

PetSitter Connect – Full Project Specification

This document expands the initial concept into a complete specification format. It includes structured sections such as Introduction, Scope, User Stories, Functional and Non-Functional Requirements, UI Wireframes Overview, System Architecture, and Technical Notes.

1. Introduction

PetSitter Connect is a web-based platform designed to connect pet owners with trustworthy local pet sitters. The purpose of this document is to define all functional and non-functional requirements, user stories, UI/UX guidelines, and technical aspects needed to develop the MVP version of the application.

2. Scope

The initial release (MVP) will focus on core features enabling pet owners to find, contact, and book sitters safely. Future iterations will extend the system with advanced features such as online payments, premium listings, and mobile applications.

3. Objectives

- Simplify finding and hiring trusted pet sitters.
- Increase transparency through ratings, reviews, and verified profiles.
- Enable secure communication between users.
- Deliver a scalable MVP ready for further product evolution.

4. User Roles and Permissions

Role	Description	Key Permissions
Pet Owner	End-user searching for and booking pet sitters.	Search sitters, send messages, create bookings, leave reviews.
Pet Sitter	Service provider offering pet-sitting services.	Create profile, manage availability, accept bookings, view reviews.
Admin	System manager maintaining quality and security.	Approve sitters, manage users, monitor content, generate reports.

5. User Stories

ID	As a...	I want to...	So that...
US-01	Pet Owner	search pet sitters by city and pet type	I can quickly find nearby services for my pet.
US-02	Pet Owner	view sitter profiles with reviews	I can choose the best sitter for my needs.
US-03	Pet Owner	book a sitter for specific dates	I can plan care for my pet while I'm away.

ID	As a...	I want to...	So that...
US-04	Pet Sitter	create and update my profile	Pet owners can learn more about my experience and availability.
US-05	Pet Sitter	manage booking requests	I can accept or reject jobs based on my schedule.
US-06	Admin	approve new sitters and monitor reviews	I can maintain platform quality and trustworthiness.
US-07	Pet Owner	rate and review a sitter after booking	I can share my experience and help others choose better.
US-08	Pet Sitter	view feedback and overall rating	I can improve my service based on client reviews.

6. Functional Requirements

Each functional requirement below follows the structure used in the Gridlink specification, including purpose, inputs, outputs, preconditions, postconditions, and exceptions.

6.1 User Registration & Login

Purpose: Allow users to securely create accounts and authenticate.

Description:

Users can register via email or Google account. They choose their role (Pet Owner or Pet Sitter) and must verify their email before gaining full access.

Inputs: Email, password, user role, Google auth token (optional).

Outputs: Account confirmation email, user session token.

Preconditions: User has valid internet connection.

Postconditions: User is redirected to dashboard after login.

Exceptions: Invalid email, password mismatch, token expiration.

6.2 Search & Filtering

Purpose: Enable owners to quickly locate suitable sitters.

Description:

Search functionality supports filters by location, pet type, price, availability, and rating. Results are displayed as sitter cards.

Inputs: City, pet type, availability date, price range, rating filter.

Outputs: Filtered sitter list.

Preconditions: At least one sitter is registered in the system.

Postconditions: User sees matching results in under 2 seconds.

Exceptions: No results found, invalid location input.

6.3 Sitter Profiles

Purpose: Provide comprehensive sitter information to help owners make informed choices.

Description:

Profiles include sitter photo, hourly rate, description, experience, calendar, and reviews. Sitter can update info via dashboard.

Inputs: Sitter ID, profile data fields (photo, description, rate).

Outputs: Profile page rendered with dynamic data.

Preconditions: Sitter must be approved by Admin.

Postconditions: Changes are reflected immediately.

Exceptions: Invalid file type for photo upload, missing fields.

6.4 Booking System

Purpose: Manage booking lifecycle from request to confirmation.

Description:

Owners submit booking requests to sitters including dates, pet details, and comments. Sitters respond with accept/decline.

Inputs: Booking request form (owner ID, sitter ID, dates, notes).

Outputs: Booking confirmation or rejection notification.

Preconditions: Both users must have verified accounts.

Postconditions: Booking record stored, notifications sent.

Exceptions: Sitter unavailable, overlapping bookings, timeout in confirmation.

6.5 Messaging System

Purpose: Facilitate direct and secure communication between users.

Description:

A real-time messaging interface using Supabase channels. Notifications appear instantly on both ends.

Inputs: Sender ID, recipient ID, message text.

Outputs: Message displayed in conversation thread.

Preconditions: Active booking or prior contact.

Postconditions: Message stored and retrievable in chat history.

Exceptions: Network failure, authentication timeout.

6.6 Ratings & Reviews

Purpose: Collect and display feedback to ensure trust and transparency.

Description:

After a completed booking, owners can rate sitters from 1–5 stars and leave a short text review. The system calculates average ratings per sitter and displays them in search results.

Inputs: Booking ID, rating value (1–5), review text.

Outputs: Confirmation message, updated sitter rating.

Preconditions: Booking must have “Completed” status.

Postconditions: Review added to sitter profile, rating recalculated.

Exceptions: Duplicate review submission, profanity detected in review text.

6.7 Admin Dashboard

Purpose: Provide administrators with tools to monitor, moderate, and maintain system integrity.

Description:

Admin interface shows user metrics, pending sitter approvals, reported reviews, and booking statistics.

Inputs: Admin credentials, user database queries.

Outputs: Dashboard analytics, approval confirmations, logs.

Preconditions: Admin authentication required.

Postconditions: System data updated upon admin actions.

Exceptions: Database query failure, unauthorized access.

7. Non-Functional Requirements

- **Performance:** All pages should load under 2 seconds on broadband connections.
- **Scalability:** System must support at least 10,000 concurrent users in MVP.
- **Security:** SSL encryption, hashed passwords, and secure authentication.
- **Availability:** 99% uptime target during MVP phase.
- **Accessibility:** WCAG 2.1 AA compliance.

8. Design & UX Guidelines

- Clean and approachable interface with soft green and neutral palette.
- Card-based layout for sitter listings.
- Responsive design using CSS Grid/Flexbox.
- Consistent button placement for primary actions (Book, Message).
- Mobile-first approach ensuring equal functionality on mobile and desktop.

9. UI Wireframes Overview

Main Screens:

1. Home / Search Page
2. Sitter Profile
3. Booking Form
4. Inbox (Messaging)
5. Admin Dashboard

Each screen will have a consistent top navigation bar, floating action buttons on mobile, and footer links for About, Contact, and Terms.

10. System Architecture Overview

- **Frontend:** React / Next.js + Tailwind CSS for responsive layouts.
- **Backend:** Supabase (PostgreSQL + Auth + API Functions).

- **Hosting:** Vercel or Cloudflare Pages.
- **Integrations:** Email notifications (Resend or SendGrid), Google Auth.

11. API Endpoints (Example)

Endpoint	Method	Description
/api/sitters	GET	Retrieve list of sitters with filters.
/api/sitter/:id	GET	Retrieve sitter profile details.
/api/bookings	POST	Create new booking request.
/api/messages	POST	Send new message.
/api/reviews	POST	Submit a review after completed booking.

12. Data Flow & Entity Diagram

Entities: **User, Profile, Booking, Message, Review.**

Flow: Owner → Search → Sitter → Booking → Messaging → Review.

13. Glossary

- **MVP:** Minimum Viable Product – first functional version.
- **Supabase:** Backend-as-a-Service for database, authentication, and API.
- **WCAG:** Web Content Accessibility Guidelines.