PetWatch
System Design Document
Sprint 2

EECS 3311 N
April 3rd, 2025

Gjergj Kroqi 219387331
Noah Harrison 219562719
Isha Hanchate 219448133
Arjon Sadikaj 218301747
Sakshi Talwar 218875898

Table of Contents

CRC Cards

Class Name: Pet
Responsibilities: Defines a pet entity and represents it with attributes
Collaborators: PetOwner, PetSitter, PetDAO
Attributes: petId, PetOwnerId, petName, petAge, type

Class Name: PetOwner
Responsibilities: Defines a user of the application that is a pet owner and represents them with attributes
Collaborators: Pet, Booking, PetOwnerDAO, BookingDAO, UserDAO
Attributes: petOwnerId, userId, name, phone, address

Class Name: PetSitter
Responsibilities: Defines a user of the application that is a pet sitter and represents them using attributes
Collaborators: Pet, Booking, PetSitterDAO, BookingDAO
Attributes: petSitterId, userId, name, experience, availability, rating

Class Name: User
Responsibilities: Defines and represents a user of the application
Collaborators: PetOwner, PetSitter (typically, a user can be either a PetOwner or a PetSitter), UserDAO, PetDAO
Attributes: id, email, password, role

Class Name: Booking
Responsibilities: Defines and represents a booking in which a pet sitter is looking after the pet of a pet owner
Collaborators: PetOwner, PetSitter, BookingDAO
Attributes: bookingId, petOwner, petSitter, status

Class Name: PetDAO
Responsibilities: Manages database operations relating to pets, such as adding, removing, and retrieving pet entities from the database
Collaborators: Pet, User

Class Name: PetOwnerDAO
Responsibilities: Manages database operations relating to pet owners, such as adding, removing, and retrieving pet owners from the database
Collaborators: PetOwner, BookingDAO

Class Name: PetSitterDAO
Responsibilities: Manages database operations relating to pet sitters, such as adding, removing, and retrieving pet sitters from the database
Collaborators: PetSitter, BookingDAO

Class Name: UserDAO
Responsibilities: Manages database operations relating to users, such as adding, removing, and retrieving users from the database
Collaborators: User, PetOwner, PetSitter

Class Name: BookingDAO
Responsibilities: Manages database operations relating to bookings, such as adding, removing, and retrieving bookings from the database
Collaborators: Booking, PetOwnerDAO, PetSitterDAO

Class Name: BookingPetsDAO
Responsibilities: Manages the database relationships between bookings and pets
Collaborators: Booking, Pet

Class Name: MainController
Responsibilities: Handles the root URL and directs users to the homepage (index.html)

Class Name: LoginController
Responsibilities: Provides endpoints for different login options depending on user type

Class Name: SignupController
Responsibilities: Provides endpoints for different signup options, showing different signup pages depending on user type

Class Name: DashboardController
Responsibilities: Handles user dashboards, directing users to the dashboard for pet owners (dashboard-owner.html) and for pet sitters (dashboard-sitter.html)

Class Name: ApiController
Responsibilities: Handles backend logic and database interactions for user authentication and pet management, connects database to frontend
Collaborators: PetDAO, PetOwnerDAO, PetSitterDAO, UserDAO, Pet, PetOwner, PetSitter, User

Class Name: SecurityConfig
Responsibilities: Handles user authentication, authorization of page views, and user login

System Architecture Description

The architecture of the system follows a model-view-controller (MVC) model. The system is divided into three parts: the model, the view, and the controller. The role of the model is to define and represent the structure of data, to interact with the database using DAOs, and to handle the data logic of requests from the controller, notifying the controller when it is finished. The role of the view is to display data to the user based on requests from the controller and to retrieve user input and send it to the controller. The role of the controller is to handle request flow, receive user input from the view, process user requests, manage application logic, and update the model and view accordingly.

In our application, the model is represented by the Java classes Pet, PetOwner, PetSitter, User, and Booking, which are responsible for defining the internal representations of data and information in our application, as well as the classes PetDAO, PetOwnerDAO, PetSitterDAO, UserDAO, BookingDAO, and BookingPetsDAO, which are responsible for handling database interactions corresponding to each of their classes. The view is represented by HTML files index, signup-choice, signup-owner, signup-sitter, login-choice, login-owner, login-sitter, dashboard-owner, and dashboard-sitter. These are responsible for displaying page structure, content, data, and information to the user. The controller in our application is represented by the JavaScript files dashboard, login, and signup, which are responsible for retrieving and processing user input that the user provides to the view, as well as the Java classes LoginController, MainController, and SignupController, DashboardController, and ApiController which handle the request flow, manage application logic, and communicate with the view.

Software Architecture Diagram