to create a profile with matching login credentials
      2. The system shall allow the user to set their location
      3. The system shall require login credentials for any changes to a user’s profile
      4. The system shall allow the user to create multiple pet profiles under their name
      5. Pet Profiles
         1. The system shall allow the user to input a photo into their pet profile, a name, as well as attributes that will be publicly available throughout all other users in the system
            1. These attributes include the user’s pet’s species, energy level, general size, as well as other attributes used in other features
         2. The system shall allow users to view their pet profiles after creation
         3. The system shall allow users to edit each pet’s profile
         4. The system shall allow users to manage individual privacy settings for each pet profile.
      6. The system shall allow the user to update their profile after user authentication
   2. **Timeline**
      1. The system shall display to all users a set of posts made by other users that are close in locational proximity
      2. The system shall allow users to create posts to be displayed on the timeline
         1. Posts may contain a photo and a caption
      3. The system shall allow users to comment on posts that are visible on their timeline
         1. Comments may only contain text
      4. The system shall display the profile name of the user who authored a post, as well as the time when the post was published
      5. The system should automatically update the user’s timeline when new posts are published within a user’s geographical area
   3. **Adoption**
      1. The system shall allow users to indicate whether a pet is available to be adopted, as well as provide additional information relevant to anyone who may want to adopt the pet
      2. The system shall allow users to search for adoptable pets within their area
      3. The system shall allow the user to edit their pet adoption posting for any given pet at any given time
      4. The system shall allow users to refine their search with basic filters and criteria present within the pet’s profile
      5. The system shall allow users to send a message request to other users with a pet that is available to be adopted
      6. The system shall open an active message thread upon the user’s request being accepted by the other user
   4. **Playdates**
      1. The system shall allow users to indicate whether their pet is interested in play dates with other pets
      2. The system shall provide the user with a list of ideally compatible pets within an area that are available for playdates
         1. The system shall compute compatibility based on number of shared interests between pets, as well as pet size, species, and energy level
      3. The system shall allow users to request to message other users to communicate regarding a playdate
      4. The system shall allow users to accept or reject any message requests made for a playdate
      5. The system shall open a message thread if a user agrees to a message request
      6. The system shall allow users to regenerate the list at the user’s request
      7. The system shall allow two users to agree upon a set time for a play date, and will be notified with reminders before the time
   5. **Messaging**
      1. The system shall provide users with a list of all the other users that they have active message threads with
      2. The system shall notify the user when they receive a message
      3. The system shall allow users to send a message to any other user the have an active message thread with
      4. All messages shall be done in a unified chat ui, regardless of whether the chat was initiated via the timeline, adoption, or playdate
      5. Users will be able to send both text and photos via messages
      6. Messages will contain timestamps of when they were sent
2. **Non-Functional Requirements**
   1. **User Interface**
      1. The system shall be compatible with most major browser, as well as most computer systems
      2. The system shall be usable on mobile devices
      3. The system shall be accessible to the average user in terms of ui
      4. The system shall have an onboarding process that gives users an example of the webapp’s features prior to sign up
      5. The system will have different tabs for the user to click on to switch between the different feature sets
   2. **Data Privacy**
      1. All user credentials shall be secured with modern encryption practices
      2. Posts and Information not directly queried by the webapp for it’s features will be unavailable for the User
   3. **General Data Model**
      1. The system’s database should be ACID Compliant to ensure proper concurrency and integrity
      2. All inputs by a user should be sanitized before being inputted into any database, either as an addition to the database or as a database query
      3. The system’s queries should be optimized for proper performance on the user end
         1. Queries should take less than 1 minute on average for the user for features
   4. **Coding Base**
      1. The webapp’s coding base should be managed via Git to ensure best practices for version control, CI, and collaborative coding
      2. Linting practices for code should be implemented within the CI to ensure best practices in code formatting are followed

**Tech Stack**

1. **Frontend**

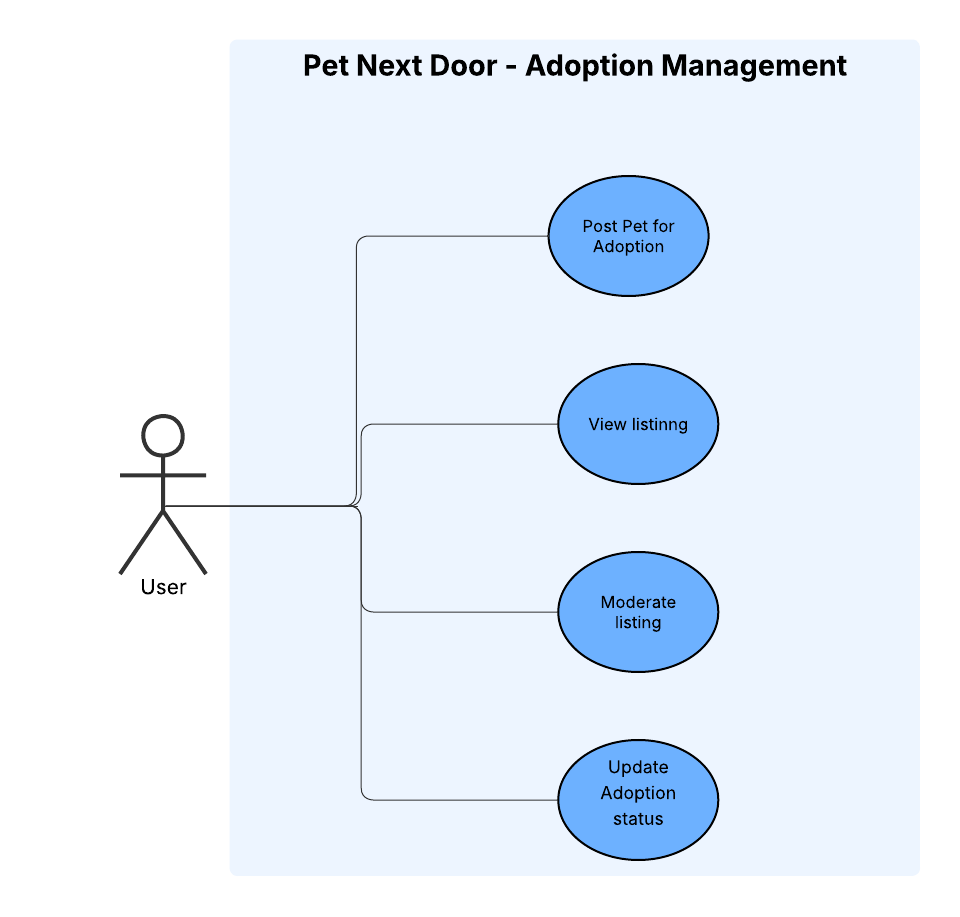
* Framework/Library: Django Templates (server-rendered) + optional HTMX for sprinkles of interactivity without a SPA.
* UI Kit: Tailwind CSS (utility-first, fast to iterate). If you prefer components, add DaisyUI or use Bootstrap instead.
* State Mgmt: Not required (server-rendered). If you add richer client interactions, use Alpine.js (lightweight) alongside HTMX.
* Forms & Validation: Django Forms + django-crispy-forms (Tailwind or Bootstrap template packs). Server-side validation is primary; optional client hints with HTML5 constraints.
* **Routing:** **Django URLconf** (no client router needed).

1. **Backend**

* Runtime: Django (Python)
* API Style: REST via Django REST Framework (DRF) for any endpoints needed by HTMX or future mobile clients. (You can add GraphQL later with Strawberry/Graphene if a strong use case emerges.)
* Auth Mechanism: Django sessions for web (built-in, secure, CSRF-protected). JWT (via djangorestframework-simplejwt) only for API clients if/when required.
* Integration Approach (sentence-level): Integrate external services (payments, emails, etc.) via service-layer modules called from views/DRF viewsets; use Celery tasks for async work (emails, webhooks), and keep provider SDKs behind thin adapters to swap vendors easily.

1. **Database**

* DBSqlite for Development and PostgreSQL for Deployment

**Use cases** 

### **Actors:**

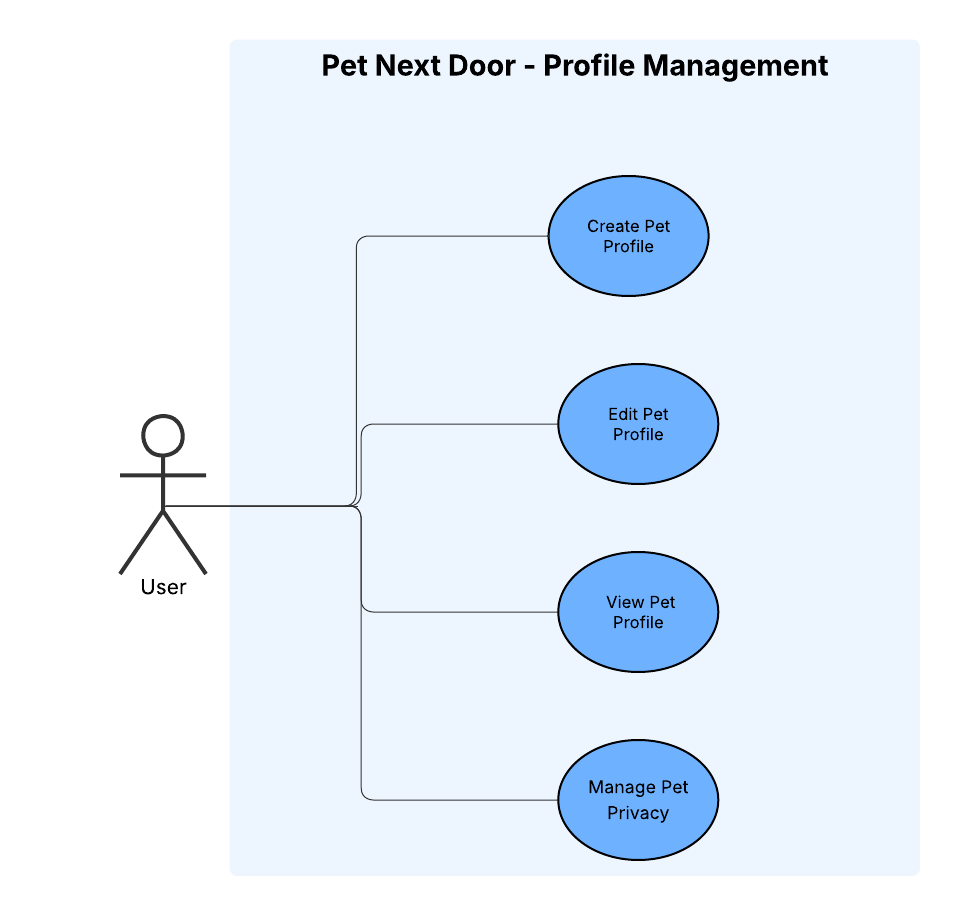
* User – The main actor who interacts with the system.

### **Use Cases:**

1. Post Pet for Adoption – The user can create a listing for a pet that is available for adoption by providing details such as name, age, breed, and adoption requirements.
2. View Listing – The user can browse through available pets listed for adoption.
3. Moderate Listing – The user can update or remove incorrect/duplicate entries (if they are the one who created the listing or if there’s an admin role in the future).
4. Update Adoption Status – Once a pet is adopted, the user can update the status to avoid confusion.

### **Purpose:**

This module manages the entire adoption workflow, ensuring pets are properly listed, visible, and updated.



### **Actors:**

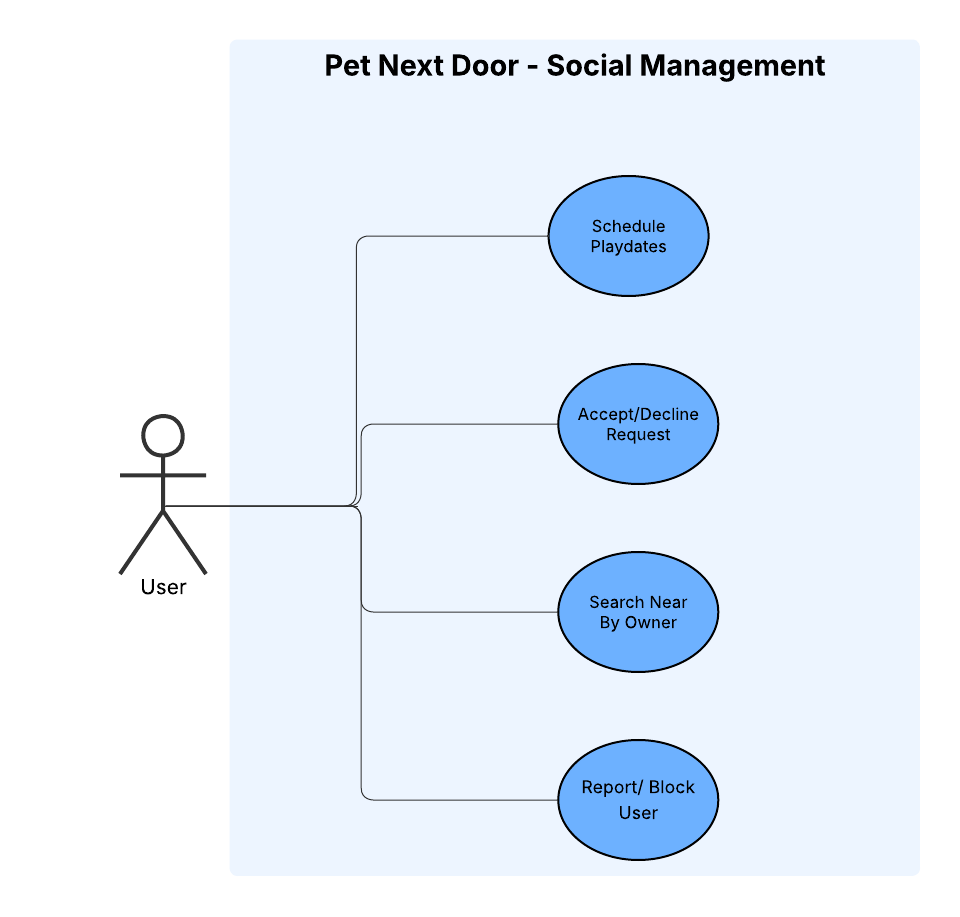
* User

### **Use Cases:**

1. Create Pet Profile – Users can create a profile for their pet with personal details, medical history, and photos.
2. Edit Pet Profile – Users can update information (e.g., if the pet’s vaccination is completed or age changes).
3. View Pet Profile – Other users (or the same user) can view the pet’s profile.
4. Manage Pet Privacy – The user can set visibility levels (e.g., public, private, or friends-only).

### **Purpose:**

This module allows personalized management of pet information and controls visibility/privacy settings.



### **Actor:**

* User (Pet Owner)

### **Use Cases:**

1. Schedule Play Dates – The user can send requests to arrange pet play dates by specifying time, location, and preferences.
2. Accept / Decline Request – When the user receives a play date request, they can either accept it (confirming the event) or decline it.
3. Search Nearby Owner – The user can search for other pet owners in nearby areas, possibly filtered by pet breed, size, or distance.
4. Report / Block User – The user can report inappropriate behavior or block another user to prevent further interactions.

### **Purpose:**

This module provides the social interaction features of the system, allowing pet owners to connect, arrange play dates, and maintain a safe community environment.