Introduction

Tailwaggr's vision is to be a one-stop shop for all things pet, fostering a vibrant and a supportive online community for pet owners and enthusiasts.

Our objectives are to:

- Empower pet owners: Provide a platform that simplifies pet care and strengthens the bond between owners and their furry companions.
- Connect the pet care community: Facilitate connections between pet owners, veterinarians, pet sitters, and other animal lovers through forums and relevant services.
- Enhance pet well-being: Offer resources and tools that promote responsible pet ownership and contribute to the overall health and happiness of pets.

Business Need

The pet care industry is booming, with a growing number of people considering pets as valued members of their families. Tailwaggr addresses this need by:

- Centralizing pet care resources: Simplifying the process of finding qualified pet care professionals and relevant pet information.
- Enhancing convenience: Offering a mobile app allows pet owners to manage their pet's needs and access services on the go.
- Building a community: Providing a platform for pet owners to connect, share experiences, and offer support to each other.
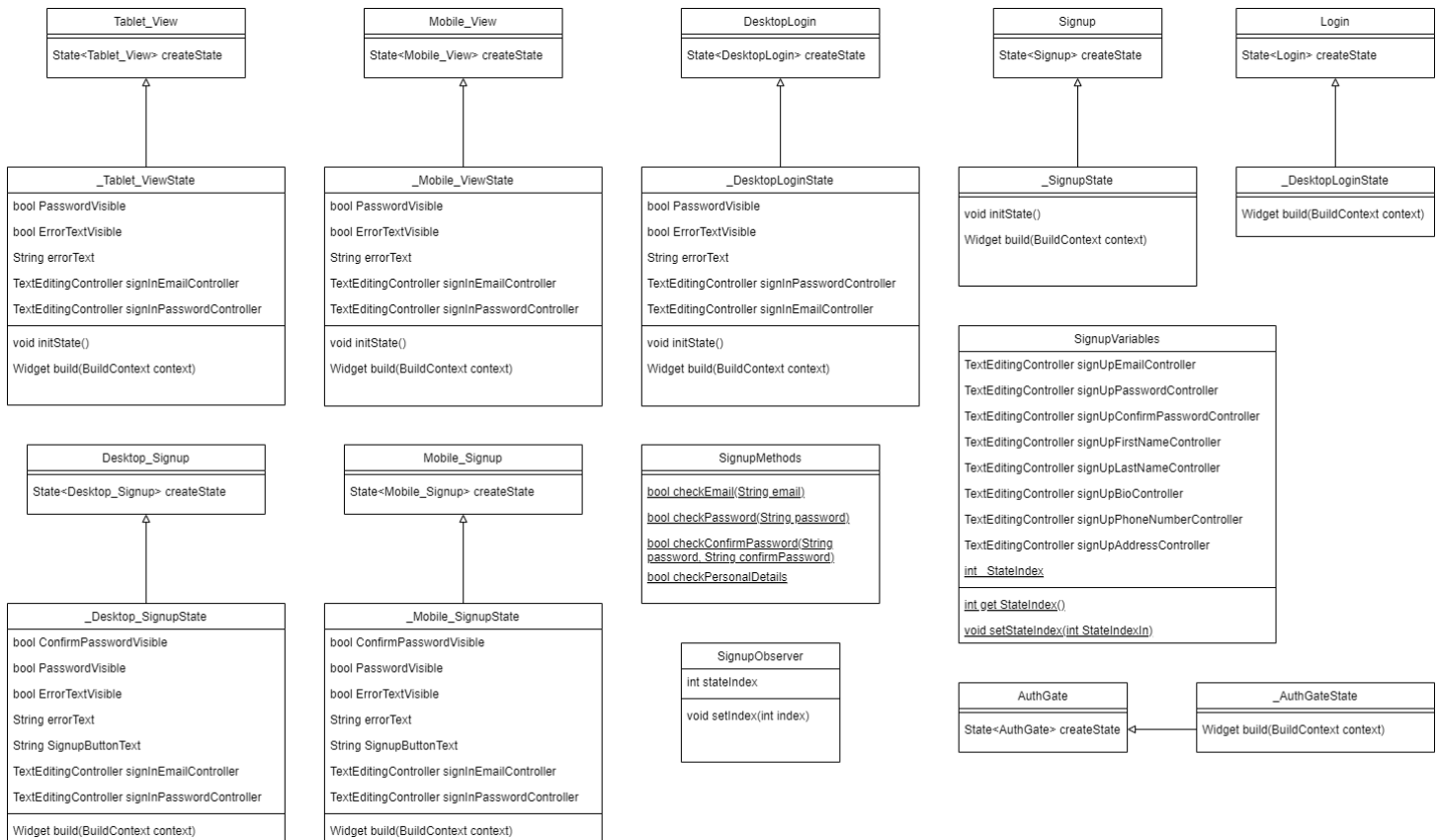
Project Scope

The initial scope of Tailwaggr focuses on core functionalities:

- Pet profiles and owner profiles: Create detailed profiles for pets and their owners, fostering a sense of community and personalization.
- Find a veterinarian: Allow pet owners to locate and connect with licensed veterinarians in their area.
- Find a pet sitter: Enable pet owners to discover reliable pet sitters and dog walkers for in-home or scheduled pet care.

- Pet forums: Establish a dedicated forum section where pet owners can ask questions, share experiences, and build relationships.
- Lost & Found Pets: Create a database to reunite lost pets with their worried owners and facilitate faster reunions.

Class diagram

Tablet_View

State<Tablet_View> createState

Mobile_View

State<Mobile_View> createState

DesktopLogin

State<DesktopLogin> createState

Signup

State<Signup> createState

Login

State<Login> createState

_Tablet_ViewState

bool PasswordVisible

bool ErrorTextVisible

String errorText

TextEditingController signInEmailController

TextEditingController signInPasswordController

void initState()

Widget build(BuildContext context)

_Mobile_ViewState

bool PasswordVisible

bool ErrorTextVisible

String errorText

TextEditingController signInEmailController

TextEditingController signInPasswordController

void initState()

Widget build(BuildContext context)

_DesktopLoginState

bool PasswordVisible

bool ErrorTextVisible

String errorText

TextEditingController signInPasswordController

TextEditingController signInEmailController

void initState()

Widget build(BuildContext context)

_SignupState

void initState()

Widget build(BuildContext context)

_DesktopLoginState

Widget build(BuildContext context)

SignupVariables

TextEditingController signUpEmailController

TextEditingController signUpPasswordController

TextEditingController signUpConfirmPasswordController

TextEditingController signUpFirstNameController

TextEditingController signUpLastNameController

TextEditingController signUpBioController

TextEditingController signUpPhoneNumberController

TextEditingController signUpAddressController

int _StateIndex

int get StateIndex()

void setStateIndex(int StateIndexIn)

Desktop_Signup

State<Desktop_Signup> createState

Mobile_Signup

State<Mobile_Signup> createState

SignupMethods

bool checkEmail(String email)

bool checkPassword(String password)

bool checkConfirmPassword(String password, String confirmPassword)

bool checkPersonalDetails

SignupObserver

int stateIndex

void setIndex(int index)

_Desktop_SignupState

bool ConfirmPasswordVisible

bool PasswordVisible

bool ErrorTextVisible

String errorText

String SignupButtonText

TextEditingController signInEmailController

TextEditingController signInPasswordController

Widget build(BuildContext context)

_Mobile_SignupState

bool ConfirmPasswordVisible

bool PasswordVisible

bool ErrorTextVisible

String errorText

String SignupButtonText

TextEditingController signInEmailController

TextEditingController signInPasswordController

Widget build(BuildContext context)

AuthGate

State<AuthGate> createState

_AuthGateState

Widget build(BuildContext context)

User stories/user characteristics:

User Characteristics:

Normal User:

- Loves their pet
- Looks after their pet
- Makes posts

Pet Caretaker User:

- Has regular user characteristics
- Has contact details listed

* Has location listed

User Stories for TailWagger

User Accounts Subsystem

1. As a new user, I want to sign up for an account so that I can access the application's features.
2. As a user, I want to enter my username during sign-up so that I can create a unique identity.
3. As a user, I want to enter my password during sign-up so that my account is secure.
4. As a user, I want to enter my email address during sign-up so that I can receive notifications and account-related information.
5. As a user, I want to fill in my personal details during sign-up so that my profile is complete.
6. As a returning user, I want to log in to my account so that I can access my profile and pets' information.
7. As a returning user, I want to reset my password if I forget it so that I can regain access to my account.
8. As a user, I want to log in with my email and password so that I can access my account securely.

Profile Subsystem

1. As a user, I want to edit my profile photo so that I can personalise my account.
2. As a user, I want to edit my username so that I can update my display name.
3. As a user, I want to edit my personal details so that my profile information is up-to-date.
4. As a user, I want to add a pet profile so that I can keep track of my pets' information.
5. As a user, I want to add my pet's details so that the profile is complete.
6. As a user, I want to delete my profile so that I can remove my account from the system.
7. As a user, I want to be prompted to confirm profile deletion so that I don't accidentally delete my profile.
8. As a user, I want my profile to be deleted from the database so that my information is removed.
9. As a user, I want to delete a pet profile so that I can remove pets that I no longer own.
10. As a user, I want to be prompted to confirm pet profile deletion so that I don't accidentally delete it.
11. As a user, I want my pet profile to be deleted from the database so that the information is removed.
12. As a user, I want to edit my pet's details so that the profile information is up-to-date.
13. As a user, I want to edit my pet's profile photo so that it is personalised.
14. As a pet caretaker, I want to be able to label my profile appropriately
15. As a user, I want to be able to search for other users

Location Subsystem

1. As a user, I want to search for found pets so that I can find my lost pet.
2. As a user, I want to enter a pet name during the search so that I can find specific lost pets.
3. As a user, I want to view search results so that I can see the found pets' information.
4. As a user, I want to find veterinarians around my area so that I can get medical care for my pet.
5. As a user, I want to see the location of the vet clinic so that I can visit it.
6. As a user, I want to see the vet's details so that I can contact them for an appointment.
7. As a user, I want to find pet caretaker details so that I can find someone to take care of my pet.
8. As a user, I want to register my pet as a lost pet so that people can help locate it.
9. As a user, I want to record my sightings of lost pets.

Content Subsystem

1. As a user, I want to create forums so that I can initiate discussions on relevant topics about pets.
2. As a user, I want to like a forum post so that I can show my appreciation for the content.
3. As a user, I want to comment on a forum post so that I can participate in the discussion.
4. As a user, I want to reply to messages within a forum post to have discussion with others.
5. As a user, I want to be able to like messages within a forum to show appreciation for them.
6. As a user, I want to be able to delete unwanted forums.
7. As a user, I want to be able to create and delete my posts
8. As a user, I want to be able to like and comment on other user's posts

Notifications Subsystem

1. As a user, I want to receive notifications for new comments and likes on my posts so that I can stay updated.
2. As a user, I want to receive notifications for new likes on my forum messages and repplies to them so that I know when others appreciate my content.
3. As a user, I want to receive notifications about requests to follow me.

Functional requirements

1. User Accounts subsystem:
   i. User must be able to Sign up:
      a. Enter their name
      b. User must be able to enter their valid password
      c. User must be able to enter their valid email address

ii. User must be able to login:

    a. User must be able to reset their password

    b. User must be able to login with email

    c. User must be able to login with their password

2. Profile subsystem:

    i. User must be able to edit their profile:

        a. User must be able to edit their profile photo

        b. User must be able to edit their personal details

    ii. User must be able to add a pet profile:

        a. User must be able to add pet details

        b. User must be able to upload pet profile photo

    iii. User must be able to delete their profile:

        a. User must be prompted to confirm profile deletion

        b. User should be deleted from the database

    iv. User must be able to delete a pet profile

        a. User must be prompted to confirm profile deletion

        b. Pet should be deleted from the database

    v. User must be able to edit a pet profile:

        a. User must be able to edit pet details

        b. User must be able to edit pet gallery

    vi. User must be able to become Pet Caretaker

        a. Caretaker must be able to provide contact details

    vii. User must be able to search for other users

        a. User must be able to enter other user's name

        b. User must be able to view matching user search results

    viii. User must be able to change profile between public and private

    ix. User must be able to follow and be followed by other users

3. Location subsystem:

    i. User must be able to search for found pets:

        a. User must be able to enter a pet name

        b. The user must be able to view search results

    ii. User must to able to find Veterinarians around their area:

        a. Users must be able to see the location of the vet clinic

        b. Users must be able to see the vet details

    iii. Users must be able to find a pet caretaker:

        a. Users must be see pet caretaker details

        b. User must be able to see the pet caretaker location
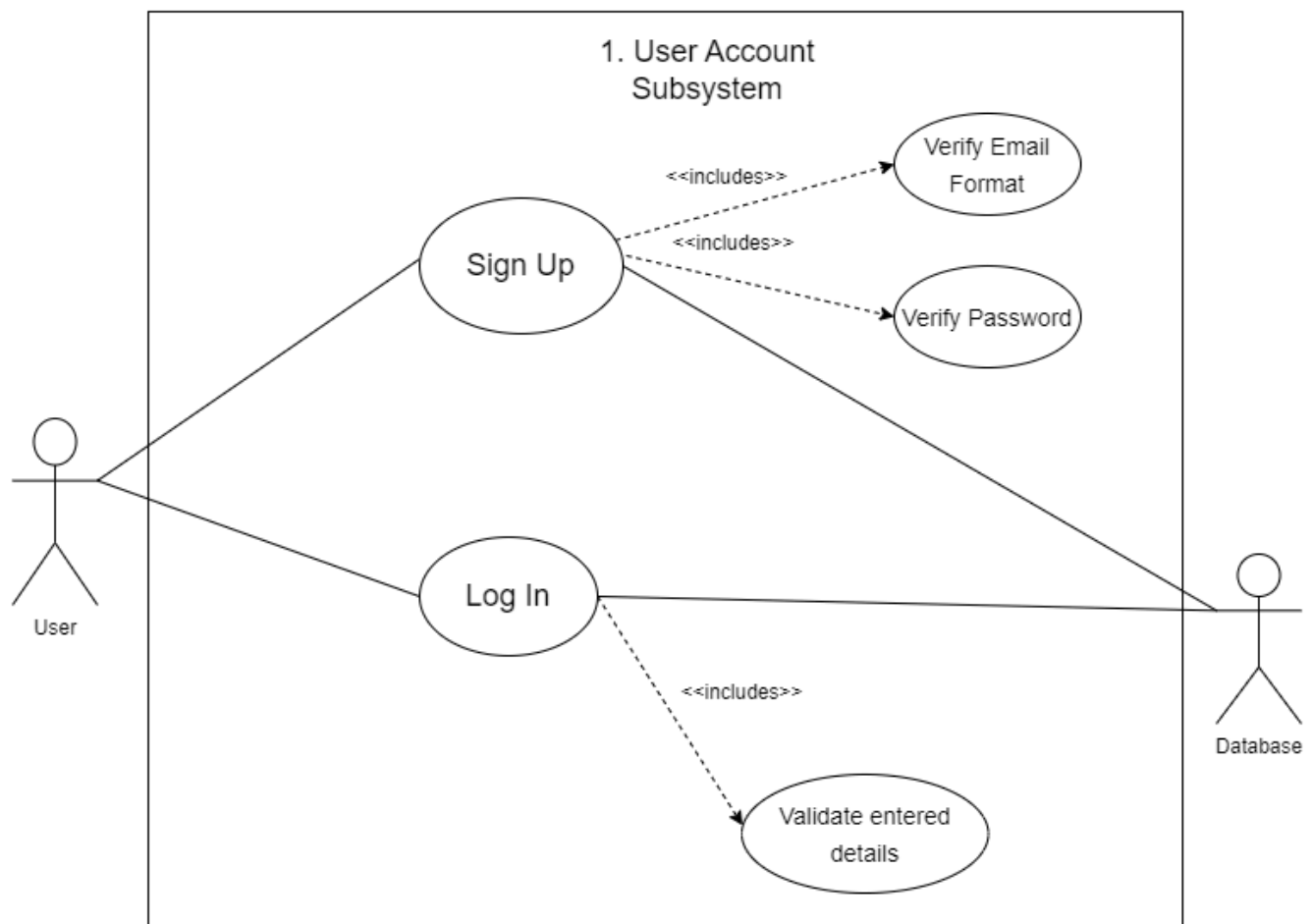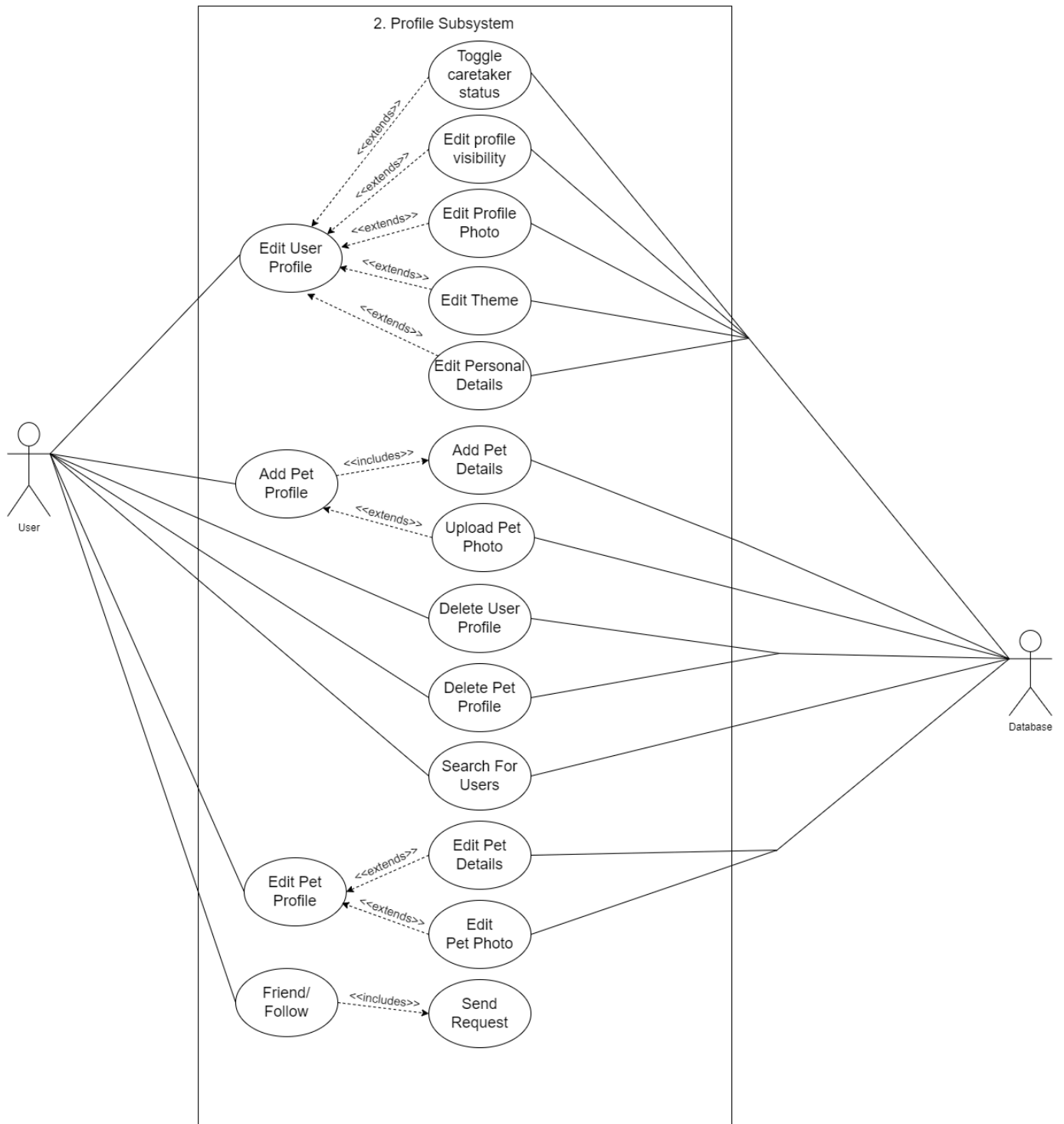
4. Content subsystem:

    i. Users must be able to create forum

        a. Users must be able to create forums

        b. Users must be able to create forum posts

        c. Users must be able to delete their forums

        d. Users must be able to delete their messages

   ii. Users must be able to interact with posts

        a. Users must be able to like a post

        b. Users must be able to share a post

        c. Users must be able to comment on a post

        d. Users must be able to delete their posts
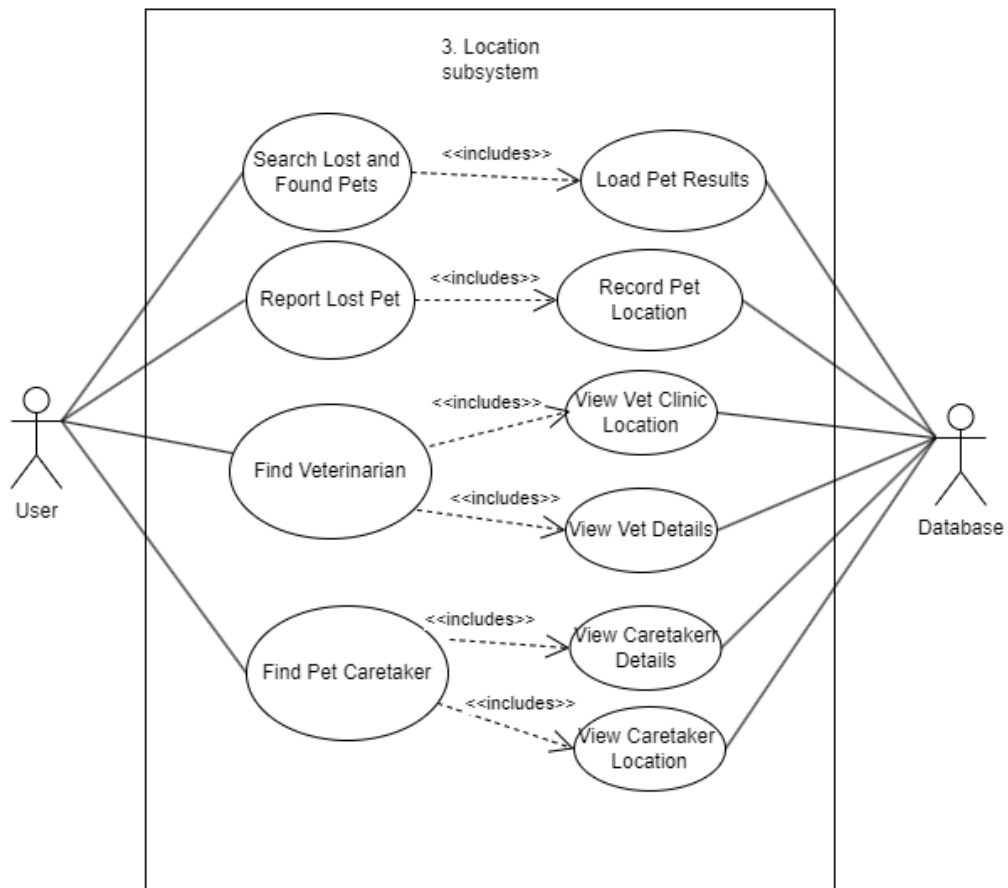
5. Notifications subsystem:

    i. Users must be able to receive notifications from their posts

        a. Users must receive notifications if they receive a like on their posts

        b. User must receive notifications if they receive a comment on their posts

   ii. Users must be able to receive notifications from their forums

        a. Users must be able to receive notifications about new messages on forums they have created

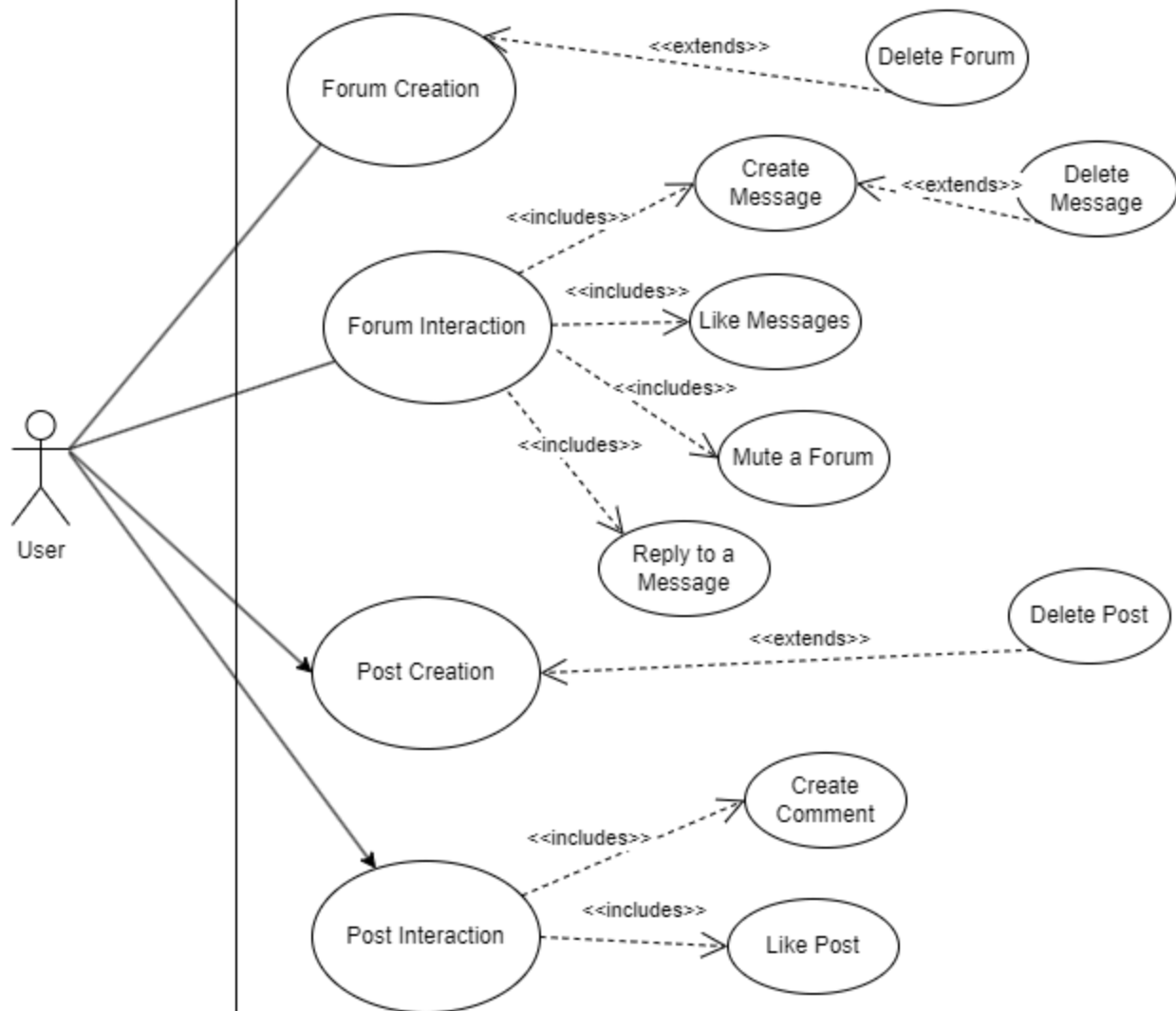   b. Users must be able to receive notifications for replies on their messages on forums
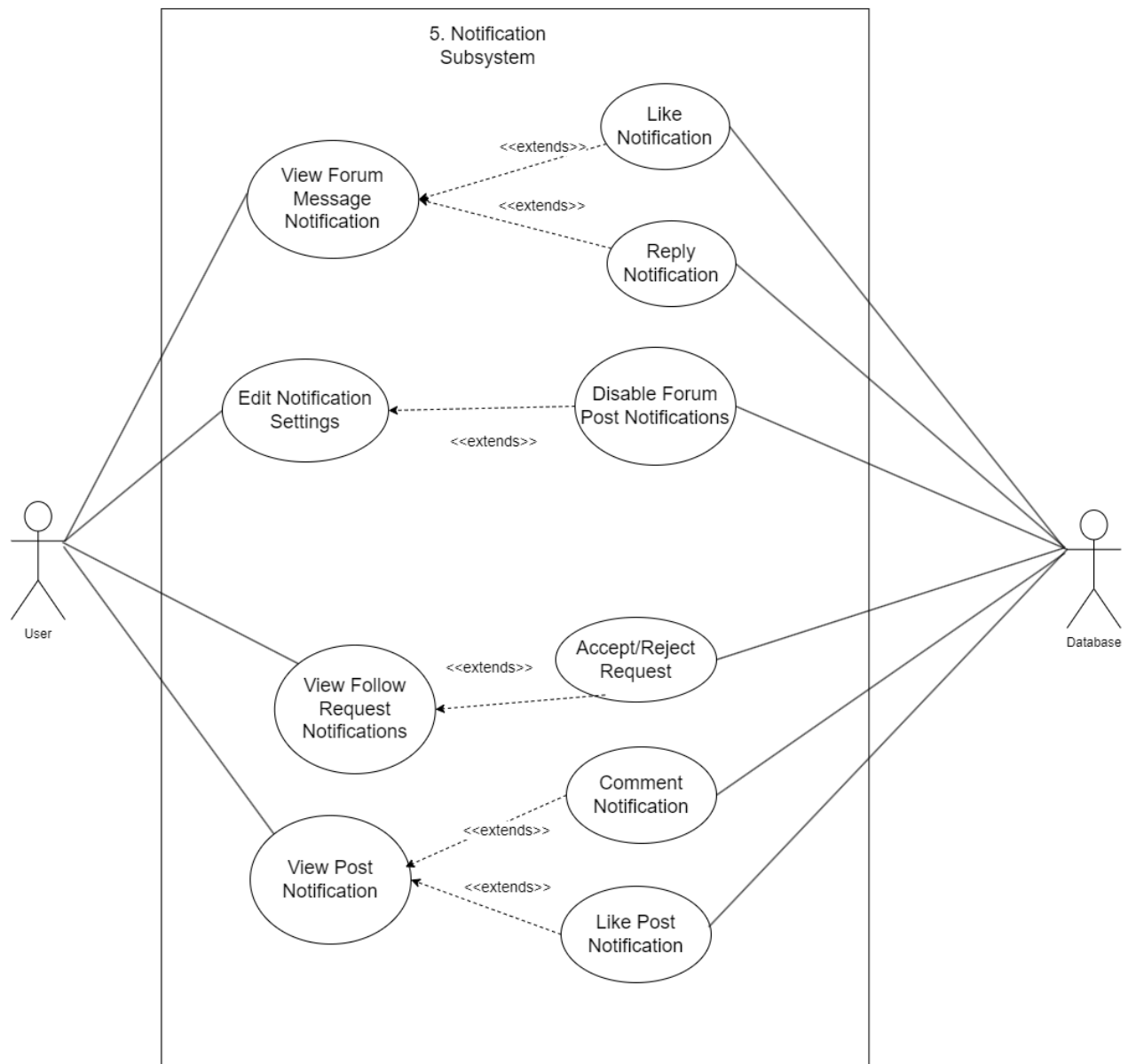
Use Case diagrams:

2. Profile Subsystem

User

Database

Toggle caretaker status

Edit profile visibility

Edit Profile Photo

Edit Theme

Edit Personal Details

Edit User Profile

<<extends>>
<<extends>>
<<extends>>
<<extends>>
<<extends>>

Add Pet Profile

<<includes>>

Add Pet Details

<<extends>>

Upload Pet Photo

Delete User Profile

Delete Pet Profile

Search For Users

Edit Pet Profile

<<extends>>

Edit Pet Details

<<extends>>

Edit Pet Photo

Friend/ Follow

<<includes>>

Send Request

3. Location subsystem

- User
- Search Lost and Found Pets
- `<<includes>>` Load Pet Results
- Report Lost Pet
- `<<includes>>` Record Pet Location
- Find Veterinarian
- `<<includes>>` View Vet Clinic Location
- `<<includes>>` View Vet Details
- Find Pet Caretaker
- `<<includes>>` View Caretaker Details
- `<<includes>>` View Caretaker Location
- Database

Quality Attributes and Architecture Strategies

1. **Performance:**
   - Optimize Firebase queries to ensure efficient data retrieval and usage throughout the application.
   - Use caching mechanisms in Flutter to minimize unnecessary calls to the backend.
   - Implement lazy loading for images and data to improve app load times.
2. **Scalability:**
   - Use Firebase's scalable infrastructure to handle changes in traffic and usage effectively.

- Implement Firebase Functions to handle backend logic, allowing the app to scale horizontally.
- Design the app architecture to support future enhancements and additional features without significant refactoring.

3. **Reliability:**
   - Monitor Firebase backend using Firebase Performance Monitoring to ensure system components are operating within safe parameters.
   - Implement robust error handling and retry mechanisms in Flutter to handle intermittent network issues.
   - Use Firebase Firestore's offline data capabilities to ensure the app remains functional during network disruptions.

4. **Security:**
   - Use Firebase Authentication to manage user sessions securely.
   - Implement Firestore Security Rules to control access to database records.
   - Ensure all data transmitted between the app and backend is encrypted using HTTPS.

5. **Usability:**
   - Design the user interface in Flutter to follow intuitive navigation patterns.
   - Provide customization options for users to adjust the interface to their preferences.
   - Ensure the app is accessible, adhering to WCAG (Web Content Accessibility Guidelines) standards.

6. **Availability:**
   - Utilize Firebase's global infrastructure to ensure high availability and low latency.
   - Implement asynchronous operations in Flutter to maintain a responsive user experience.
   - Use Cloud Firestore's offline capabilities to allow users to continue working during network outages.

7. **Maintainability:**
   - Adhere to a modular architecture in Flutter, separating UI, business logic, and data layers.
   - Use a linter and formatter to enforce coding standards and consistency.
   - Write unit and integration tests for Flutter components and Firebase Functions to ensure reliability and ease of maintenance.

8. **Extensibility:**
   - Use Flutter's plugin system to integrate new features and services seamlessly.
   - Structure Firebase Firestore schema to allow easy addition of new data fields and relationships.
   - Design backend services in a modular fashion to accommodate future enhancements with minimal changes.

9. **Cost-effectiveness:**
   - Utilize Firebase's pay-as-you-go model to manage costs effectively, scaling expenses with usage.

- Optimize cloud function deployments to minimize execution time and reduce costs.
- Use shared components and services in Flutter to reduce development and maintenance costs.