



CAPSTONE PROJECT REPORT

Report 4 – Software Design Document

– Hanoi, August 2025 –

Table of Contents

I. Record of Changes.....	3
II. Software Design Document.....	4
1. System Design.....	4
1.1 System Architecture.....	4
1.2 Package Diagram.....	6
2. Database Design.....	11
3. Detailed Design.....	28
3.1 Authentication & Authorization.....	28
3.2 User Profile Management.....	31
3.3 Pet Profile Management.....	34
3.4 Pet Discovery & Swiping.....	37
3.5 Matching.....	39
3.6 Appointment.....	41
3.7 AI Consultation.....	44
3.8 Expert Management.....	47
3.9 Event Management.....	50
3.10 Report Management.....	56
3.11 Subscription & Payment.....	59
3.12 Policy Management.....	61
3.13 Notification & Broadcast.....	66
3.14 Chat & Messaging.....	72
3.15 Block & Privacy.....	75
3.16 Attribute Management.....	79
3.17 Bad Word Management.....	82
4. Class Specifications.....	85
4.1 Models.....	85
4.2 DTO.....	154
4.3 Controller.....	236
4.4 Service.....	282
4.5 Repository.....	400
4.6 IRepository.....	442
4.7 IService.....	472

I. Record of Changes

Date	A*, M, D	In charge	Change Description
12/10/2025	A	DucNV	Software Design Document part 1 (High Level Design) is added
17/10/2025	A	SangNM	Software Design Document part 2 (Detailed Design) is added
24/10/2025	M	DucNV	Software Design Document part 1 is modified
01/11/2025	A	TuanLQ	Software Design Document part 3 (Class Specifications) is added
08/11/2025	M	VietQD	Software Design Document part 2 is modified
12/11/2025	A	CuongNTT	Software Design Document part 4 (Other Design Specifications) is added
15/11/2025	M	CuongNTT, DucNV, TuanLQ	Software Design Document part 4 (Other Design Specifications) is added is modified
26/1/2026	M	DucNV	Software Design Document part 3 is modified

*A - Added M - Modified D - Deleted

II. Software Design Document

1. System Design

1.1 System Architecture

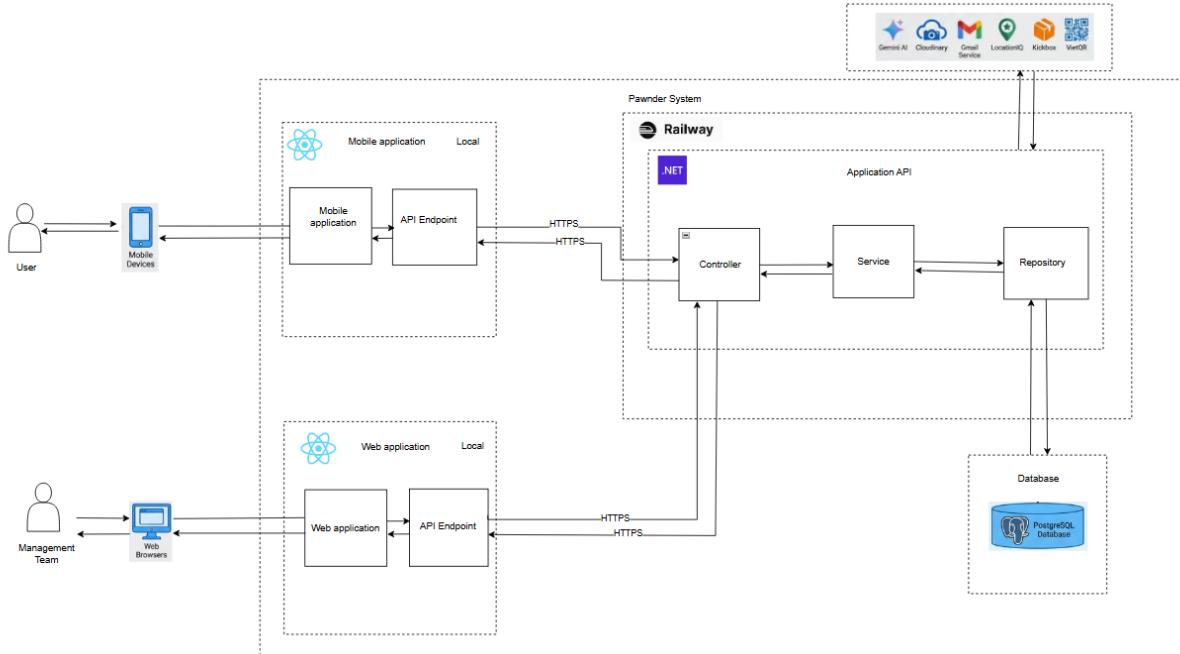


Figure 1.1. System Architecture of the Pawnder system

The Overall Architectural Diagram (Figure 1.1) illustrates the high-level components of the Pawnder system and their interactions. Key components include:

1. Actor:

No	Actor	Role	Interface
1	User	End users who use the app to find and connect pets	Mobile App (React Native)
2	Expert	Experts who verify pet information	Internet Browser (Web)
3	Administrator	System administrators	Internet Browser (Web)

2. Backend Layer:

No	Component	Function
1	Backend API	Handles business logic, JWT authentication, data management controllers
2	SignalR Hub	Handles real-time features: user-to-user chat, expert chat, match/like notifications, typing indicators

3. External service:

No	Service	Function
1	Cloudinary	Pet image storage and management
2	Google Gemini AI	Pet image analysis, auto-generated descriptions, AI chatbot
3	Gmail API	Email notifications, OTP verification
4	VietQR	QR code payment generation for VIP upgrades
5	Kickbox	Email address validation
6	LocationIQ	Reverse geocoding - converts GPS coordinates to readable addresses (city, district, ward)

1.2 Package Diagram

1.2.1 Package Backend

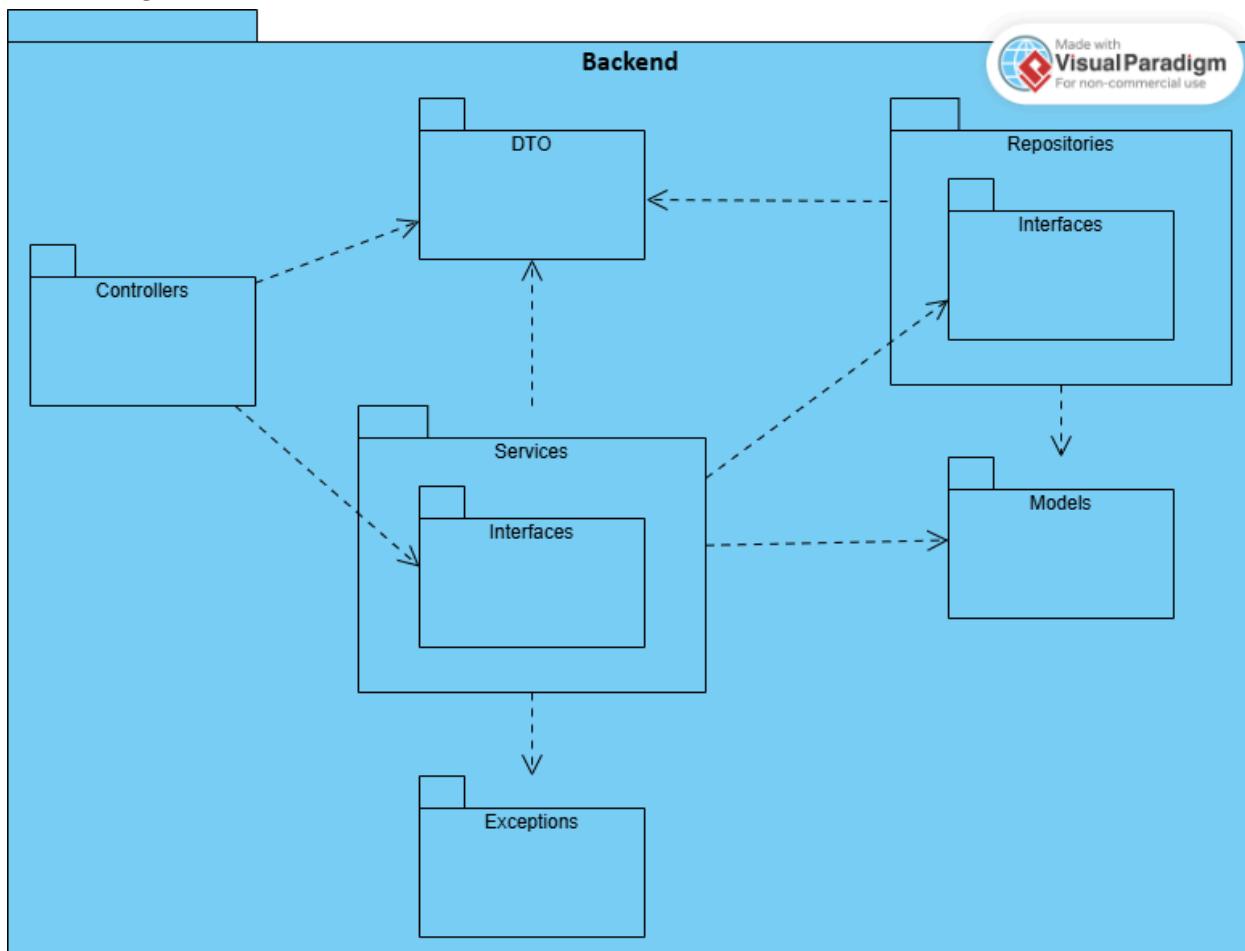


Figure 1.2.1. Backend Package Diagram

Package Descriptions

No	Package	Description
1	Controllers	API Layer - Receives and processes HTTP requests, returns HTTP responses. Contains controller classes that handle application API endpoints.
2	Services	Business Logic Layer - Handles business logic, validation, and orchestration. Contains service classes that implement the main business operations of the application.
3	Services.Interfaces	Service Contracts - Defines interfaces for services. Provides contracts/interfaces to ensure consistency and facilitate testing and mocking.
4	Repositories	Data Access Layer - Queries database, performs CRUD operations. Contains repository classes that interact directly with the database.

5	Repositories.Interfaces	Repository Contracts - Defines interfaces for repositories. Provides contracts/interfaces to ensure consistency and facilitate testing and mocking.
6	Models	Entity Layer - Database entities, DbContext, and domain models. Contains entity classes representing database tables and DbContext for managing database connections.
7	DTO	Data Transfer Objects - Request/Response models, API contracts. Contains DTO classes used to transfer data between layers, ensuring data is formatted correctly according to API requirements.
8	Exceptions	Custom Exceptions - Custom error handling, error management. Contains custom exception classes to handle special error cases in the application.

1.2.2 Package Frontend For User

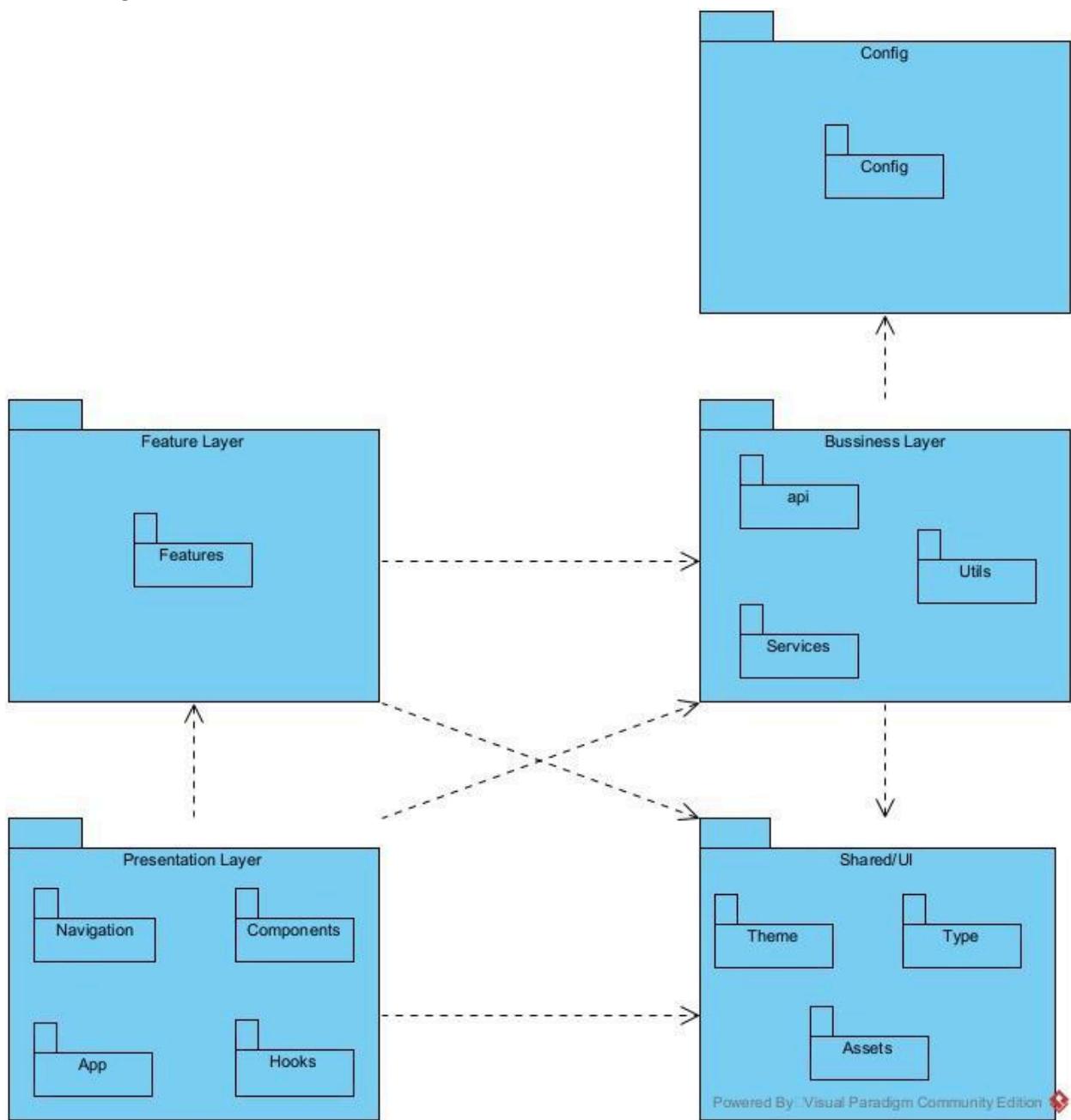


Figure 1.2.2. Package Frontend For User Diagram

Package Descriptions

No	Package	Description
1	api	Contains API client functions and endpoints. Handles all HTTP requests to the backend, including authentication, data fetching, and data submission.

2	app	Contains Redux store configuration, hooks, and global application state management. Manages application-wide state and data flow.
3	components	Contains shared, reusable React Native components used across the application. Includes modals, skeletons, navigation components, and other common UI elements.
4	config	Contains application configuration files, API endpoints, and environment settings. Centralizes configuration management for the application.
5	features	Contains feature-based modules organized by domain. Each feature module includes screens, components, and related logic for a specific application feature.
6	hooks	Contains custom React hooks for reusable logic, API cancellation, badge notifications, and custom alerts. Provides shared hook functionality across the application.
7	navigation	Contains navigation setup, route definitions, and navigation configuration. Manages application routing and screen transitions.
8	services	Contains service classes for location services, navigation services, and SignalR real-time communication. Handles external service integrations and real-time features.
9	theme	Contains theme definitions, colors, and styling constants. Centralizes application theming and color management.
10	types	Contains TypeScript type definitions and global type declarations. Provides type safety and "IntelliSense" support across the application.
11	utils	Contains utility functions for API caching, retry logic, image optimization, JWT handling, storage, and other helper functions. Provides shared utility functionality.
12	assets	Contains static files including images, icons, fonts, and other media resources used throughout the application. Includes avatar images, welcome screens, and UI graphics.

1.2.3 Package Frontend For Admin/Expert

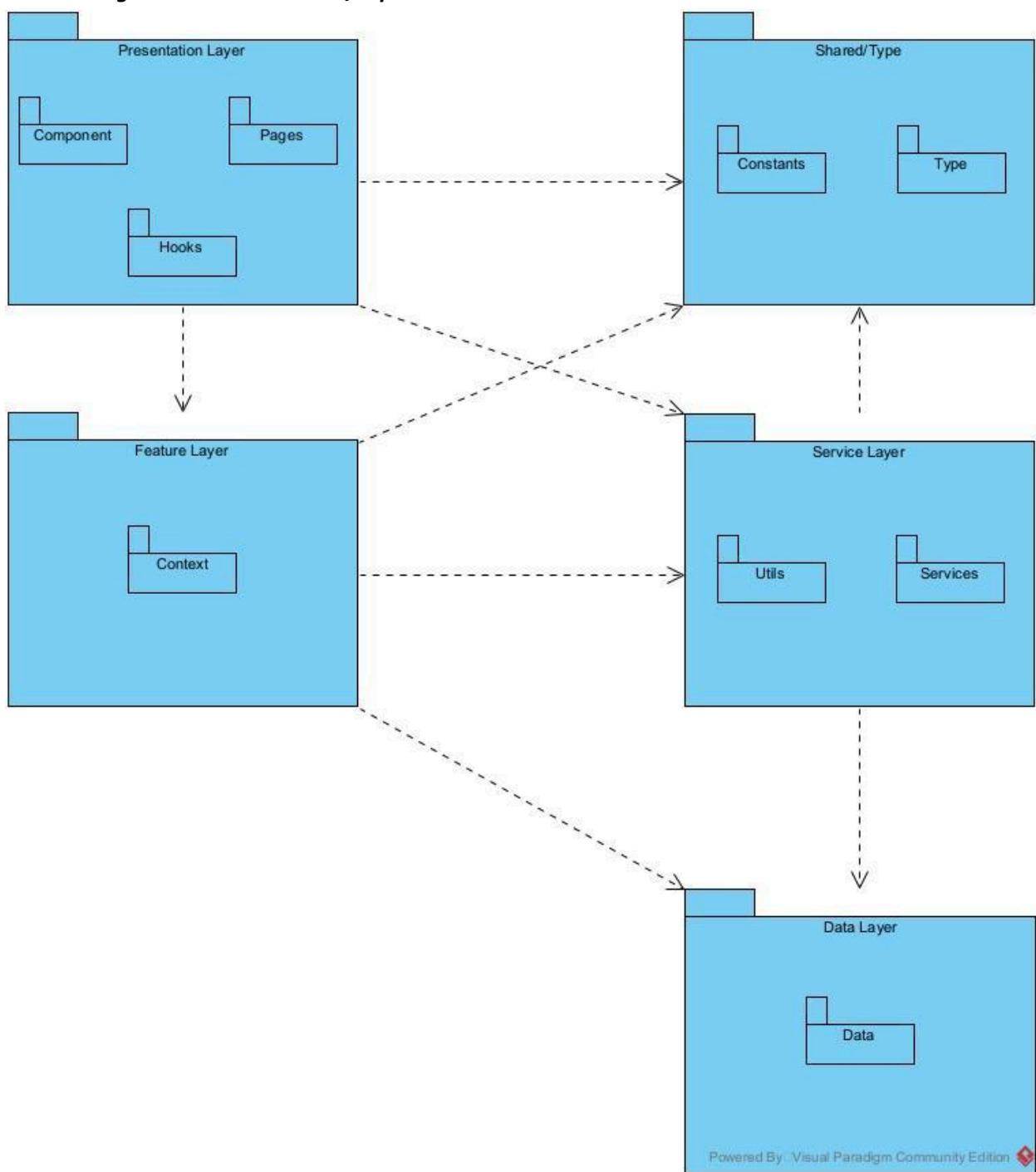


Figure 1.2.3. Package Frontend For Admin/Expert Diagram

Package Descriptions

No	Package	Description
1	components	Contains shared, reusable React components used across the admin application. Includes common components like Header, Sidebar, Layout components, and protected routes.
2	constants	Contains constant values, configuration constants, and application-wide static values used throughout the admin application.
3	context	Contains React Context providers for authentication, notifications, and theme management. Manages global application state using React Context API.
4	data	Contains mock data files for development and testing purposes. Includes sample data for payments, pets, reports, user notifications, and users.
5	hooks	Contains custom React hooks for reusable logic and shared hook functionality across the admin application.
6	pages	Contains page-level components organized by feature. Includes pages for authentication, dashboard, expert management, payment management, pet management, report management, user management, and activities.
7	services	Contains service classes for API communication, authentication, and business logic. Includes API client, dashboard service, expert service, notification service, pet service, report service, and user service.
8	types	Contains TypeScript type definitions and type declarations. Provides type safety and “IntelliSense” support across the admin application.
9	utils	Contains utility functions for formatting (currency, date), validation (email), JWT handling, storage, debouncing, and other helper functions. Provides shared utility functionality.

2. Database Design

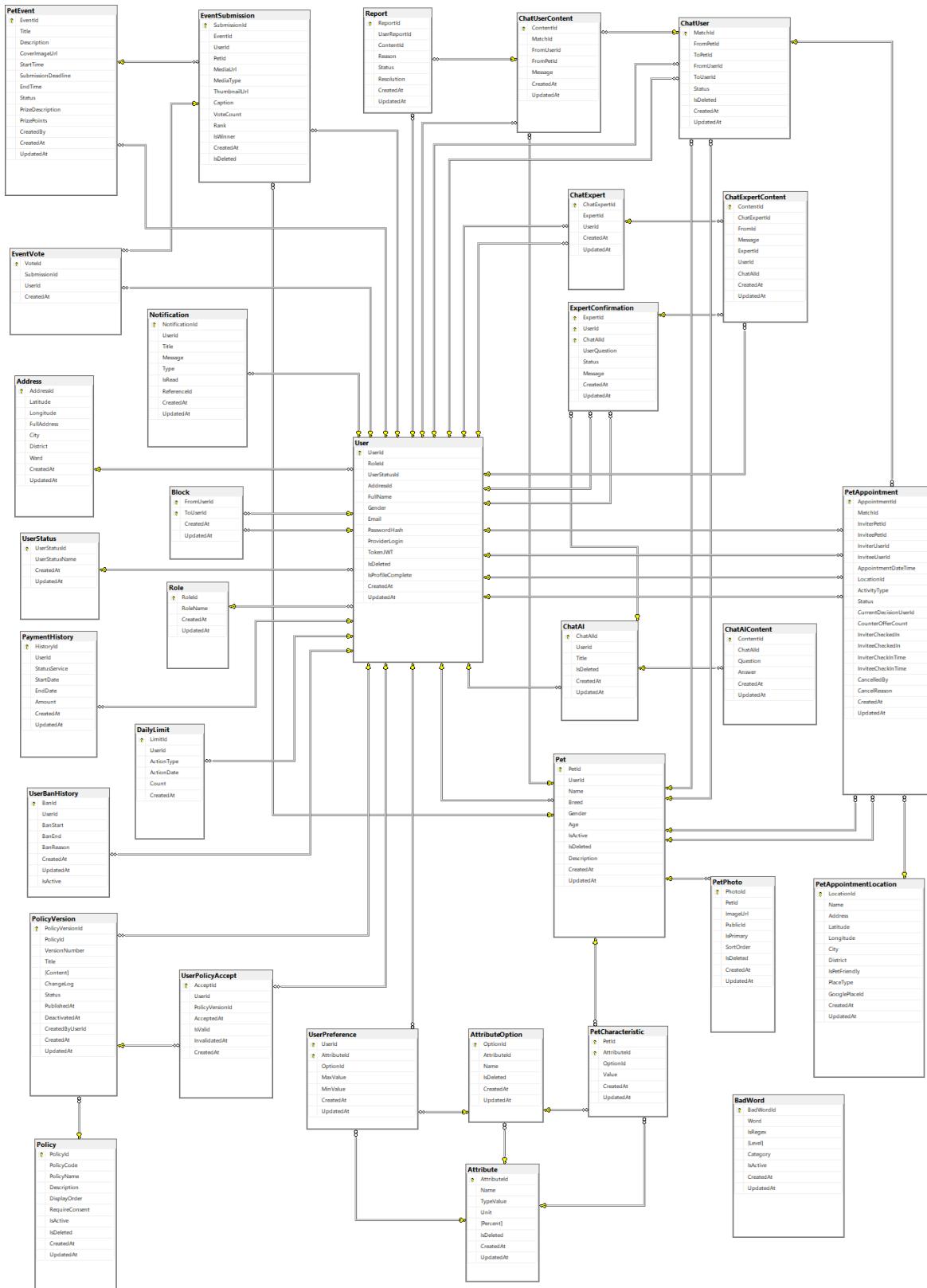


Figure 2. Database Design

2.1 Role

Description: This table stores roles available in the system (e.g., Admin, User).

No	Field	Data Type	Data Size	Constraint	Description
1	RoleId	integer		PK	Primary key, unique identifier for the role.
2	RoleName	varchar	50	NN	Name of the role.
3	CreatedAt	timestamp		NN	Timestamp when the role record was created.
4	UpdatedAt	timestamp		NN	Timestamp when the role record was last updated.

2.2. UserStatus

Description: This table stores possible statuses for users (e.g., Active, Banned, Pending).

No	Field	Data Type	Data Size	Constraint	Description
1	UserStatusId	integer	–	PK	Primary key, unique identifier for user status.
2	UserStatusName	varchar	50	NN	Name of the status.
3	CreatedAt	timestamp	–	NN	Timestamp when created.
4	UpdatedAt	timestamp	–	NN	Timestamp when updated.

2.3 Address

Description: This table stores user address information.

No	Field	Data Type	Data Size	Constraint	Description
1	AddressId	integer	–	PK	Primary key.
2	Latitude	numeric	9,6	NN	Latitude coordinate.
3	Longitude	numeric	9,6	NN	Longitude coordinate.
4	FullAddress	text	–	NN	Complete address text.
5	City	varchar	100	–	City name.
6	District	varchar	100	–	District name.
7	Ward	varchar	100	–	Ward name.

2.4 User

Description: This table stores user account information.

No	Field	Data Type	Data Size	Constraint	Description
1	UserId	integer	—	PK	Primary key.
2	RoleId	integer	—	FK	References Role table.
3	UserStatusId	integer	—	FK	References UserStatus table.
4	AddressId	integer	—	FK	References Address table.
5	FullName	varchar	100	NN	Full name of the user.
6	Gender	varchar	10	NN	Gender (Male/Female/Other).
7	Email	varchar	150	UN	User email must be unique.
8	PasswordHash	text	—	NN	Encrypted password.
9	ProviderLogin	varchar	50	—	Login method (e.g., Google).
10	TokenJWT	text	—	—	JWT token for session.
11	IsDeleted	boolean	—	NN	Soft delete flag.
12	IsProfileComplete	boolean	—	NN	Whether the user profile is complete.
13	CreatedAt	timestamp	—	NN	Created timestamp.
14	UpdatedAt	timestamp	—	NN	Updated timestamp.

2.5 Attribute

Description: This table stores characteristics/attributes that can be used to describe pets or user preferences.

No	Field	Data Type	Data Size	Constraint	Description
1	AttributeId	integer	—	PK	Primary key, unique identifier for the attribute.
2	Name	varchar	100	UN, NN	Name of the attribute (e.g., Coat Length, Color eye).
3	TypeValue	varchar	50	NN	Data type of the value (e.g., option, range, text).

4	Unit	varchar	20	–	Unit of measurement if numeric (e.g., cm, kg).
5	Percent	numeric	5,2	NN	Weight percentage for scoring.
6	IsDeleted	boolean	–	NN	Soft delete flag.
7	CreatedAt	timestamp	–	NN	Timestamp when created.
8	UpdatedAt	timestamp	–	NN	Timestamp when last updated.

2.6 AttributeOption

Description: This table stores selectable options for attributes that use dropdown/choice values.

No	Field	Data Type	Data Size	Constraint	Description
1	OptionId	integer	–	PK, UN, NN	Primary key.
2	AttributeId	integer	–	FK, NN	References Attribute table.
3	Name	Character varying	100	NN	Display name of the option.
4	IsDeleted	boolean	–	NN	Soft delete flag.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.7 UserPreference

Description: This table stores the user's preferred pet characteristics.

No	Field	Data Type	Data Size	Constraint	Description
1	UserId	integer	–	PK, FK	References User table.
2	AttributeId	integer	–	PK, FK	References Attribute table.
3	OptionId	integer	–	FK	References AttributeOption table (nullable).
4	MaxValue	integer	–	–	Maximum acceptable numeric value for range attributes.
5	MinValue	integer	–	–	Minimum acceptable numeric value for range attributes.
6	CreatedAt	timestamp	–	NN	Created timestamp.

7	UpdatedAt	timestamp	–	NN	Updated timestamp.
---	-----------	-----------	---	----	--------------------

2.8 Pet

Description: This table stores registered pets owned by users.

No	Field	Data Type	Data Size	Constraint	Description
1	PetId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	Pet owner, references User table.
3	Name	varchar	100	NN	Pet's name.
4	Breed	varchar	100	NN	Breed type.
5	Gender	varchar	10	NN	Gender of the pet.
6	Age	integer	–	NN	Age in years.
7	IsActive	boolean	–	NN	Whether the pet is visible/active.
8	IsDeleted	boolean	–	NN	Soft delete flag.
9	Description	text	–	–	Additional details about the pet.
10	CreatedAt	timestamp	–	NN	Created timestamp.
11	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.9 PetPhoto

Description: This table stores pet images.

No	Field	Data Type	Data Size	Constraint	Description
1	Photoid	integer	–	PK	Primary key.
2	PetId	integer	–	FK	References Pet table.
3	ImageUrl	text	–	NN	URL of the image.
4	PublicId	text	–	NN	Cloud storage identifier.
5	IsPrimary	boolean	–	NN	Whether this is the main photo.
6	SortOrder	integer	–	NN	Ordering index for multiple photos.
7	IsDeleted	boolean	–	NN	Soft delete flag.

8	CreatedAt	timestamp	–	NN	Created timestamp.
9	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.10 PetCharacteristic

Description: This table stores attribute values for each pet.

No	Field	Data Type	Data Size	Constraint	Description
1	PetId	integer	–	PK, FK	References Pet table.
2	AttributeId	integer	–	PK, FK	References Attribute table.
3	OptionId	integer	–	FK	References AttributeOption table (nullable).
4	Value	integer	–	–	Numeric value if TypeValue is range.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.11 ChatAI

Description: This table stores roles available in the system (e.g., Admin, User).

No	Field	Data Type	Data Size	Constraint	Description
1	ChatAId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	References User table; session owner.
3	Title	varchar	200	NN	Display title for the chat session.
4	IsDeleted	boolean	–	NN	Soft delete flag.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.12 ChatAIContent

Description: This table stores messages exchanged in a ChatAI session.

No	Field	Data Type	Data Size	Constraint	Description
1	ContentId	integer	–	PK	Primary key.
2	ChatAId	integer	–	FK	References ChatAI table.

3	Question	text	–	NN	User's query text.
4	Answer	text	–	NN	AI's response text.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.13 ExpertConfirmation

Description: This table stores confirmation/verification actions performed by experts on ChatAI results.

No	Field	Data Type	Data Size	Constraint	Description
1	ExpertId	integer	–	PK, FK	References User table; expert performing confirmation.
2	UserId	integer	–	PK, FK	References User table; user who asked AI.
3	ChatAId	integer	–	PK, FK	References ChatAI table.
4	Status	text	–	NN	Confirmation status (e.g., Approved, Rejected).
5	Message	text	–	–	Additional explanation from the expert.
6	CreatedAt	timestamp	–	NN	Created timestamp.
7	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.14 ChatUser

Description: This table stores matching/relationship status between two users for chatting.

No	Field	Data Type	Data Size	Constraint	Description
1	MatchId	integer	–	PK	Primary key.
2	FromUserId	integer	–	FK	Initiator, references User table.
3	ToUserId	integer	–	FK	Receiver, references User table.
4	Status	varchar	50	NN	Match/chat status (e.g., Approved, Blocked).
5	IsDeleted	boolean	–	NN	Soft delete flag.
6	CreatedAt	timestamp	–	NN	Created timestamp.

7	UpdatedAt	timestamp	–	NN	Updated timestamp.
---	-----------	-----------	---	----	--------------------

2.15 ChatUserContent

Description: This table stores message content exchanged between matched users.

No	Field	Data Type	Data Size	Constraint	Description
1	ContentId	integer	–	PK	Primary key.
2	MatchId	integer	–	FK	References ChatUser table.
3	FromUserId	integer	–	FK	References User table; sender.
4	Message	text	–	NN	Chat message text.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.16 Report

Description: This table stores reports of inappropriate behavior in chats.

No	Field	Data Type	Data Size	Constraint	Description
1	ReportId	integer	–	PK	Primary key.
2	UserReportId	integer	–	FK	References User table; who reported.
3	ContentId	integer	–	FK	References ChatUserContent table.
4	Reason	text	–	NN	Reason for reporting.
5	Status	Character varying	50	NN	Status (Pending, Reviewed).
6	Resolution	text	–	–	Result or admin action taken.
7	CreatedAt	timestamp	–	NN	Created timestamp.
8	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.17 Block

Description: This table stores records of users blocking another user.

No	Field	Data Type	Data Size	Constraint	Description
1	FromUserId	integer	–	PK, FK	References User; who blocks.

2	ToUserId	integer	–	PK, FK	References User; who is blocked.
3	CreatedAt	timestamp	–	NN	Created timestamp.
4	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.18 PaymentHistory

Description: This table stores subscription or service purchase history.

No	Field	Data Type	Data Size	Constraint	Description
1	HistoryId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	References User table.
3	StatusService	Character varying	100	NN	Status Service (e.g., Active, Waiting, Processing).
4	StartDate	date	–	NN	Subscription start date.
5	EndDate	date	–	NN	Subscription end date.
6	CreatedAt	timestamp	–	NN	Created timestamp.
7	UpdatedAt	timestamp	–	NN	Updated timestamp.
8	Amount	numeric	10,2	NN	The amount the user transfers.

2.19 Notification

Description: This table stores notifications for users from the system.

No	Field	Data Type	Data Size	Constraint	Description
1	NotificationId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	References User table.
3	Title	Character varying	200	NN	Notification title text.
4	Message	text	–	NN	Notification message content.
5	CreatedAt	timestamp	–	NN	Created timestamp.
6	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.20 UserBanHistory

Description: Stores a user's ban records — start/end timestamps, reason, active flag, and created/updated timestamps.

No	Field	Data Type	Data Size	Constraint	Description
1	NotificationId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	References User table.
3	BanStart	timestamp	–	NN	Start ban timestamp.
4	BanEnd	timestamp	–	NN	End ban timestamp.
5	BanReason	text	–	NN	Reasons for users being banned.
6	CreatedAt	timestamp	–	NN	Created timestamp.
7	UpdatedAt	timestamp	–	NN	Updated timestamp.
8	IsActive	boolean	–	NN	Is the record valid?

2.21 DailyLimit

Description: Records per-user daily action limits and usage for a given action type and date, including the count and creation time.

No	Field	Data Type	Data Size	Constraint	Description
1	LimitId	integer	–	PK	Primary key.
2	UserId	integer	–	FK	References User table.
3	ActionType	Character varying	100	NN	Type of action.
4	ActionDate	date	–	NN	Date apply limit.
5	Count	integer	–	NN	Number of times the action was performed during the day.
6	CreatedAt	timestamp	–	NN	Record creation time.

2.22 ChatExpert

Description: Represents a conversation between a user and an expert — references the expert and user and contains creation/update timestamps.

No	Field	Data Type	Data Size	Constraint	Description
1	ChatExpertId	integer	–	PK	Primary key.
2	ExpertId	integer	–	FK	References ChatExpertConfirmation table.
3	UserId	integer	–	FK	References User table.

4	CreatedAt	timestamp	–	NN	Created timestamp.
5	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.23 ChatExpertContent

Description: Contains messages within a ChatExpert conversation — message text, sender, optional links to expert/AI records, and created/updated timestamps.

No	Field	Data Type	Data Size	Constraint	Description
1	ContentId	integer	–	PK	Primary key.
2	ChatExpertId	integer	–	FK	References ChatExpert table.
3	FromId	integer	–	FK	References User table.
4	Message	text	–	NN	Message content.
5	ExpertId	integer	–	FK	References ExpertConfirmation table.
6	UserId	integer	–	NN	References User table.
7	ChatAid	integer	–	FK	References ChatAi table.
8	CreateAt	timestamp	–	NN	Created timestamp.
9	UpdatedAt	timestamp	–	NN	Updated timestamp.

2.24 BadWord

Description: This table stores bad words/phrases for content moderation and filtering.

No	Field	Data Type	Data Size	Constraint	Description
1	BadWordId	serial	–	PK	Primary key, auto-increment.
2	Word	varchar	200	NN	The bad word or phrase to filter.
3	IsRegex	boolean	–	–	Whether the word is a regex pattern. Default: FALSE.
4	Level	int	–	NN	Severity level (1 = low, 2 = medium, 3 = high).
5	Category	varchar	50	–	Category of the bad word.
6	IsActive	boolean	–	–	Whether this filter is active. Default: TRUE.

7	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
8	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.25 Policy

Description: This table stores policy information (Terms of Service, Privacy Policy, etc.).

No	Field	Data Type	Data Size	Constraint	Description
1	PolicyId	serial	–	PK	Primary key, auto-increment.
2	PolicyCode	varchar	50	NN	Unique code for the policy.
3	PolicyName	varchar	200	NN	Name of the policy.
4	Description	text	–	–	Description of the policy.
5	DisplayOrder	integer	–	–	Order for displaying policies. Default: 0.
6	RequireConsent	boolean	–	–	Whether user consent is required. Default: TRUE.
7	IsActive	boolean	–	–	Whether this policy is active. Default: TRUE.
8	IsDeleted	boolean	–	–	Soft delete flag. Default: FALSE.
9	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
10	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.26 PolicyVersion

Description: This table stores version history of each policy.

No	Field	Data Type	Data Size	Constraint	Description
1	PolicyVersionId	serial	–	PK	Primary key, auto-increment.
2	PolicyId	integer	–	FK	Reference to Policy table. On Delete Cascade.
3	VersionNumber	integer	–	NN	Version number of the policy.
4	Title	varchar	300	NN	Title of this version.
5	Content	text	–	NN	Full content of the policy.

6	ChangeLog	text	–	–	Description of changes in this version.
7	Status	varchar	20	NN	Status: DRAFT, ACTIVE, INACTIVE. Default: DRAFT.
8	PublishedAt	timestamp	–	–	When this version was published.
9	DeactivatedAt	timestamp	–	–	When this version was deactivated.
10	CreatedByUserId	integer	–	FK	User who created this version. On Delete Set Null.
11	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
12	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.27 UserPolicyAccept

Description: This table stores user acceptance history for policy versions.

No	Field	Data Type	Data Size	Constraint	Description
1	AcceptId	bigserial	–	PK	Primary key, auto-increment.
2	UserId	integer	–	FK	Reference to User table. On Delete Cascade.
3	PolicyVersionId	integer	–	FK	Reference to PolicyVersion table. On Delete Cascade.
4	AcceptedAt	timestamp	–	NN	When user accepted the policy. Default: NOW().
5	IsValid	boolean	–	–	Whether this acceptance is still valid. Default: TRUE.
6	InvalidatedAt	timestamp	–	–	When this acceptance was invalidated.
7	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().

2.28 PetAppointmentLocation

Description: This table stores pet-friendly locations for appointments.

No	Field	Data Type	Data Size	Constraint	Description
1	LocationId	serial	–	PK	Primary key, auto-increment.

2	Name	varchar	200	NN	Name of the location.
3	Address	text	–	NN	Full address of the location.
4	Latitude	decimal	(9,6)	NN	GPS latitude coordinate.
5	Longitude	decimal	(9,6)	NN	GPS longitude coordinate.
6	City	varchar	100	–	City name.
7	District	varchar	100	–	District name.
8	IsPetFriendly	boolean	–	–	Whether location is pet-friendly. Default: TRUE.
9	PlaceType	varchar	50	–	Type: park, pet_cafe, vet_clinic, custom.
10	GooglePlaceId	varchar	255	–	Google Maps Place ID.
11	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
12	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.29 PetAppointment

Description: This table stores pet meeting appointments between matched users.

No	Field	Data Type	Data Size	Constraint	Description
1	AppointmentId	serial	–	PK	Primary key, auto-increment.
2	MatchId	int	–	FK	Reference to ChatUser (match).
3	InviterPetId	int	–	FK	Pet ID of the inviter.
4	InviteePetId	int	–	FK	Pet ID of the invitee.
5	InviterUserId	int	–	FK	User ID of the inviter.
6	InviteeUserId	int	–	FK	User ID of the invitee.
7	AppointmentDate Time	timestamp	–	NN	Scheduled date and time.
8	LocationId	int	–	FK	Reference to PetAppointmentLocation.
9	ActivityType	varchar	50	NN	Type: walk, cafe, playdate.

10	Status	varchar	30	–	Status: pending, confirmed, rejected, cancelled, on_going, completed, no_show. Default: pending.
11	CurrentDecisionUserld	int	–	FK	User who needs to make decision.
12	CounterOfferCount	int	–	–	Number of counter offers (max 3). Default: 0.
13	InviterCheckedIn	boolean	–	–	Whether inviter has checked in. Default: FALSE.
14	InviteeCheckedIn	boolean	–	–	Whether invitee has checked in. Default: FALSE.
15	InviterCheckInTime	timestamp	–	–	Inviter check-in timestamp.
16	InviteeCheckInTime	timestamp	–	–	Invitee check-in timestamp.
17	CancelledBy	int	–	FK	User who cancelled the appointment.
18	CancelReason	text	–	–	Reason for cancellation.
19	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
20	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.30 PetEvent

Description: This table stores online pet photo/video competition events.

No	Field	Data Type	Data Size	Constraint	Description
1	EventId	serial	–	PK	Primary key, auto-increment.
2	Title	varchar	200	NN	Event title.
3	Description	text	–	–	Event description.
4	CoverImageUrl	varchar	500	–	Cover/poster image URL.
5	StartTime	timestamp	–	NN	Event start time.

6	SubmissionDeadline	timestamp	–	NN	Deadline for submissions.
7	EndTime	timestamp	–	NN	Event end time (voting ends).
8	Status	varchar	30	–	Status: upcoming, active, submission_closed, voting_ended, completed, cancelled. Default: upcoming.
9	PrizeDescription	text	–	–	Description of prizes.
10	PrizePoints	int	–	–	Points awarded to winners. Default: 0.
11	CreatedBy	int	–	FK	Admin who created the event.
12	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
13	UpdatedAt	timestamp	–	–	Record last update timestamp. Default: NOW().

2.31 EventSubmission

Description: This table stores user submissions for pet events.

No	Field	Data Type	Data Size	Constraint	Description
1	SubmissionId	serial	–	PK	Primary key, auto-increment.
2	EventId	int	–	FK	Reference to PetEvent.
3	UserId	int	–	FK	User who submitted.
4	PetId	int	–	FK	Pet in the submission.
5	MediaUrl	varchar	500	NN	URL of submitted media.
6	MediaType	varchar	20	NN	Type: image, video.
7	ThumbnailUrl	varchar	500	–	Thumbnail URL for videos.
8	Caption	varchar	500	–	Caption for the submission.
9	VoteCount	int	–	–	Total votes received. Default: 0.
10	Rank	int	–	–	Final ranking (1, 2, 3 for top 3).
11	IsWinner	boolean	–	–	Whether this is a winning entry. Default: FALSE.

12	CreatedAt	timestamp	–	–	Record creation timestamp. Default: NOW().
13	IsDeleted	boolean	–	–	Soft delete flag. Default: FALSE.

2.32 EventVote

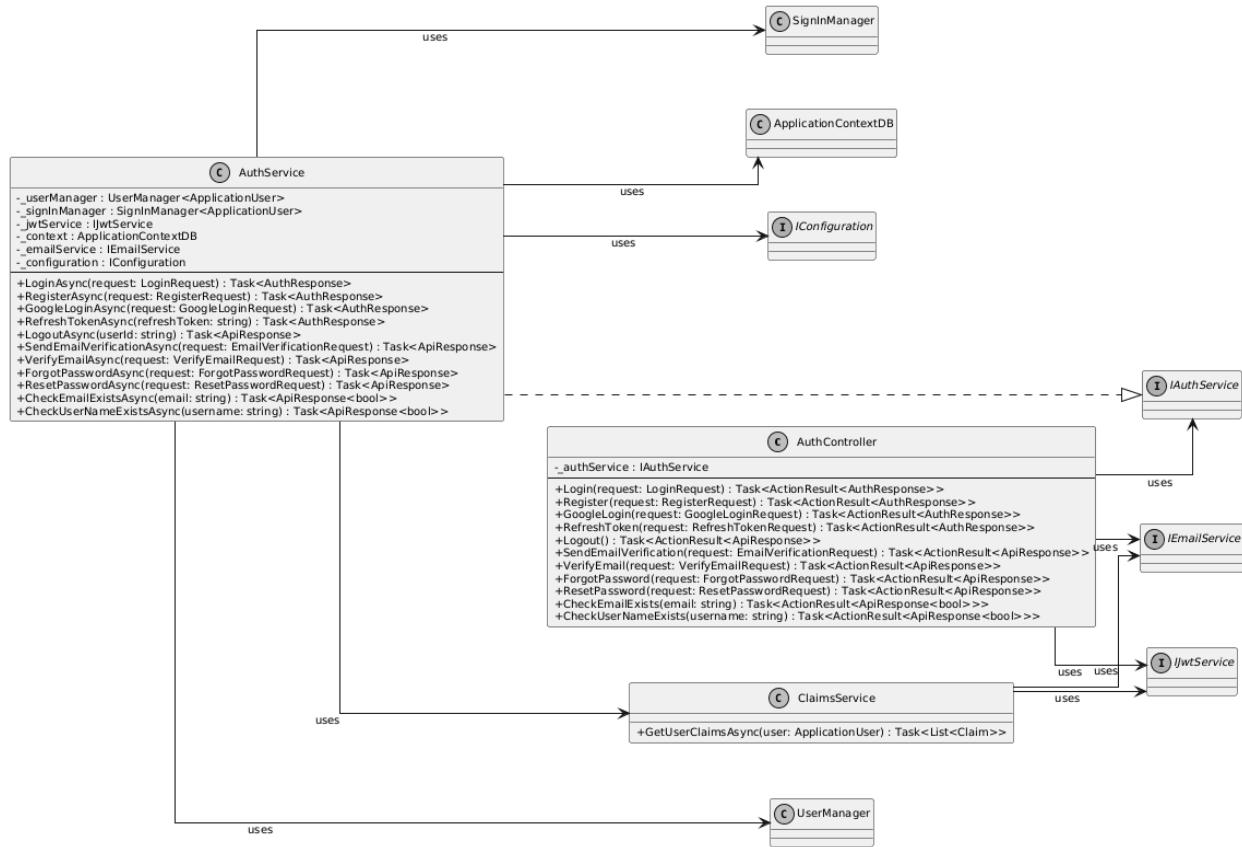
Description: This table stores votes for event submissions.

No	Field	Data Type	Data Size	Constraint	Description
1	Voteld	serial	–	PK	Primary key, auto-increment.
2	SubmissionId	int	–	FK	Reference to EventSubmission. On Delete Cascade.
3	UserId	int	–	FK	User who voted.
4	CreatedAt	timestamp	–	–	Vote timestamp. Default: NOW().

3. Detailed Design

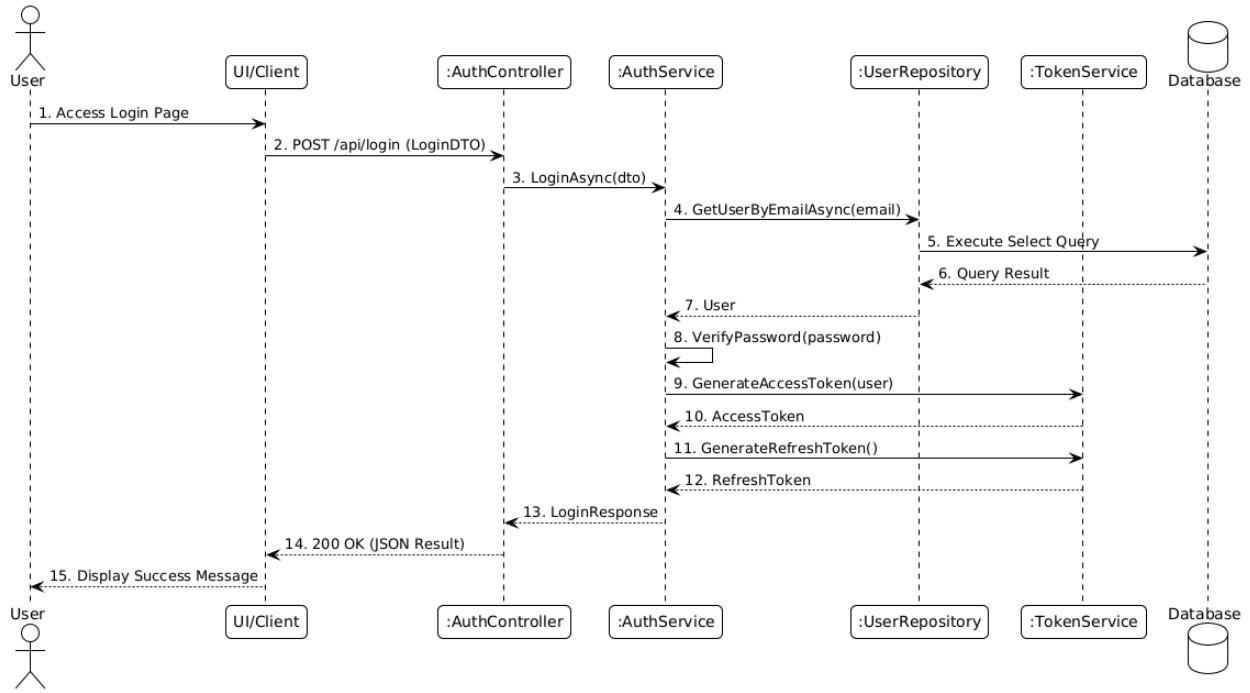
3.1 Authentication & Authorization

3.1.1 Class Diagram

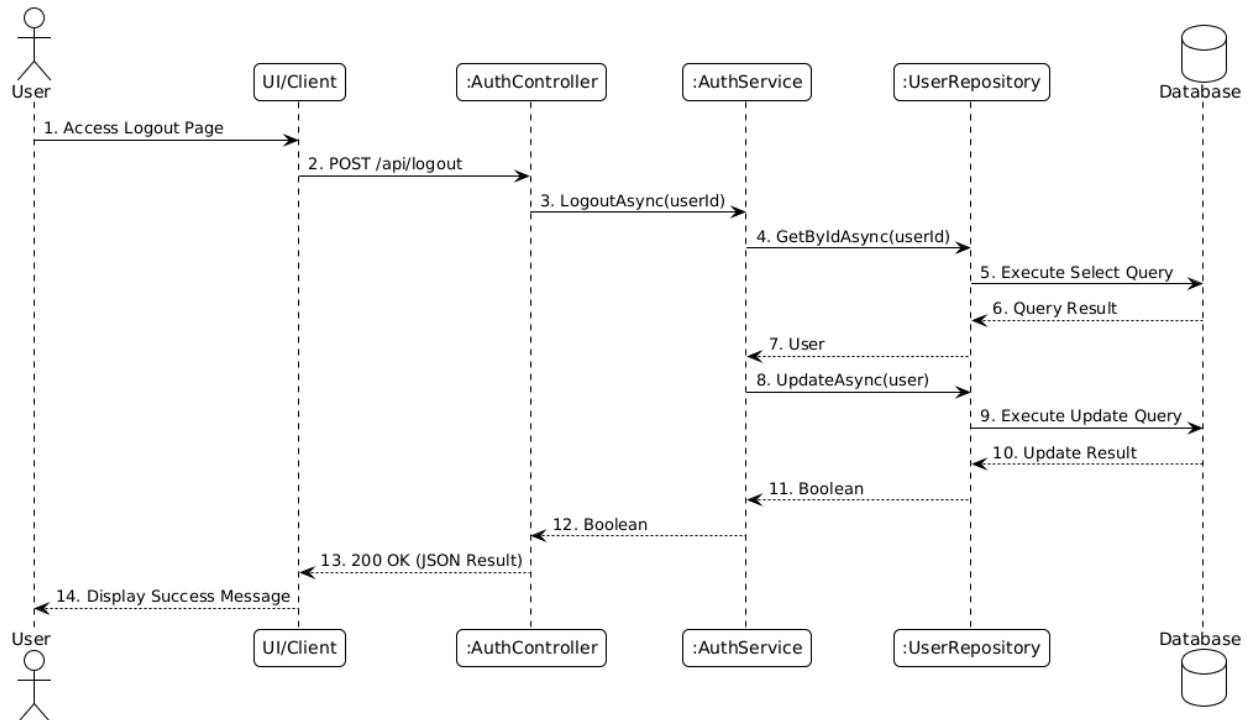


3.1.2 Sequence Diagram

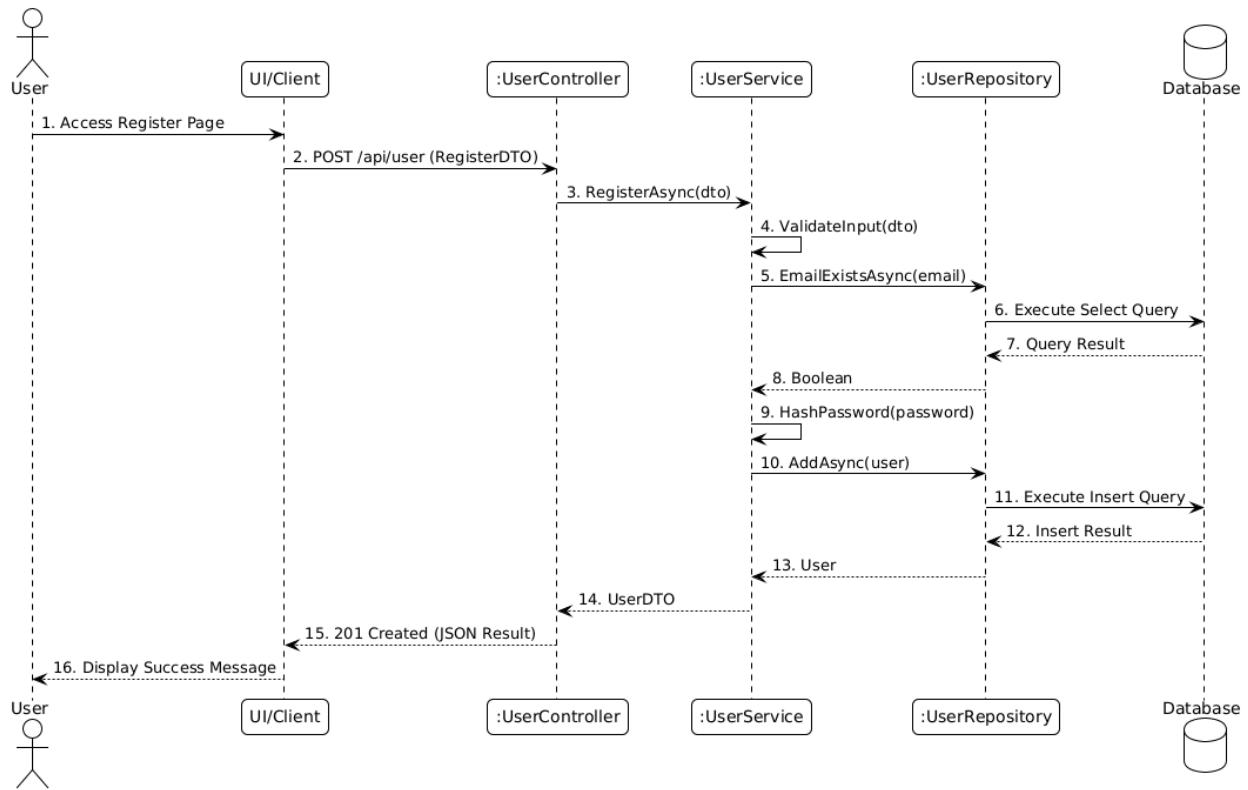
3.1.2.1 Login



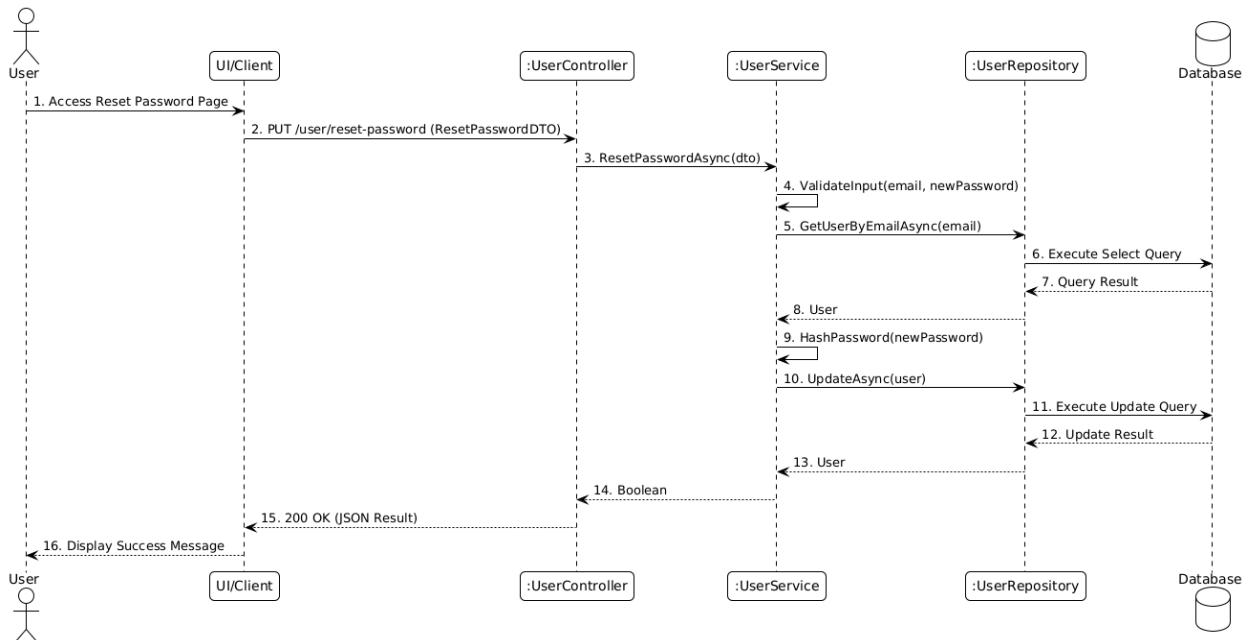
3.1.2.2 Logout



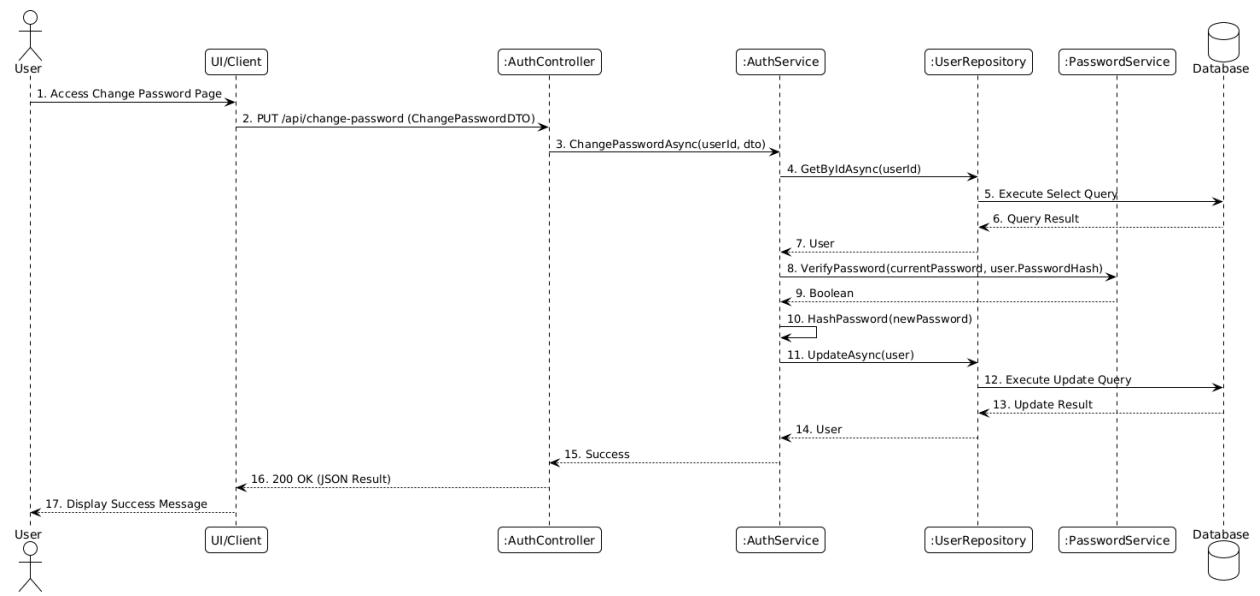
3.1.2.3 Register Account



3.1.2.4 Reset Password

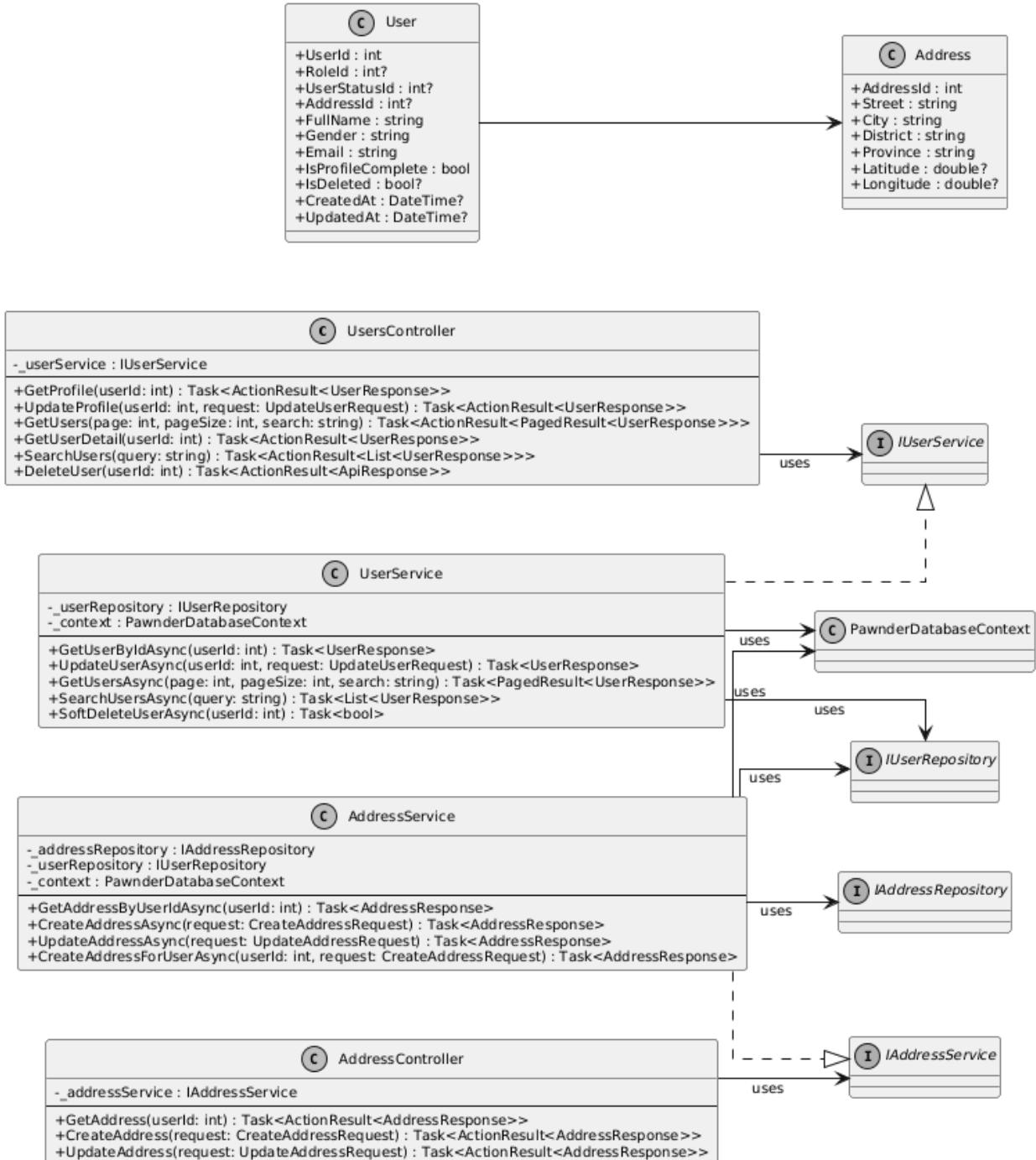


3.1.2.5 Change Password



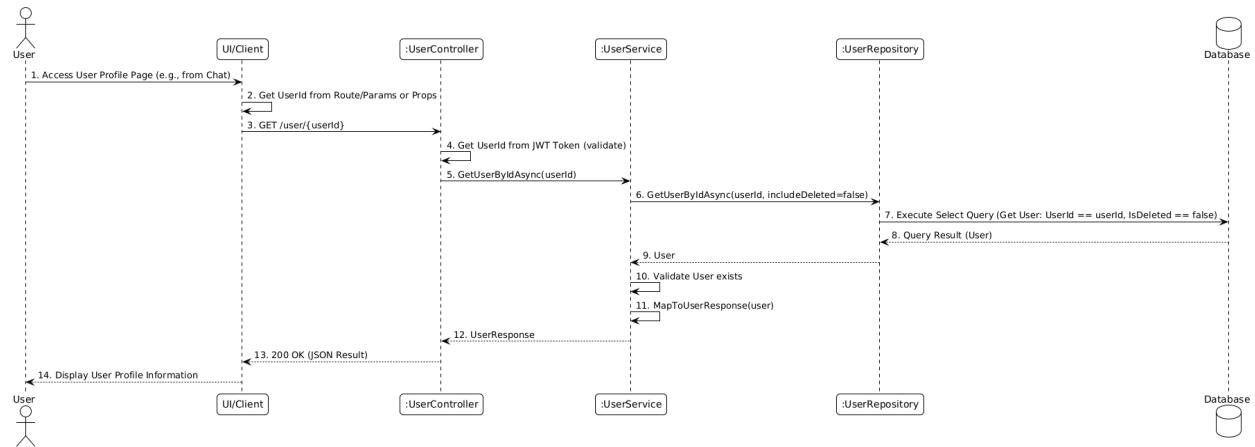
3.2 User Profile Management

3.2.1 Class Diagram

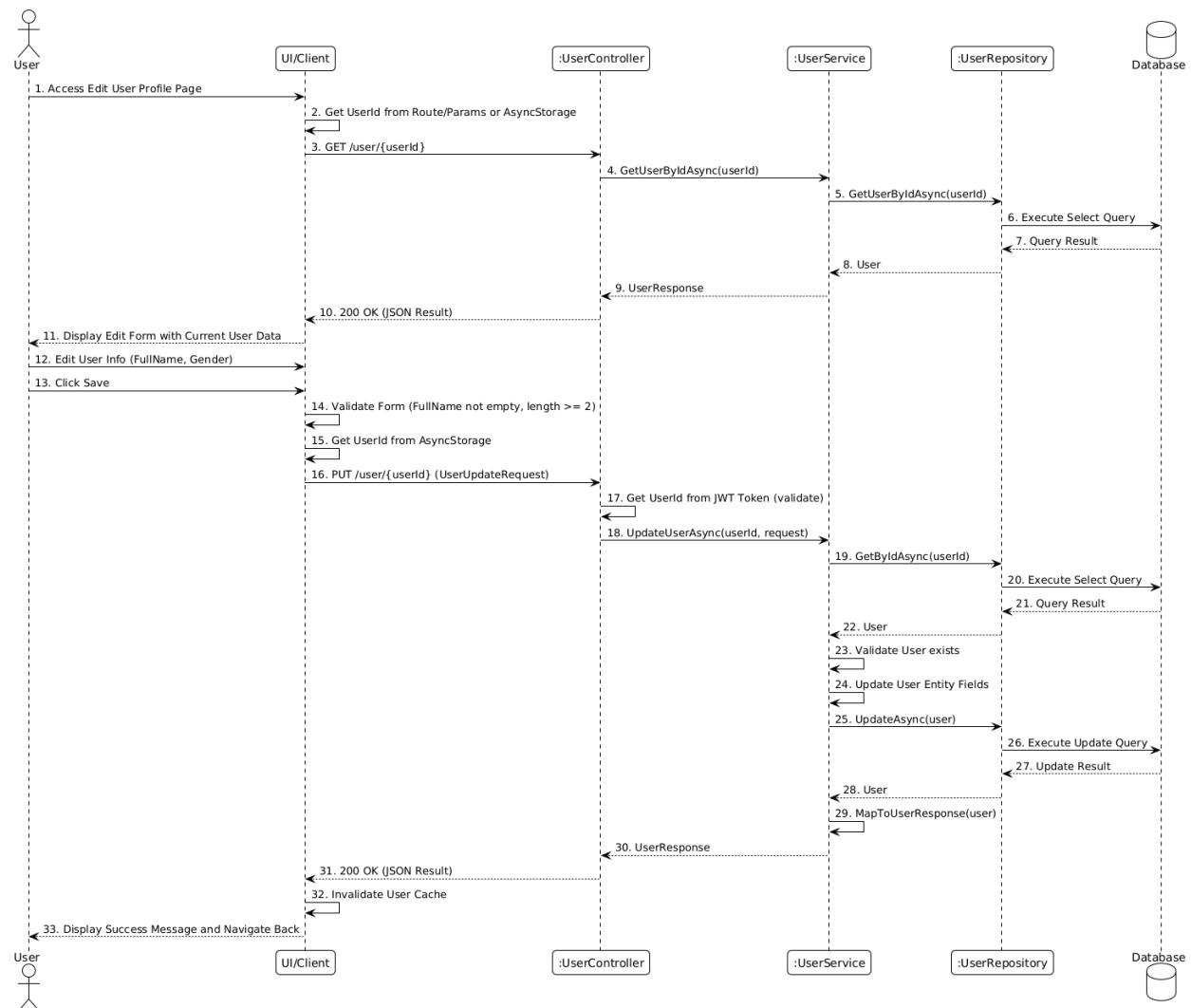


3.2.2 Sequence Diagram

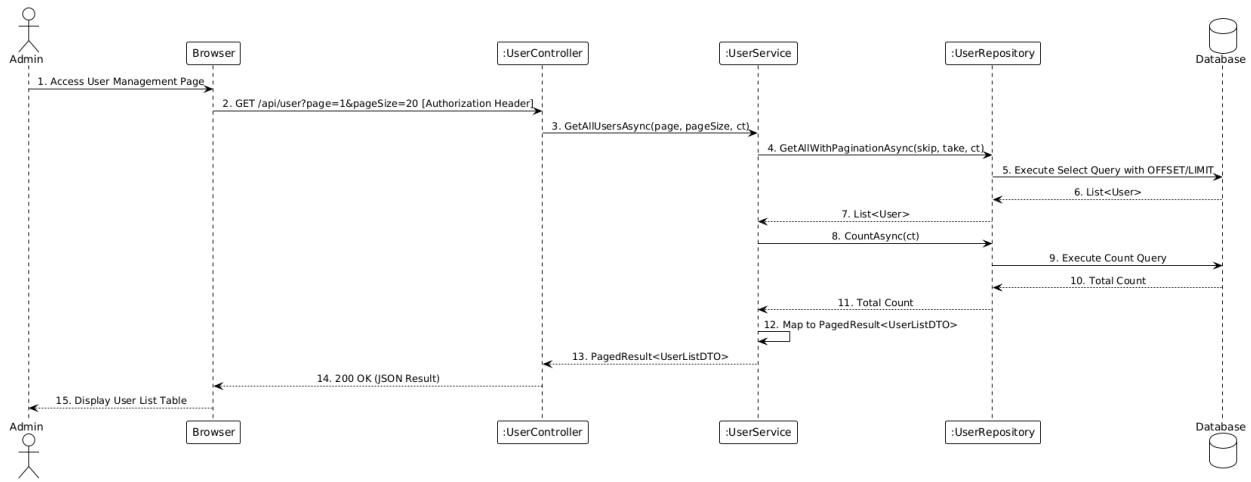
3.2.2.1 View User Profile



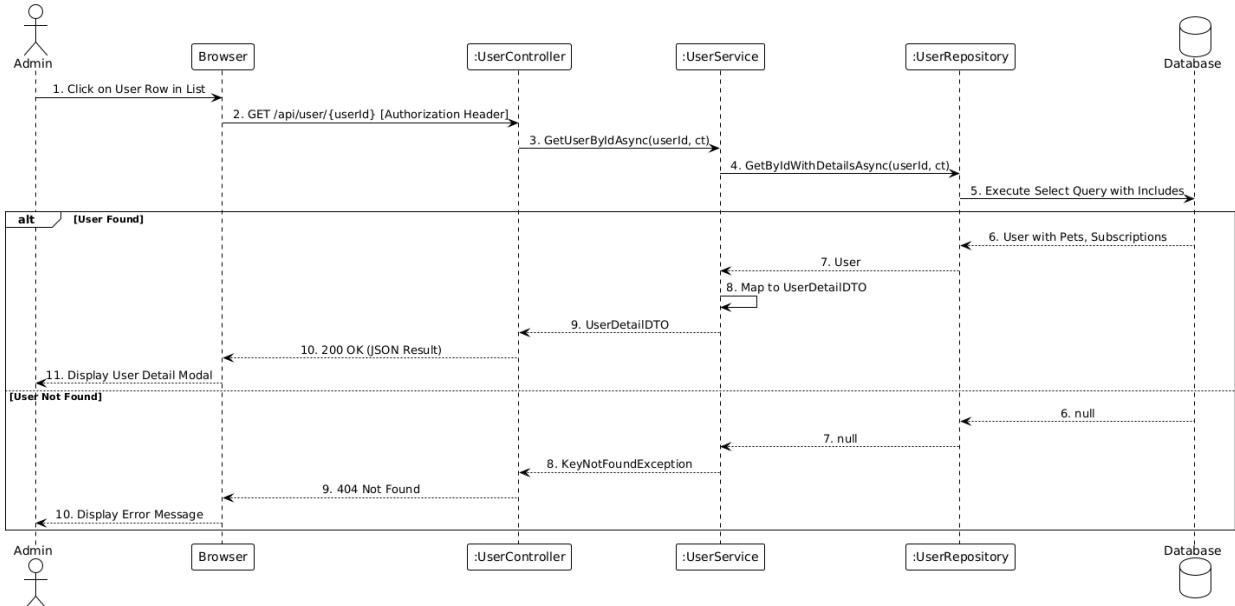
3.2.2.2 Edit User Profile

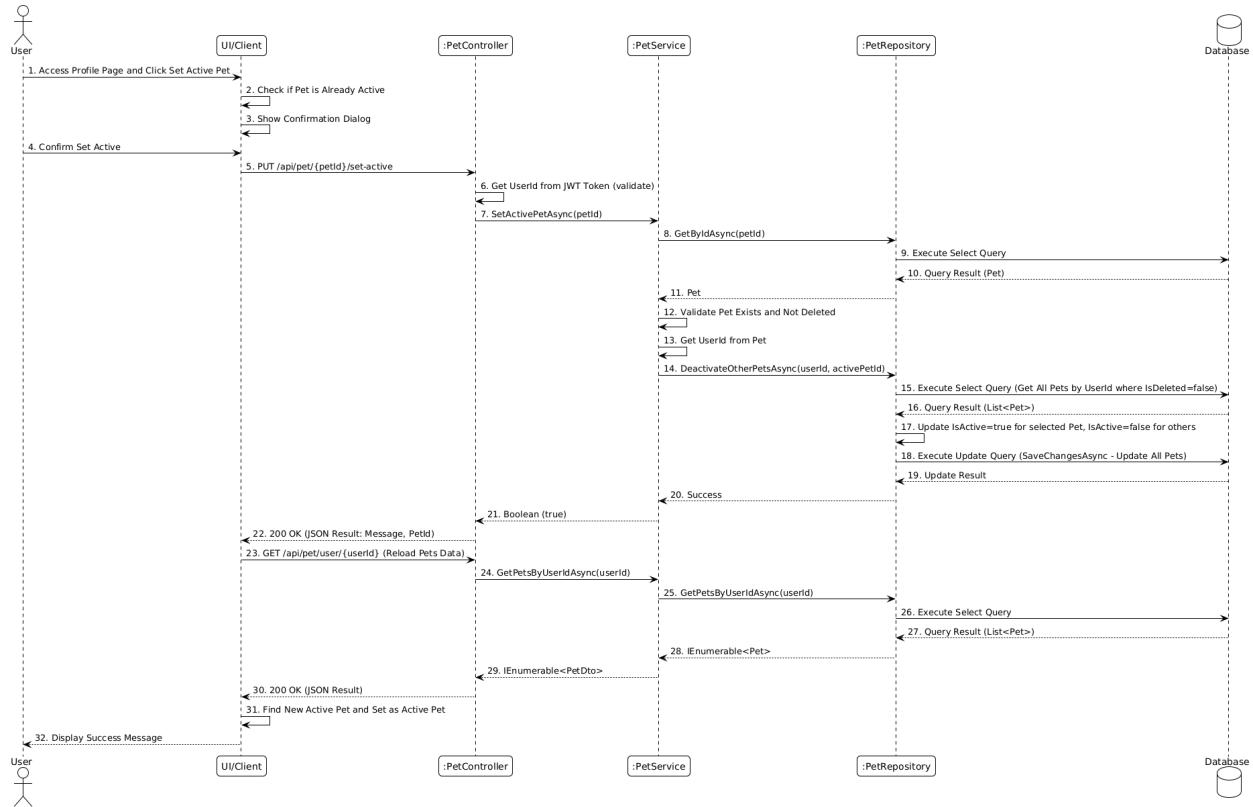


3.2.2.3 View User List



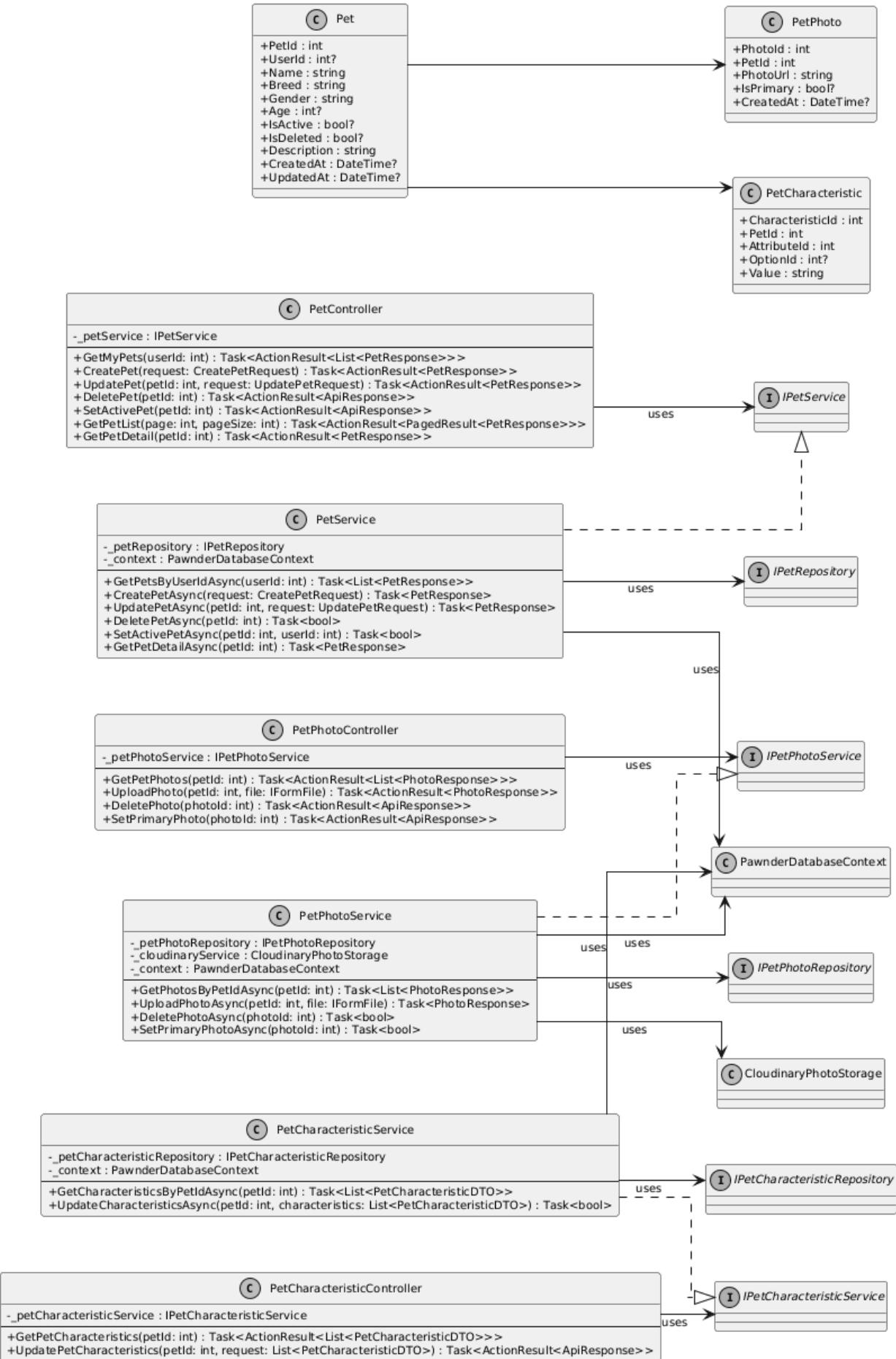
3.2.2.4 View User Detail





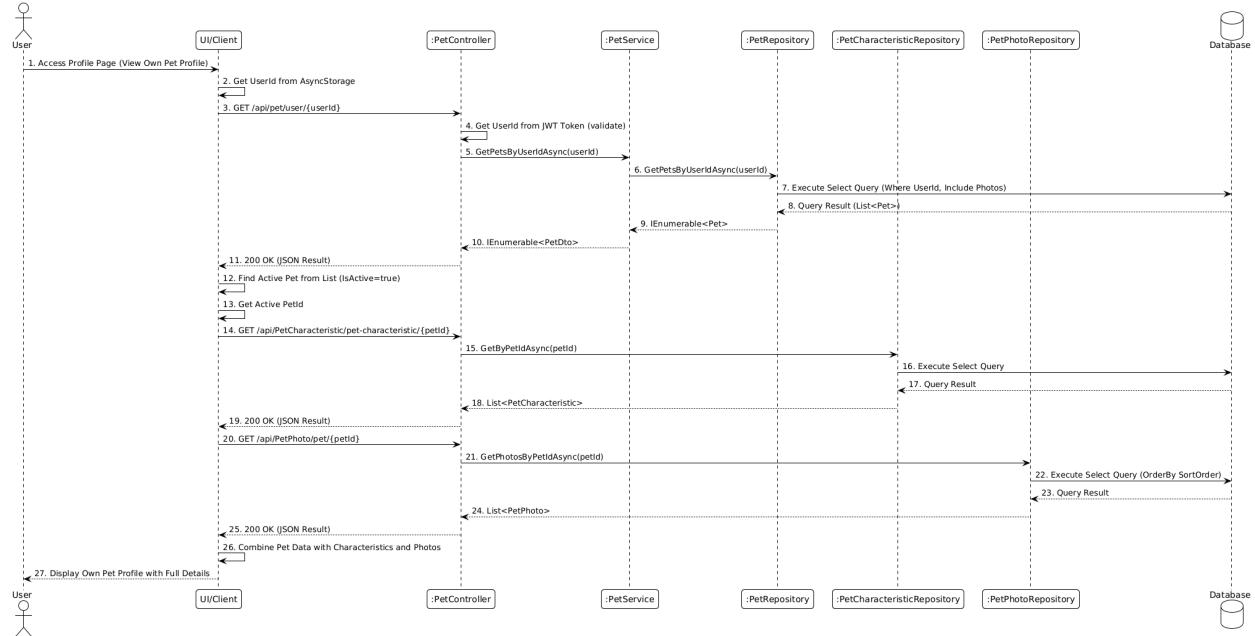
3.3 Pet Profile Management

3.3.1 Class Diagram

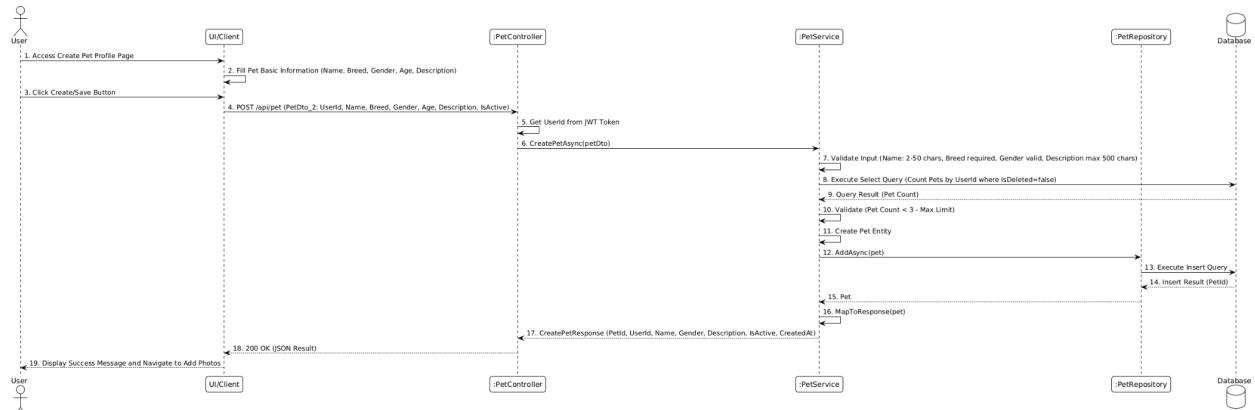


3.3.2 Sequence Diagram

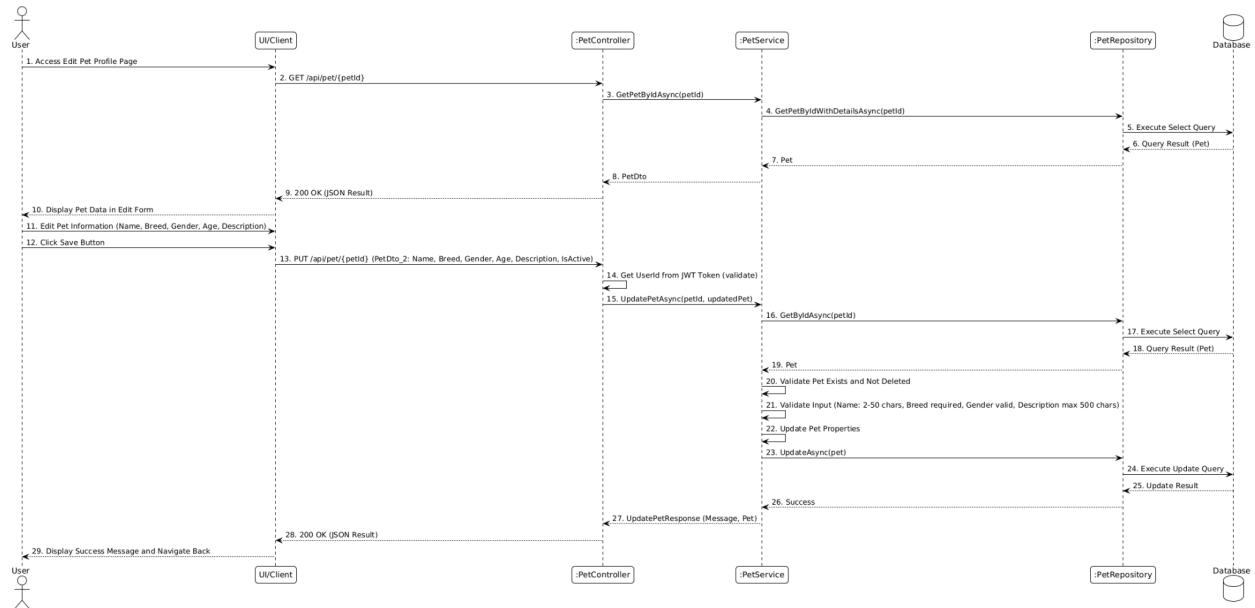
3.3.2.1 View Own Pet Profile



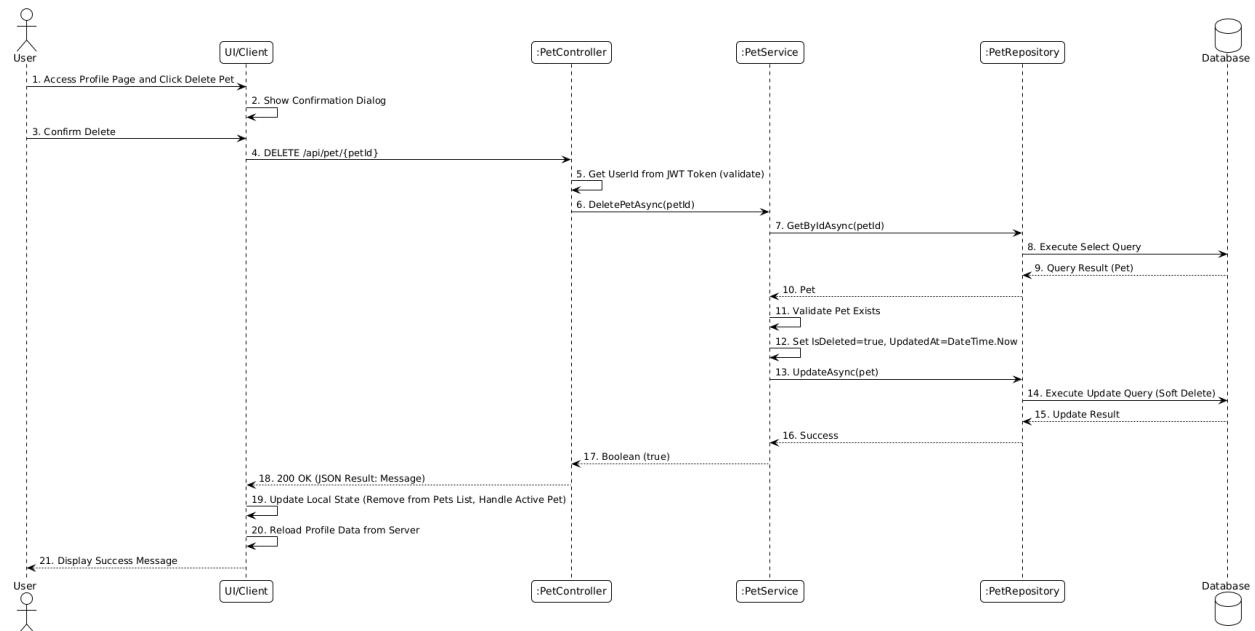
3.3.2.2 Create Pet Profile



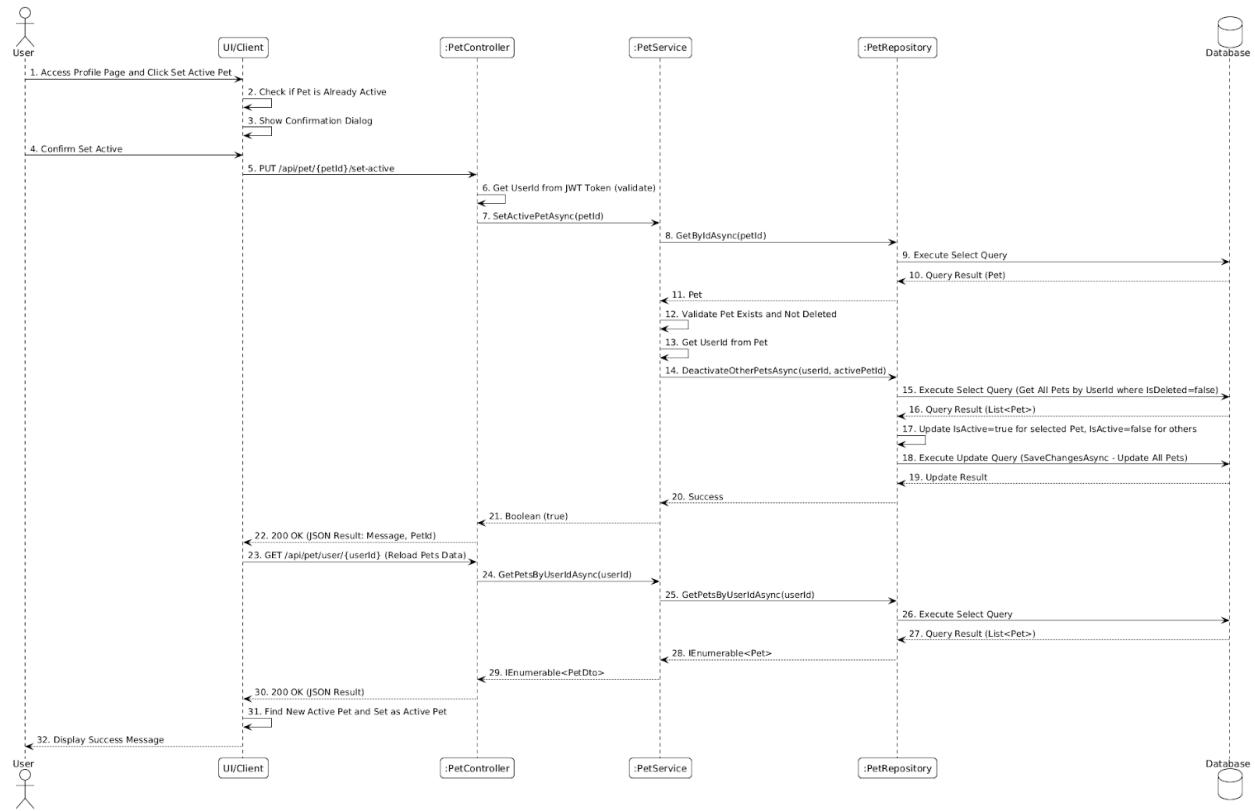
3.3.2.3 Edit Pet Profile



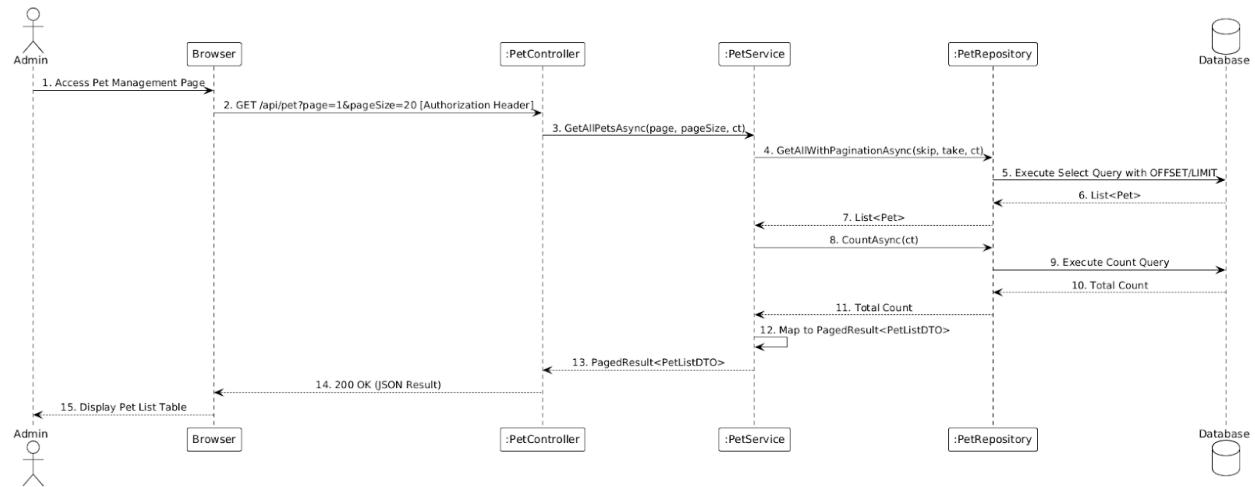
3.3.2.4 Delete Pet Profile



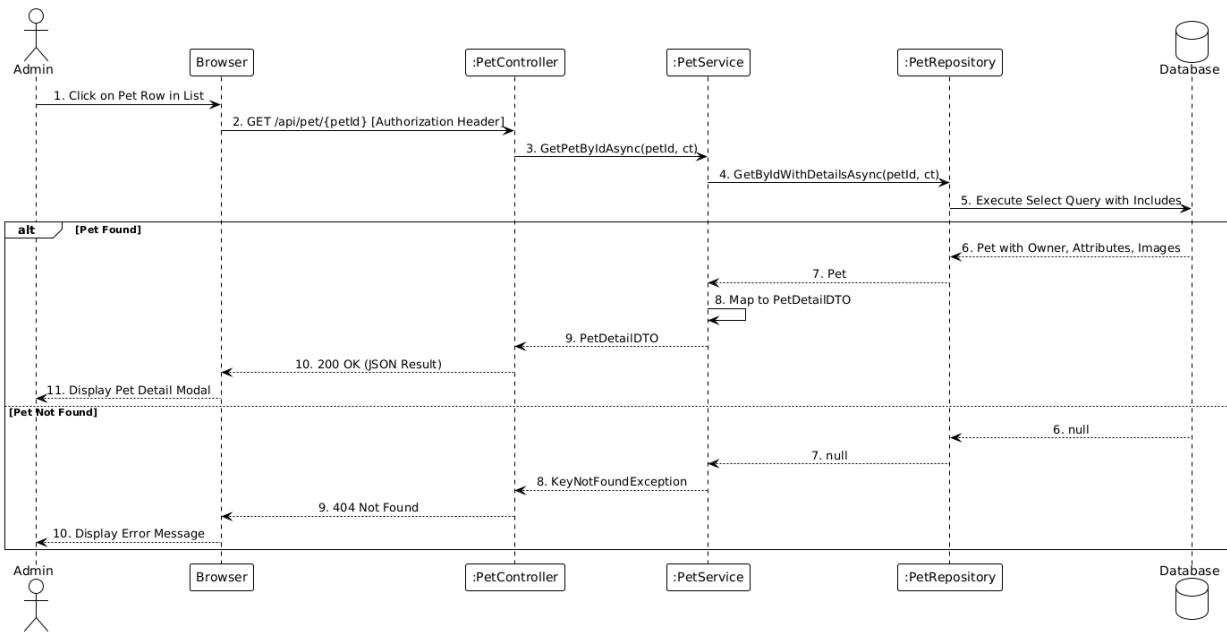
3.3.2.5 Set Active Pet Profile



3.3.2.6 View Pet List

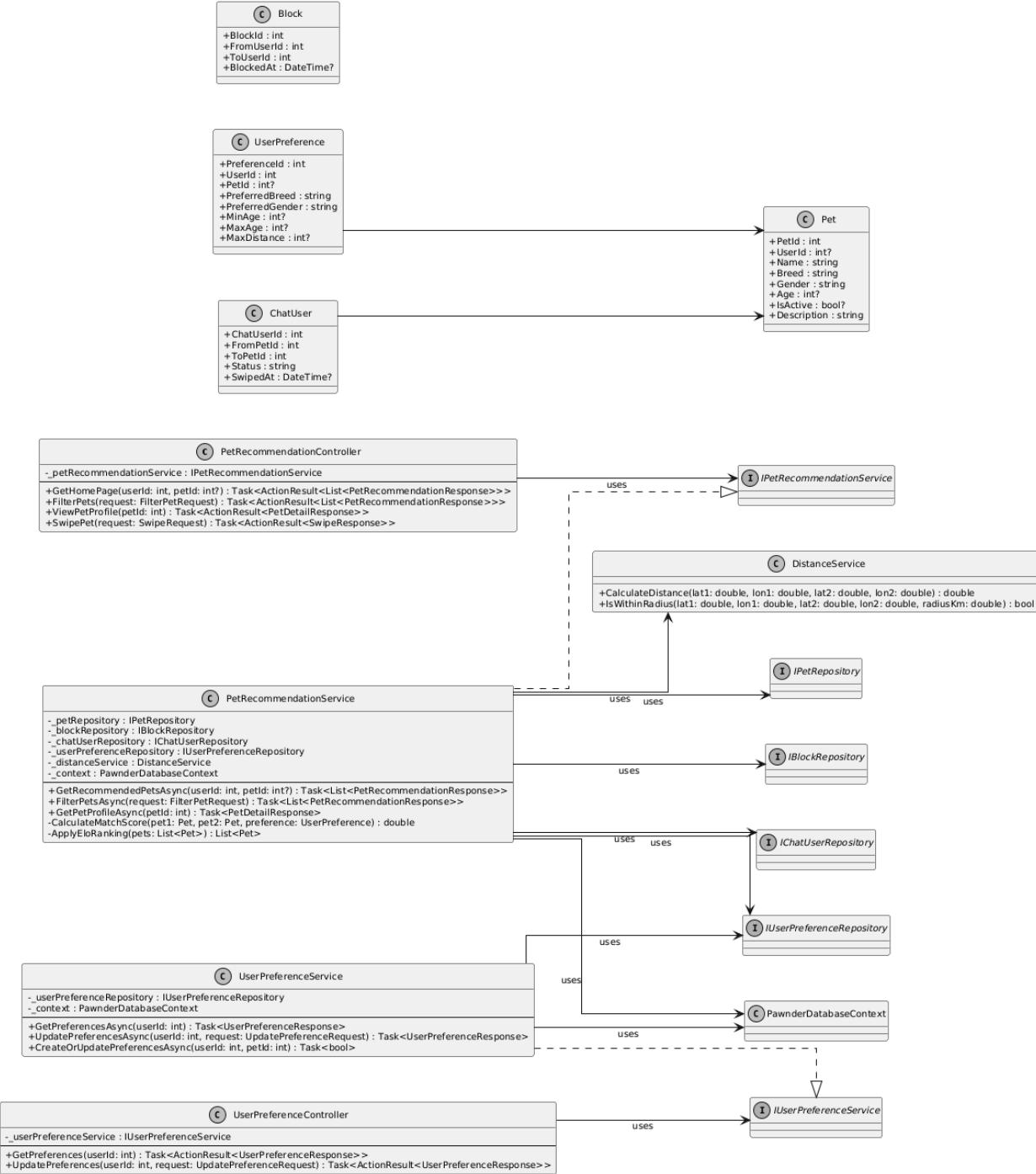


3.3.2.7 View Pet Detail



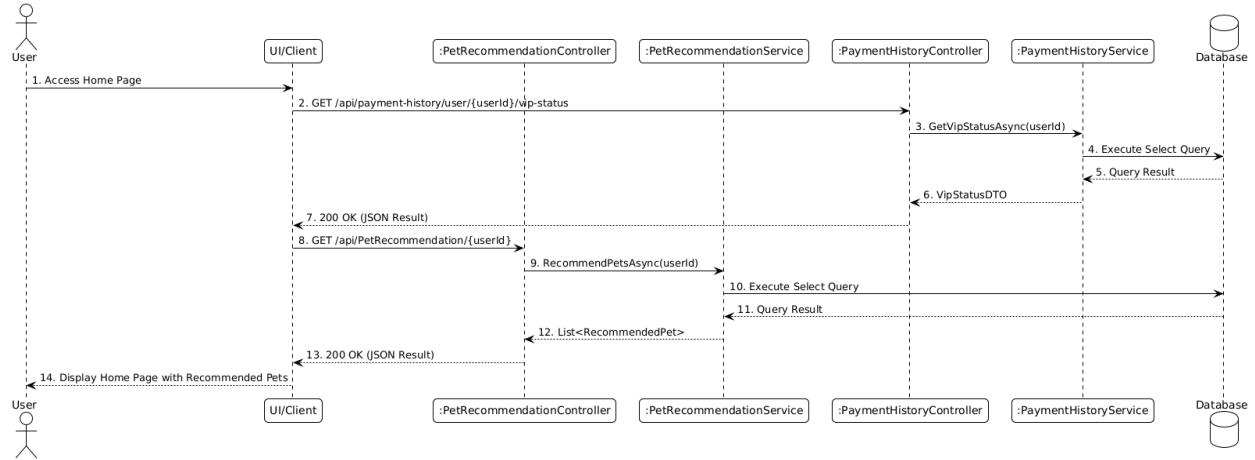
3.4 Pet Discovery & Swiping

3.4.1 Class Diagram

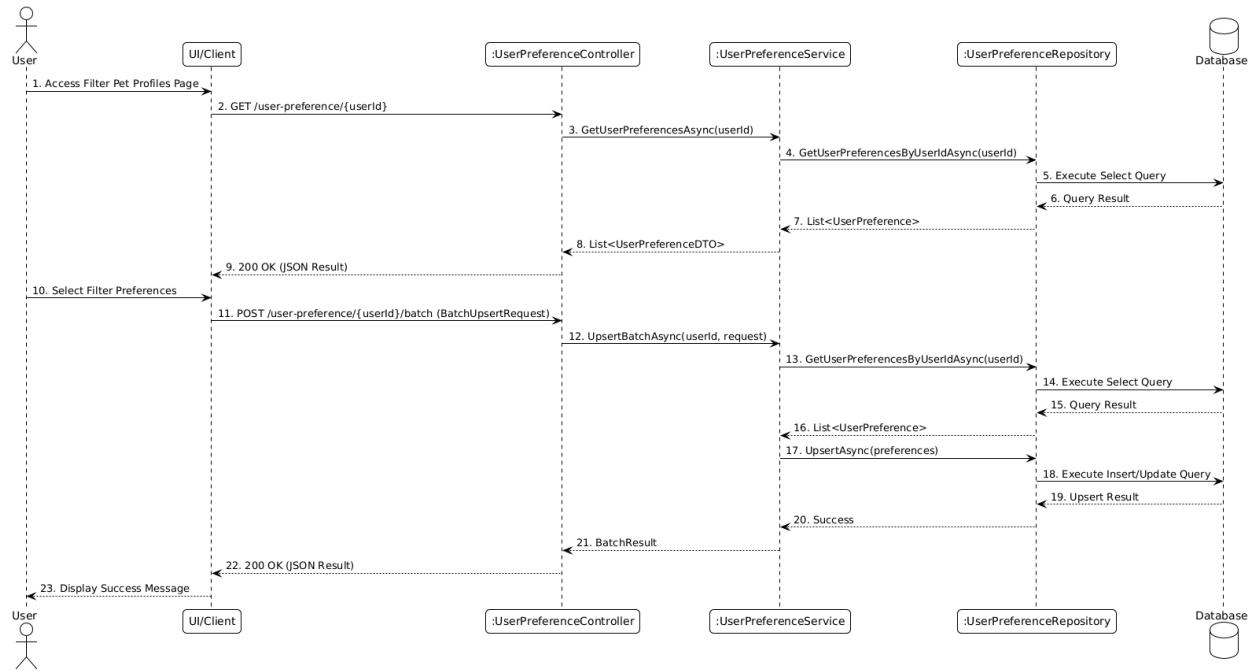


3.4.2 Sequence Diagram

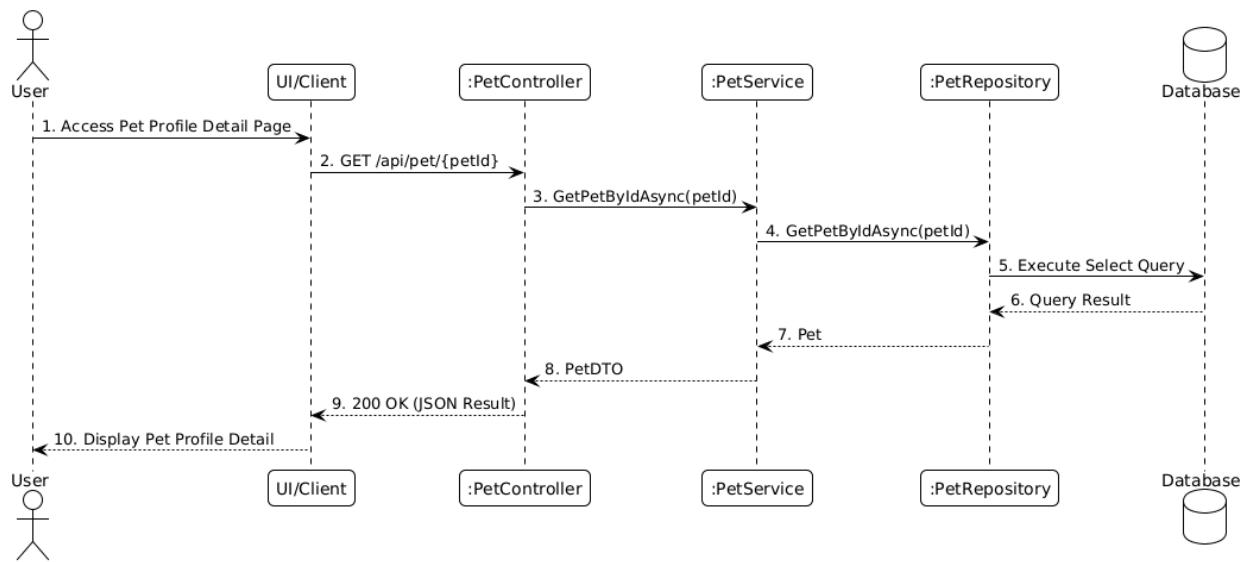
3.4.2.1 View Home Page



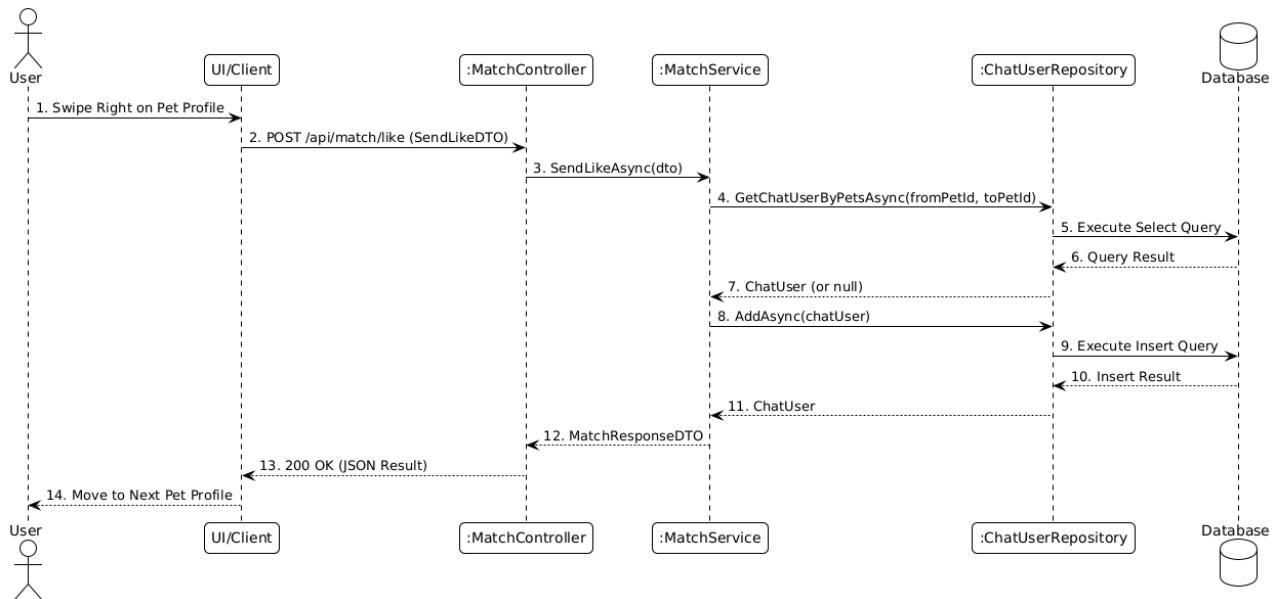
3.4.2.2 Filter Pet Profile



3.4.2.3 View Pet Profile

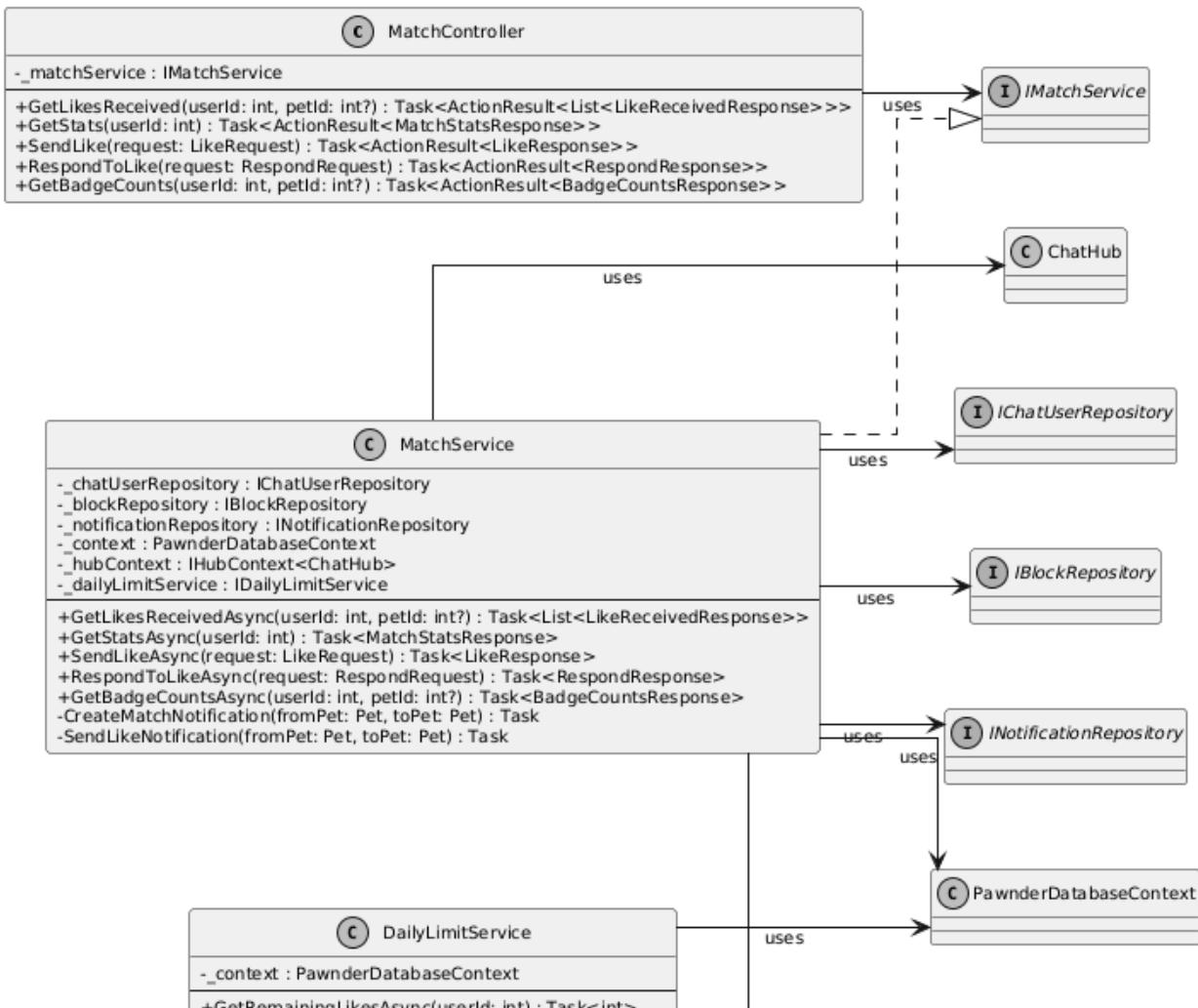
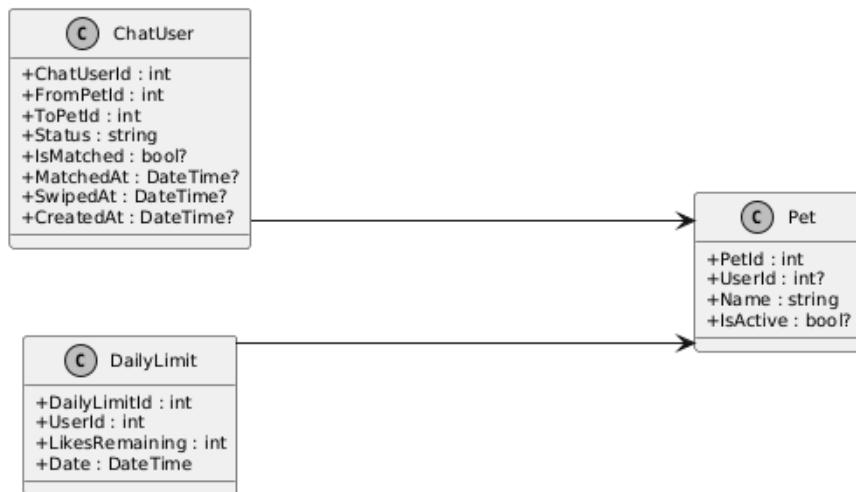
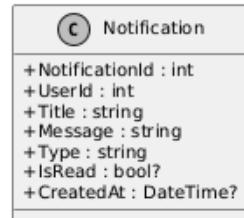


3.4.2.4 Swipe Pet Profile



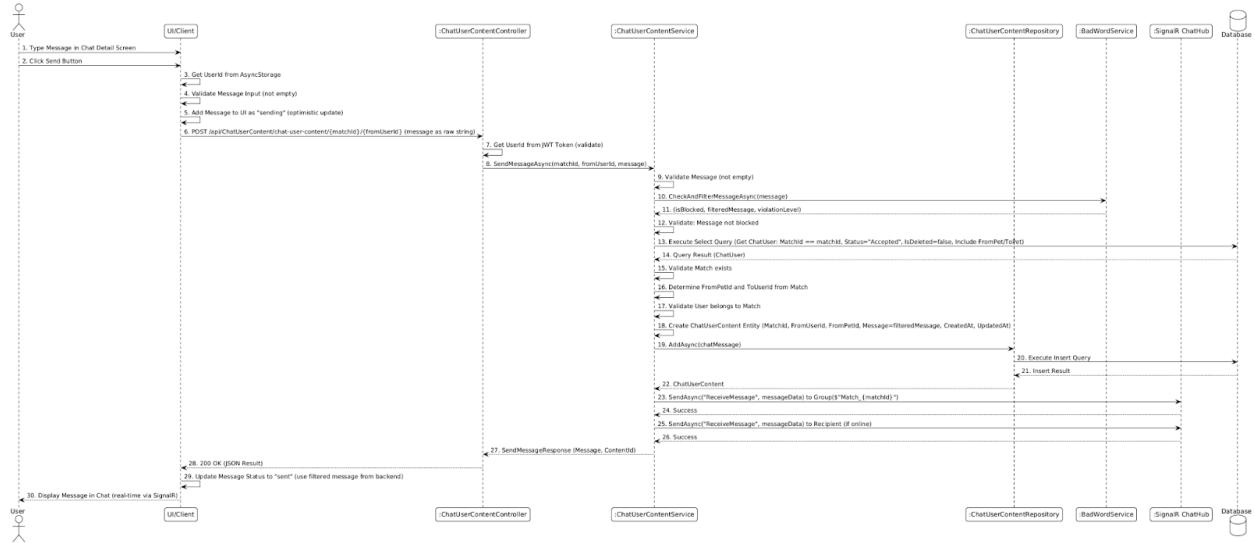
3.5 Matching

3.5.1 Class Diagram

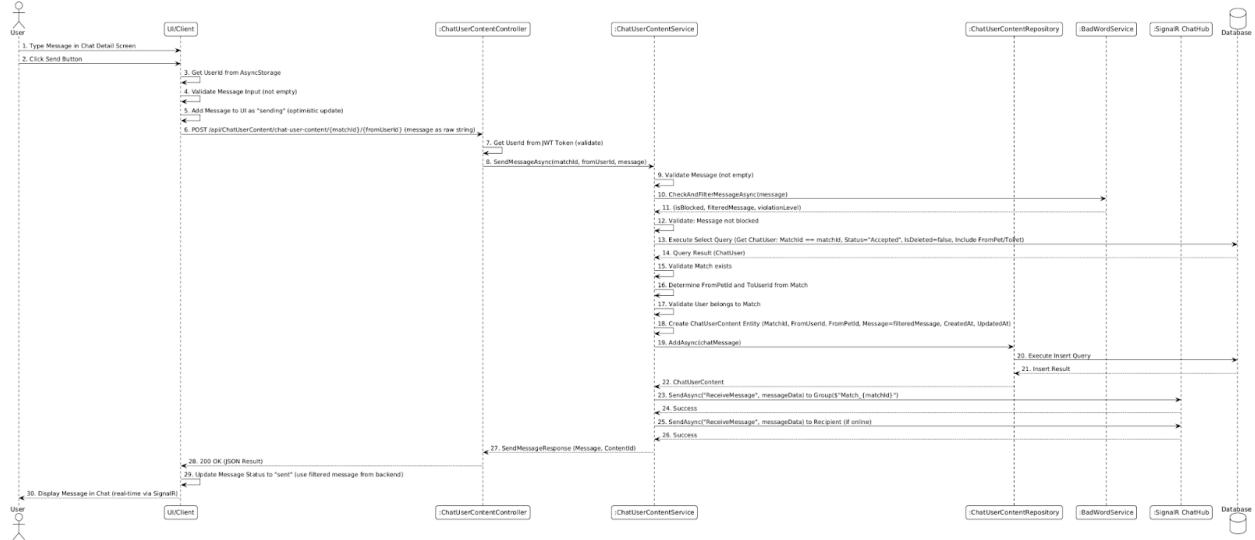


3.5.2 Sequence Diagram

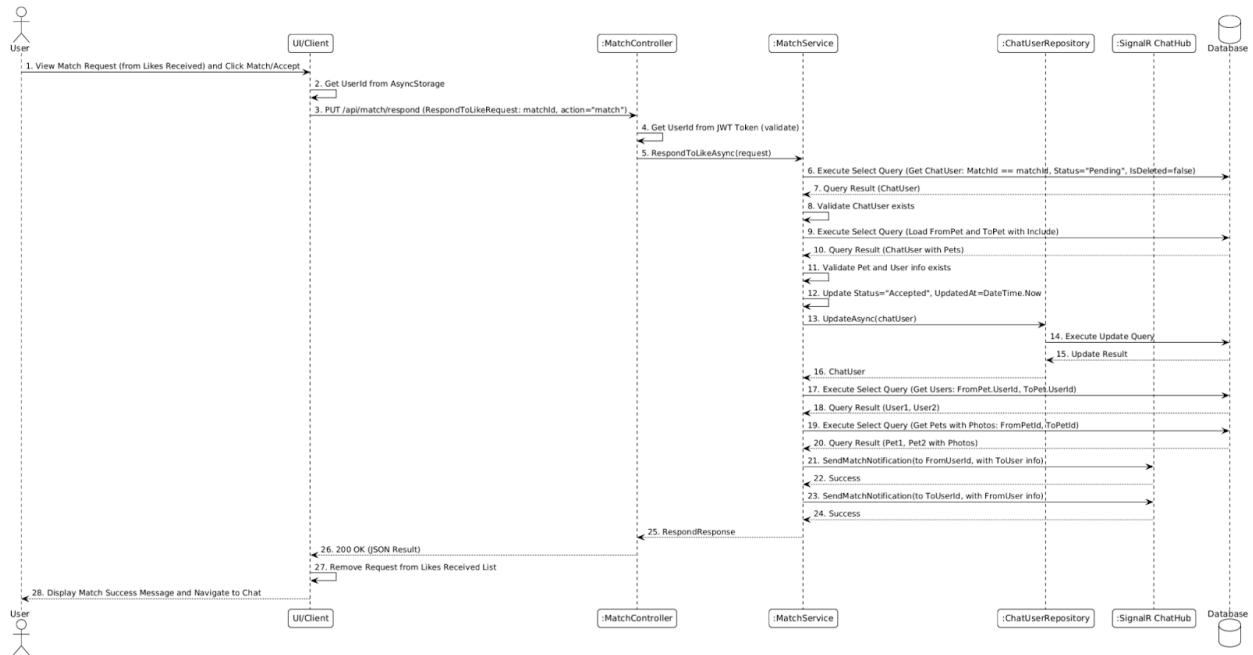
3.5.2.1 View Match List



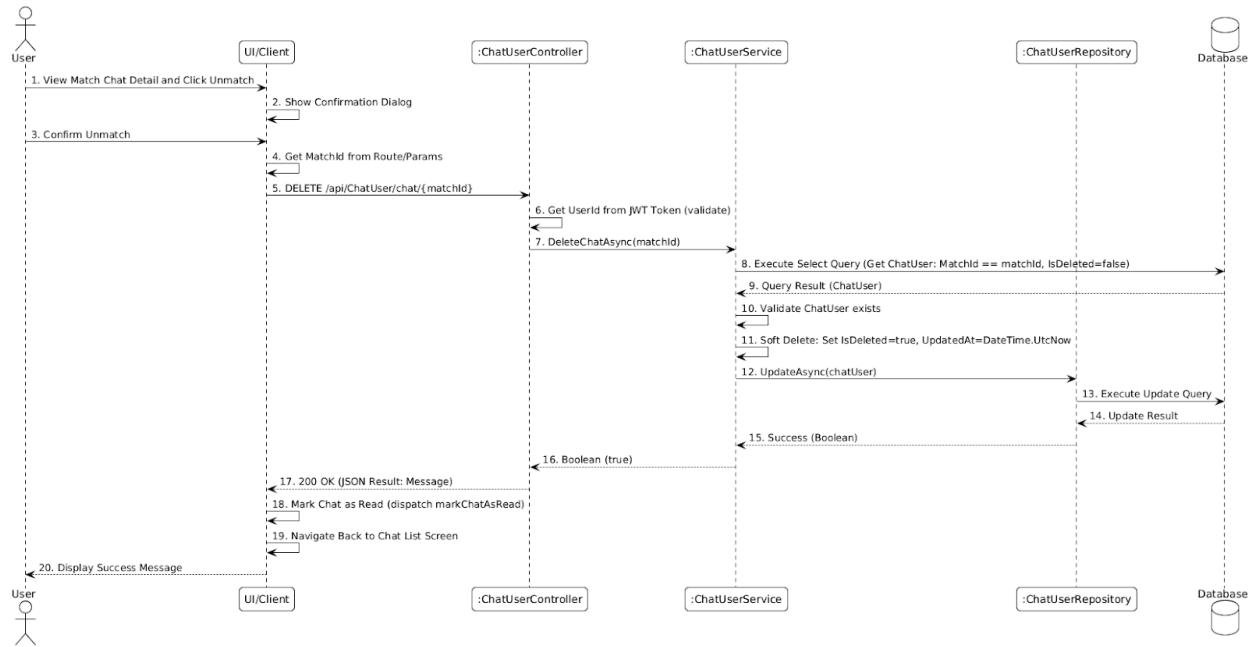
3.5.2.2 Chat with Matched User



3.5.2.3 Respond to Match Request

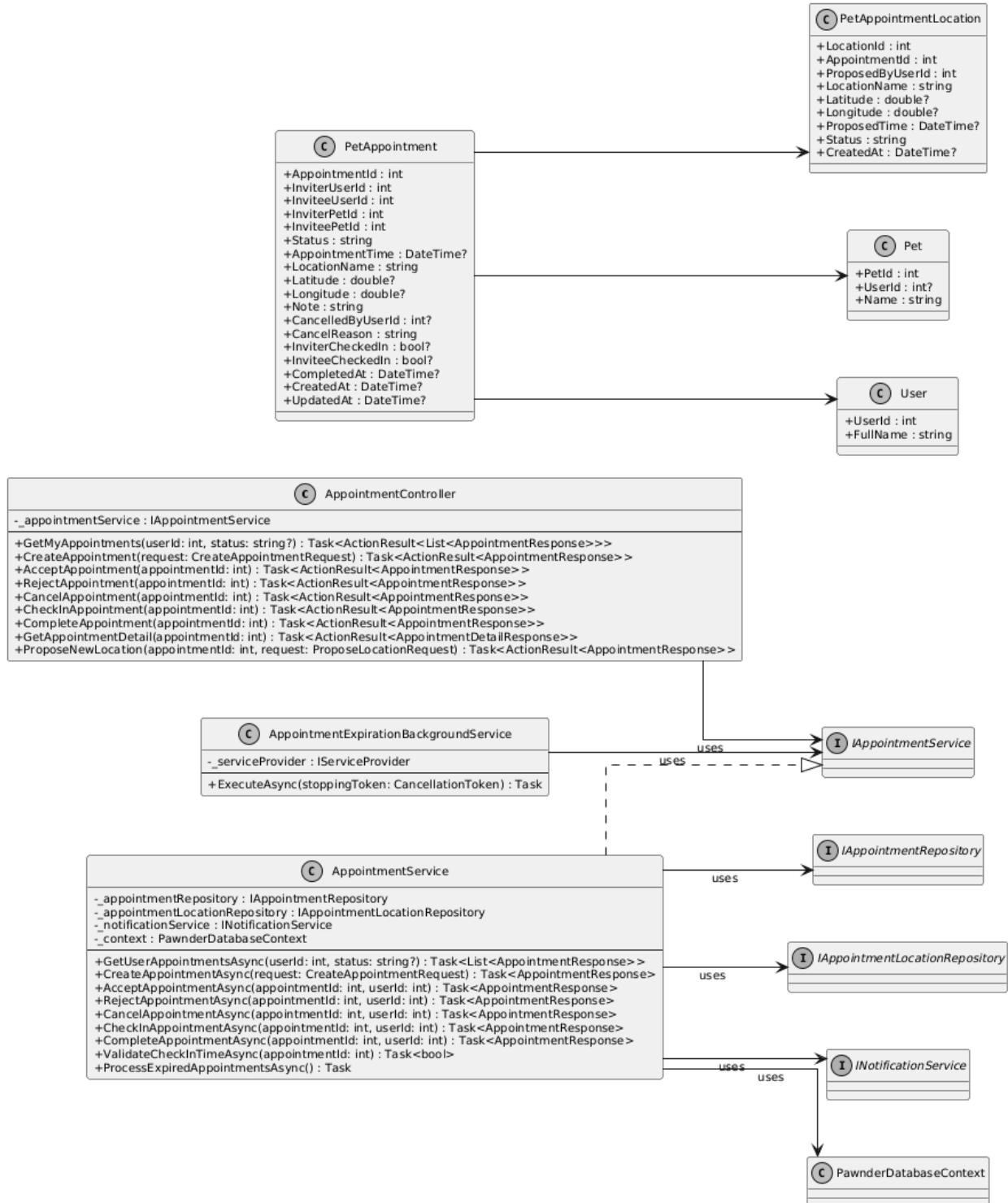


3.5.2.4 Unmatch User



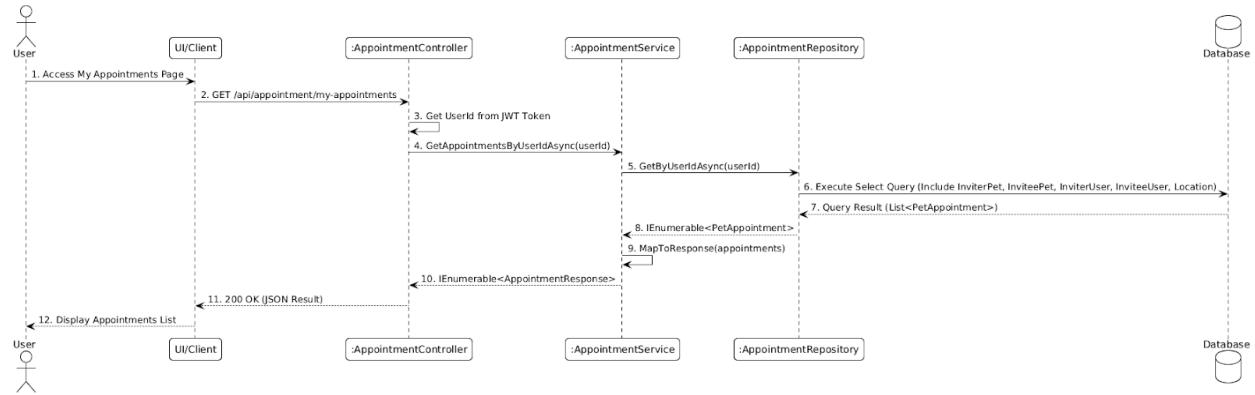
3.6 Appointment

3.6.1 Class Diagram

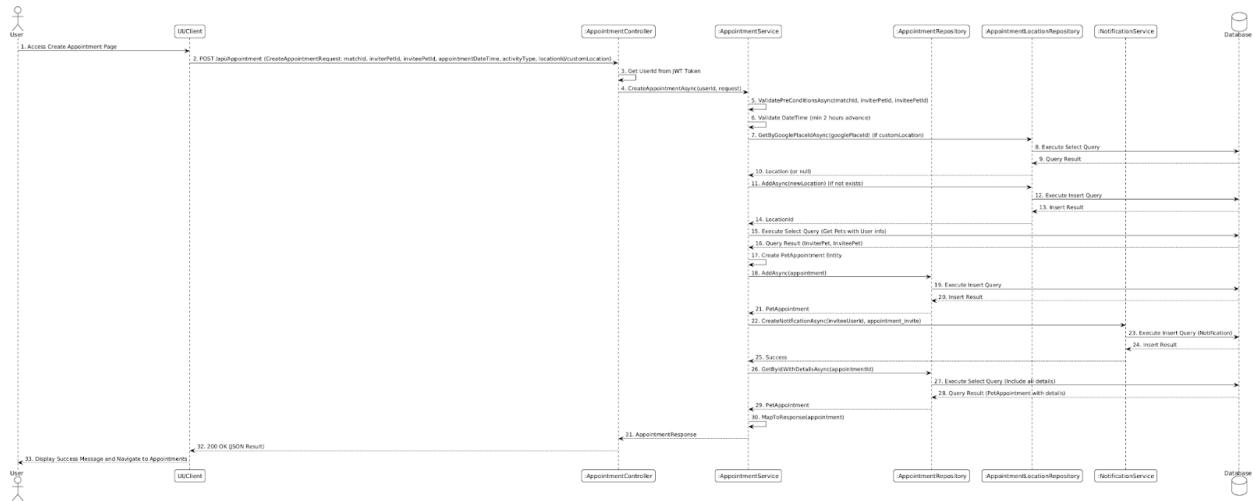


3.6.2 Sequence Diagram

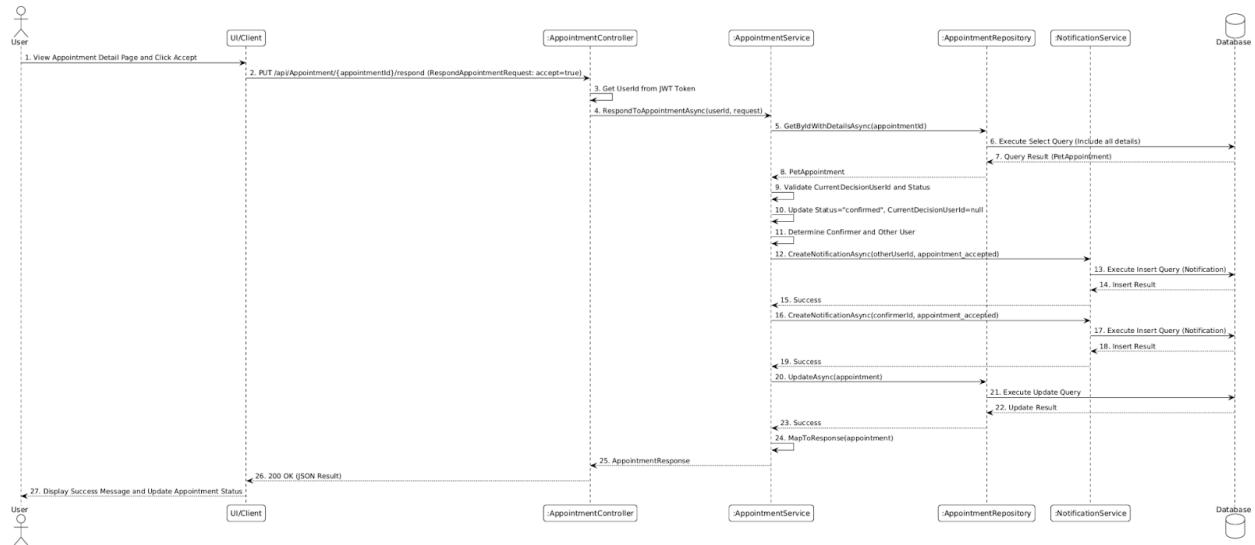
3.6.2.1 View My Appointment



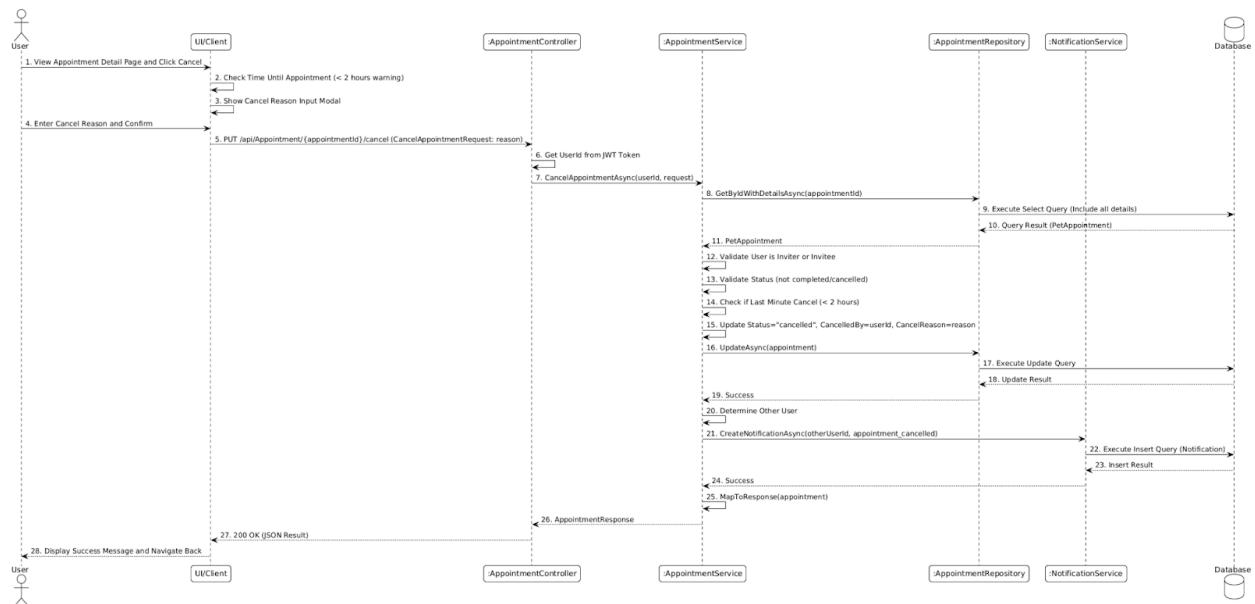
3.6.2.2 Create Appointment



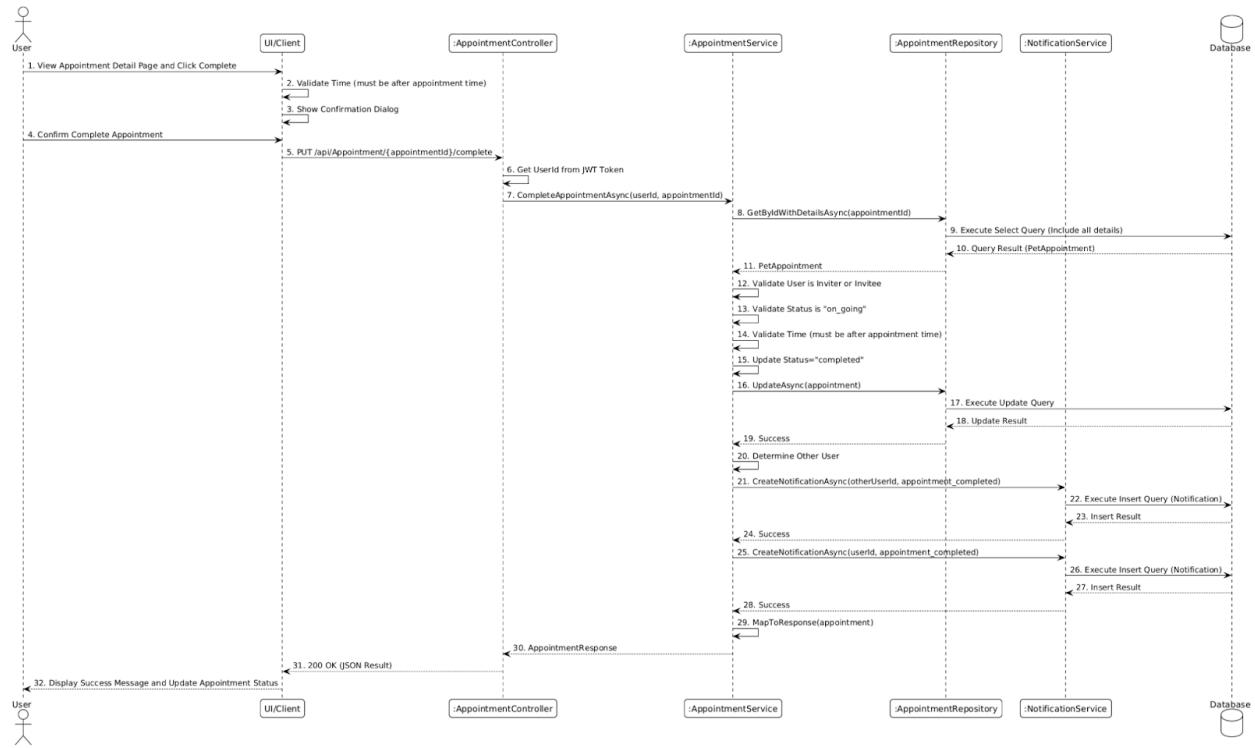
3.6.2.3 Accept Appointment



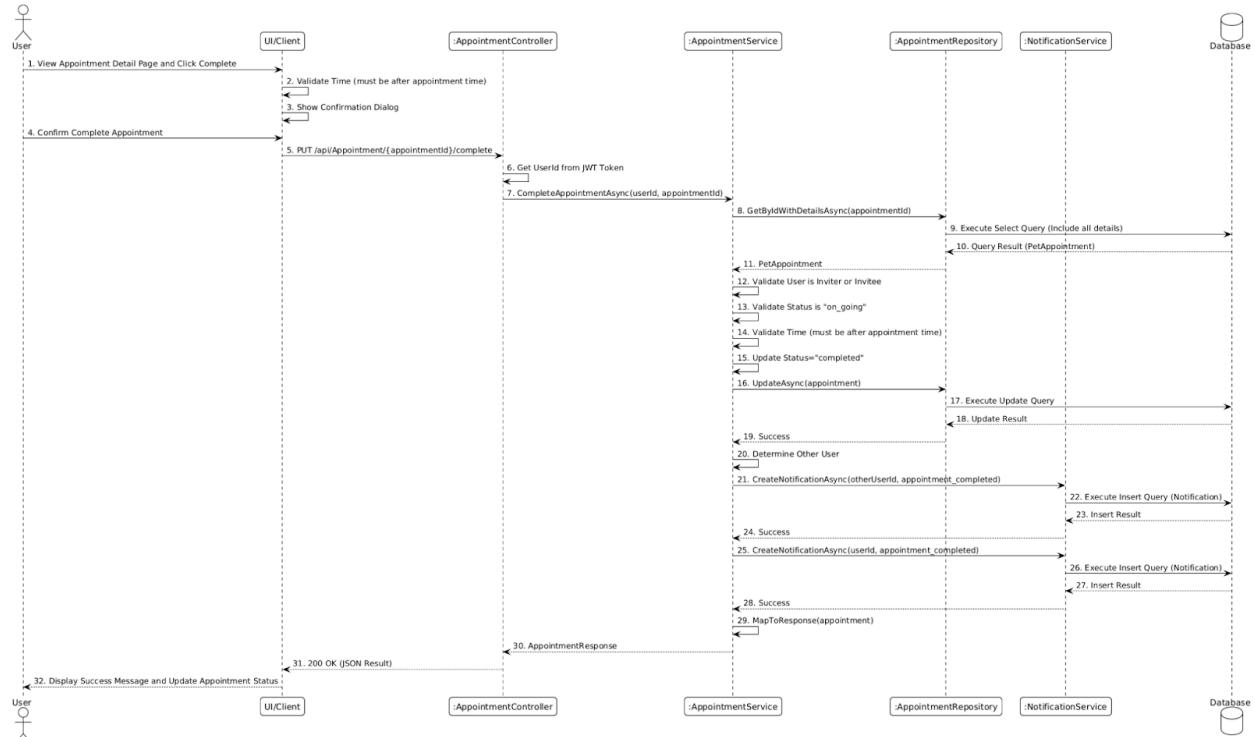
3.6.2.4 Cancel Appointment



3.6.2.5 Check-in at Appointment

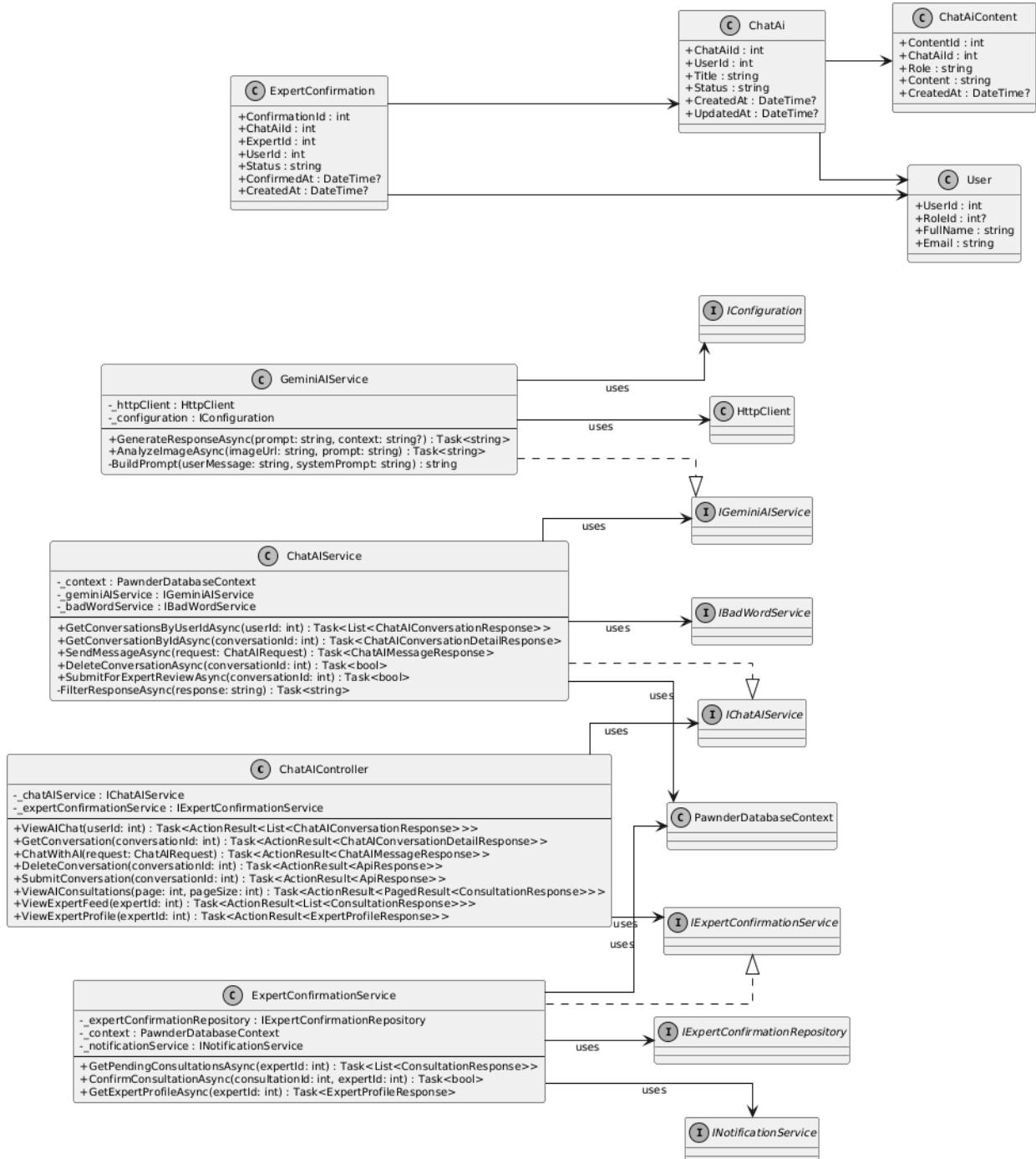


3.6.2.6 Complete Appointment



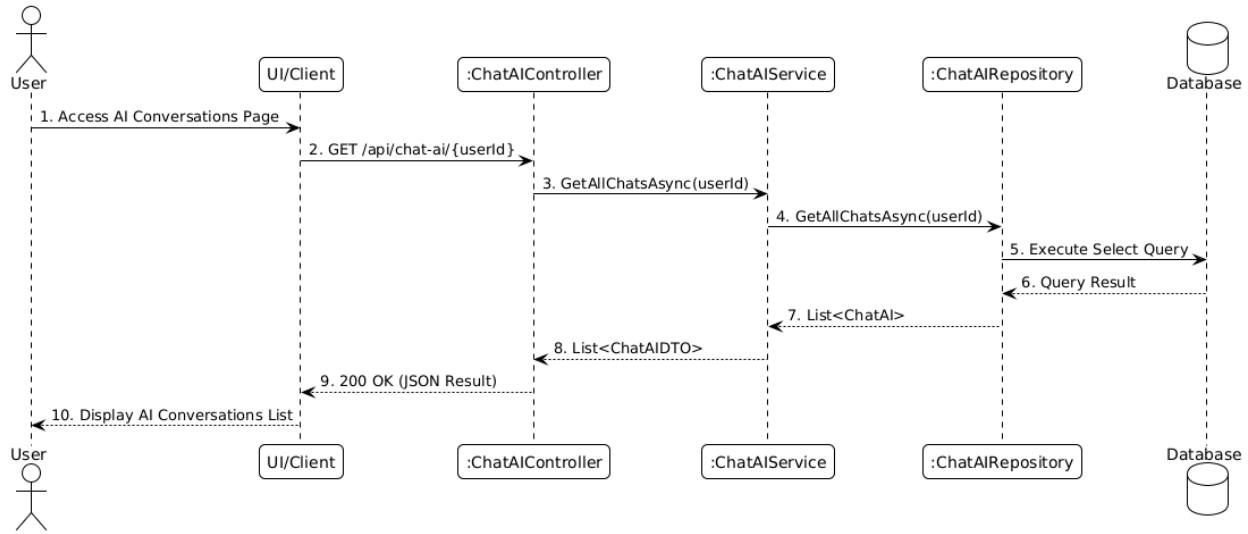
3.7 AI Consultation

3.7.1 Class Diagram

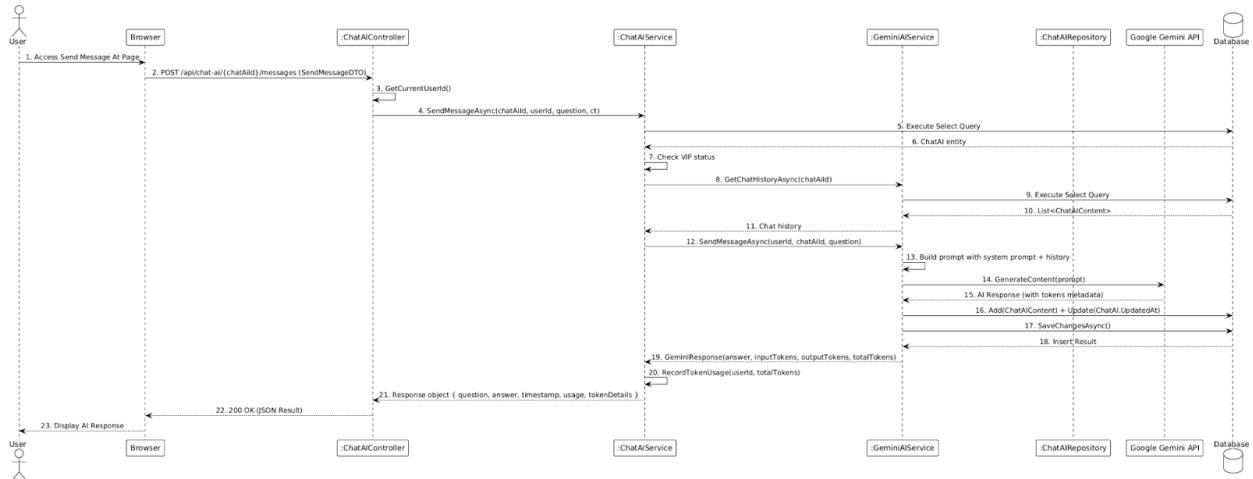


3.7.2 Sequence Diagram

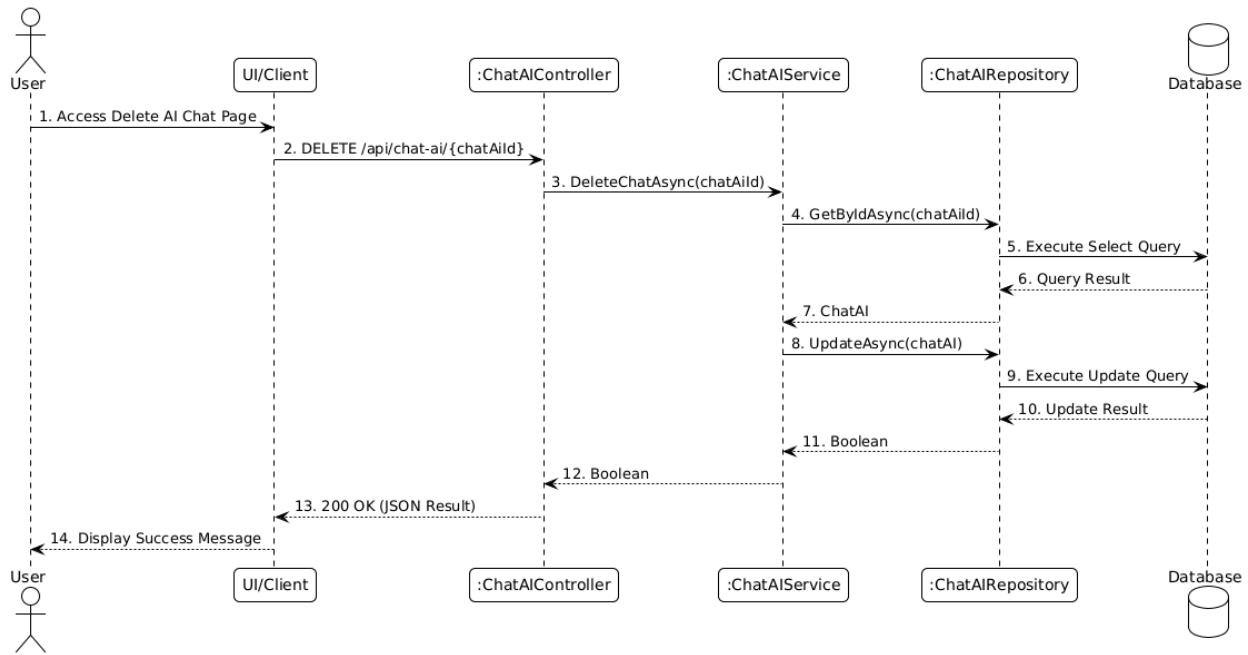
3.7.2.1 View AI Chat



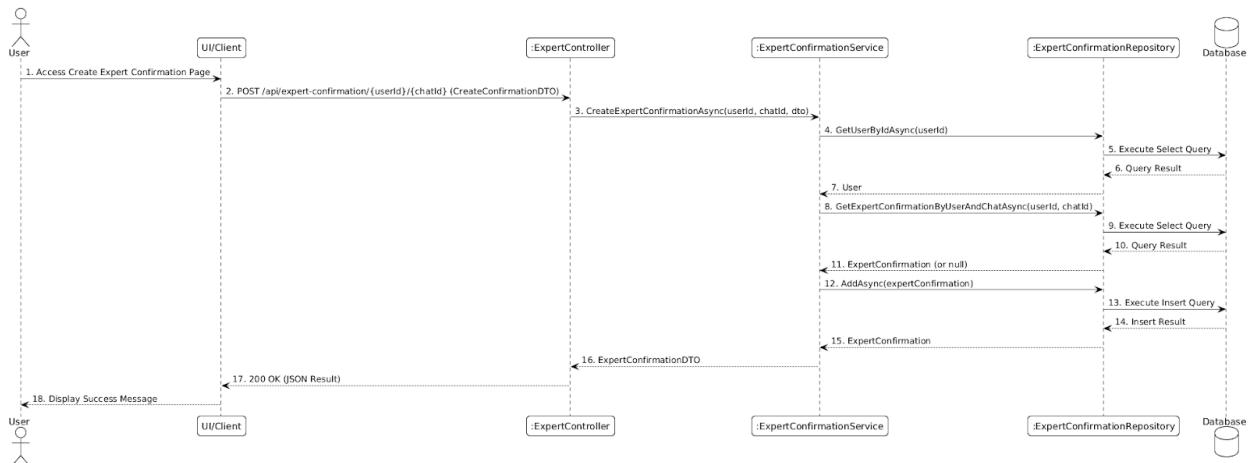
3.7.2.2 Chat with AI



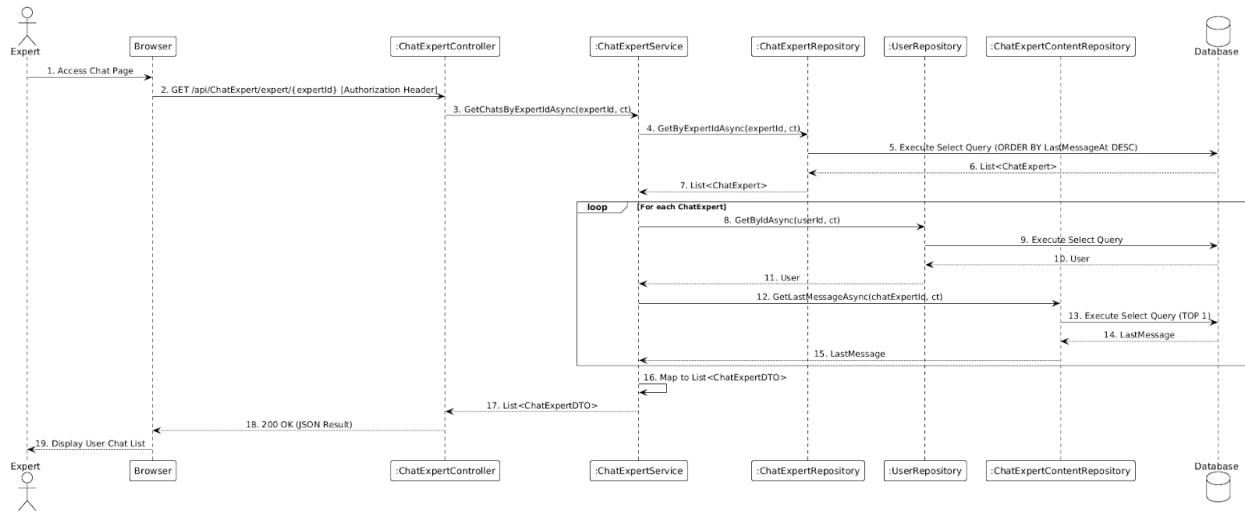
3.7.2.3 Delete AI Conversation



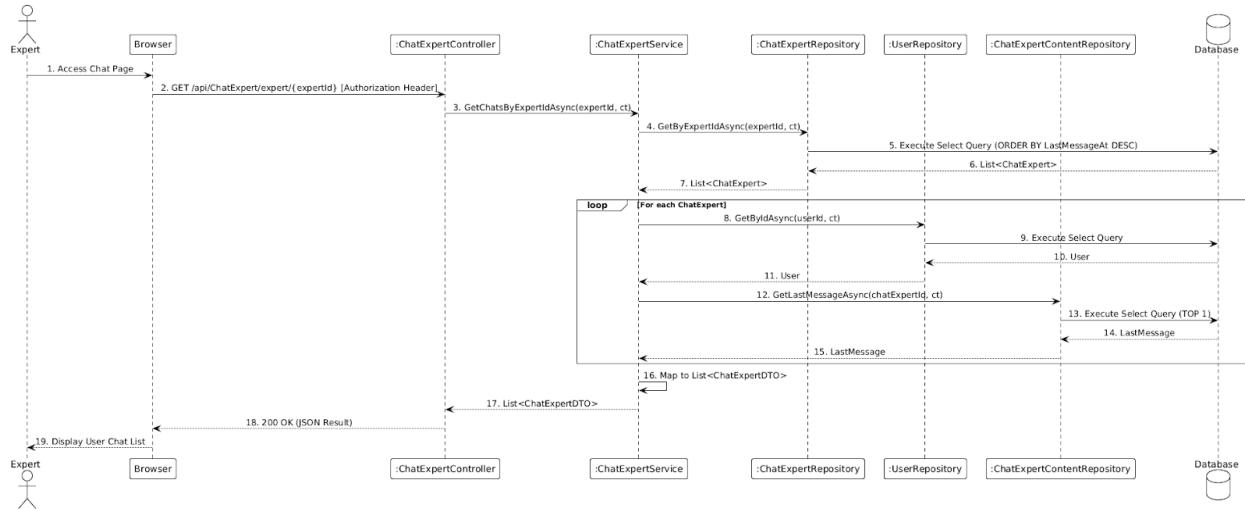
3.7.2.4 Submit AI Conversation



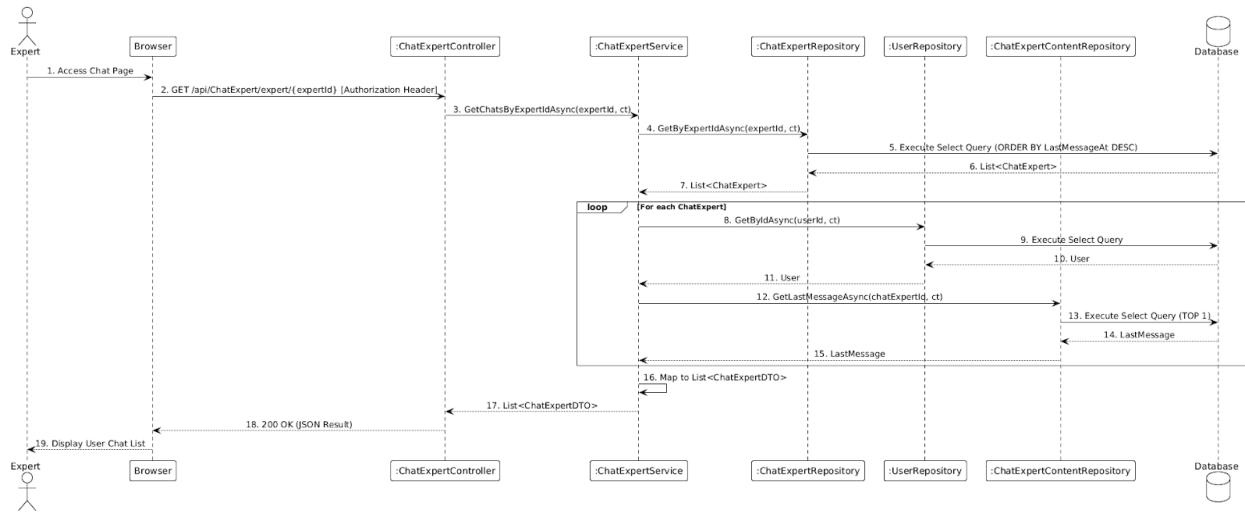
3.7.2.5 View AI Consultation



3.7.2.6 View Expert Feed

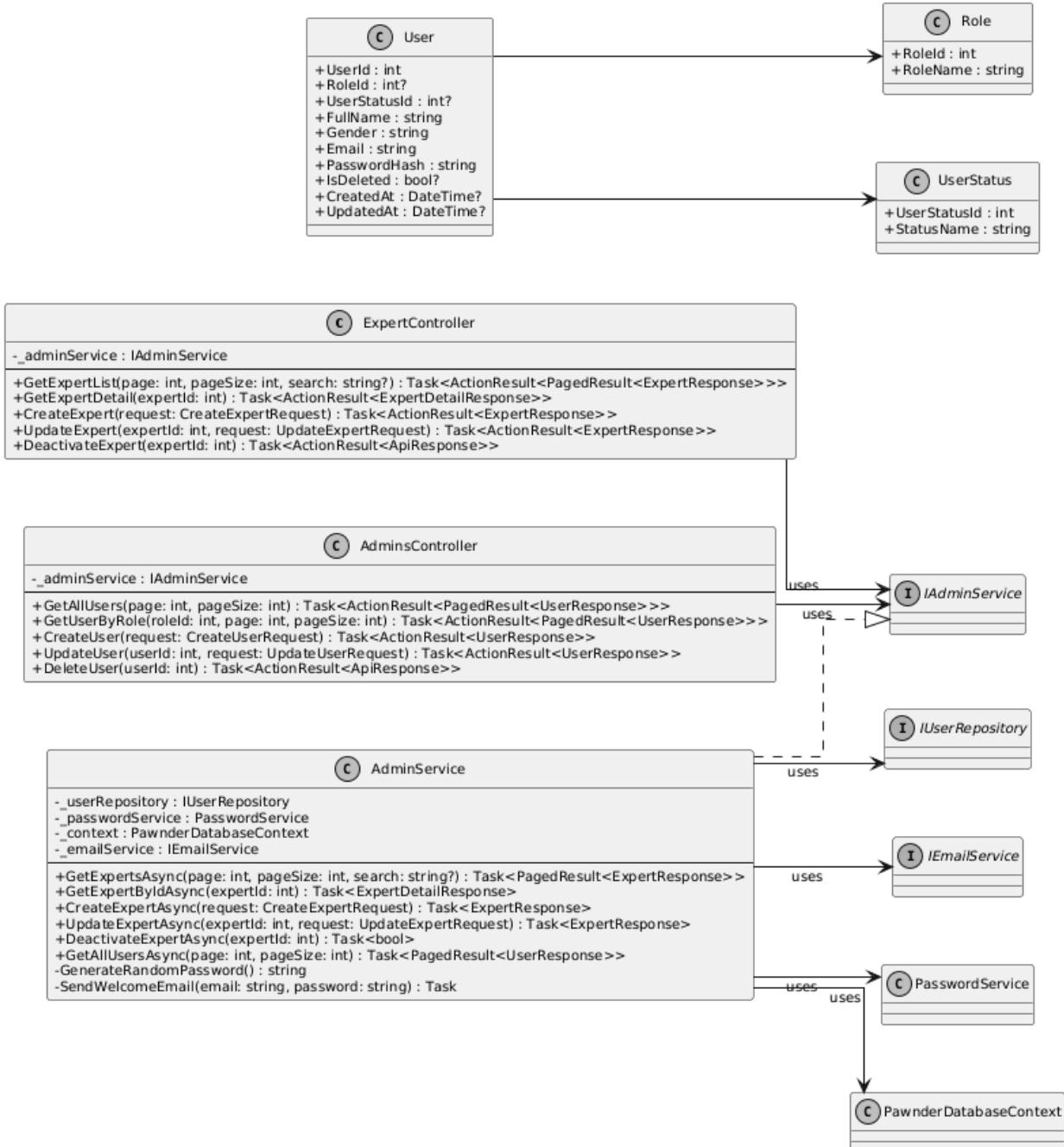


3.7.2.7 View Expert Profile



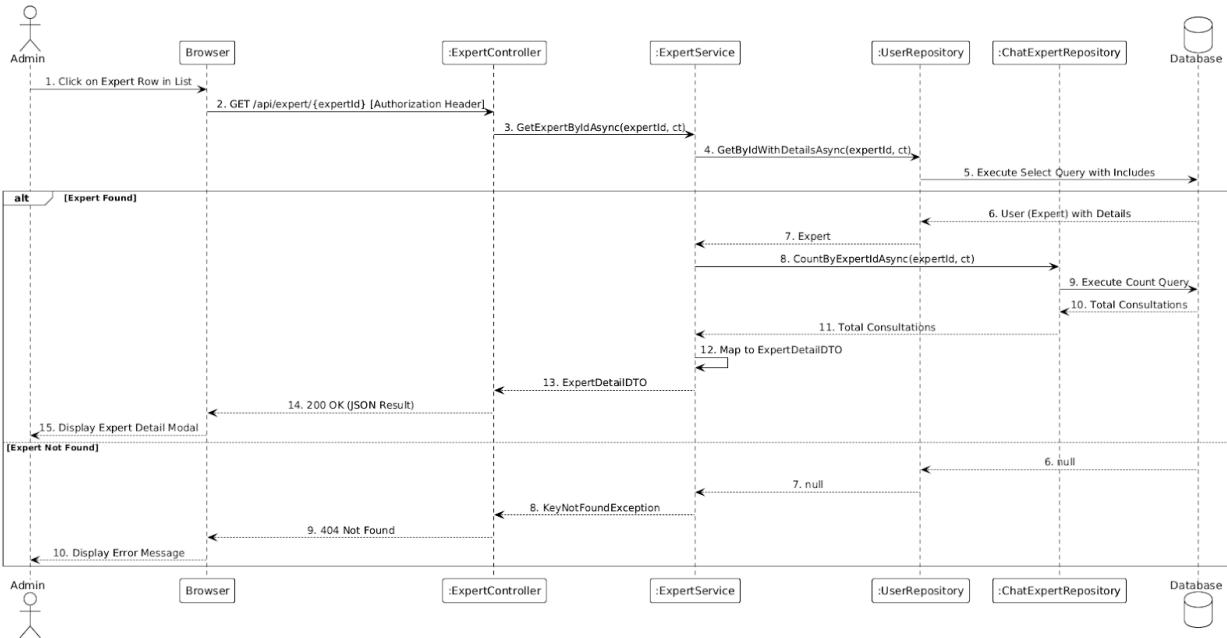
3.8 Expert Management

3.8.1 Class Diagram

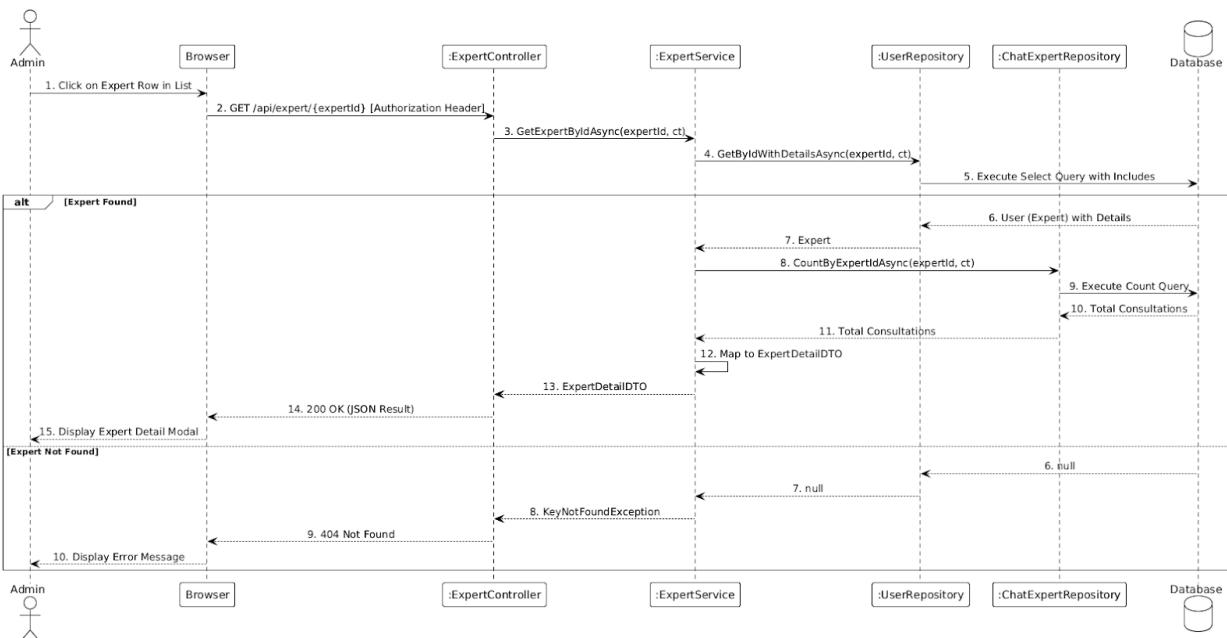


3.8.2 Sequence Diagram

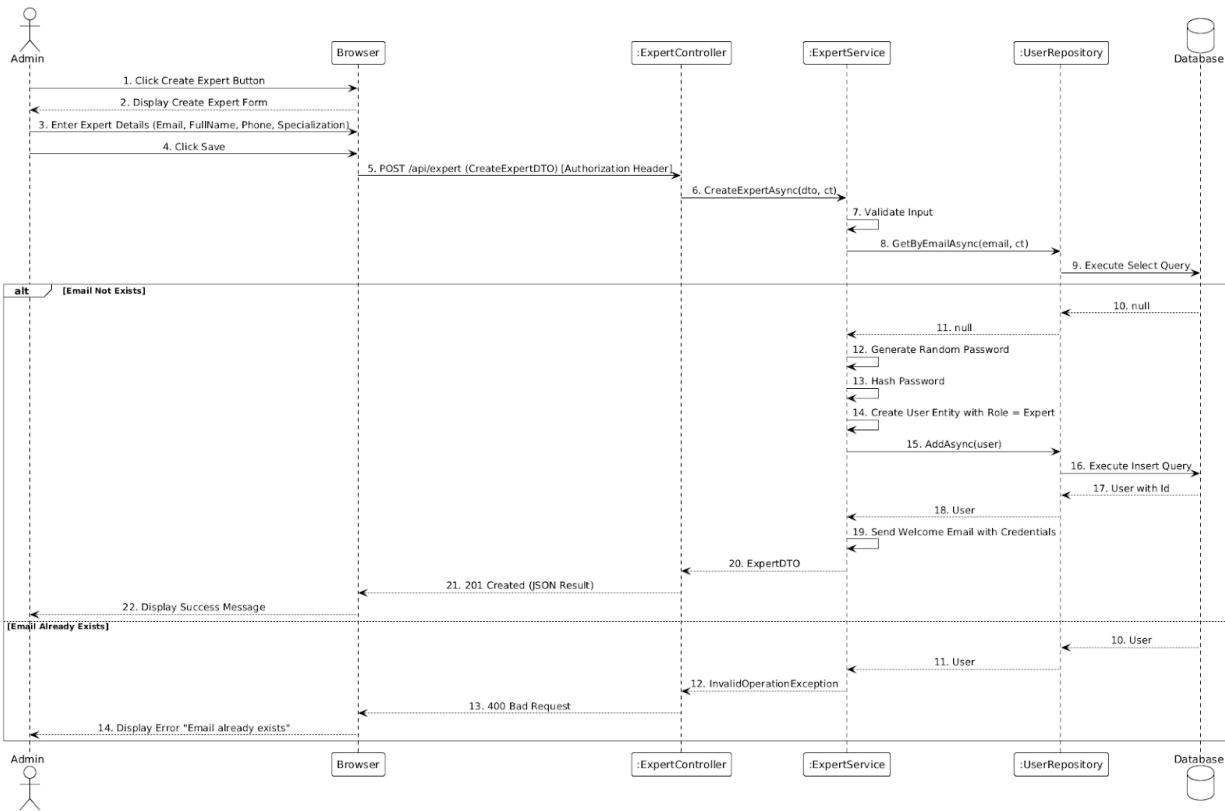
3.8.2.1 View Expert List



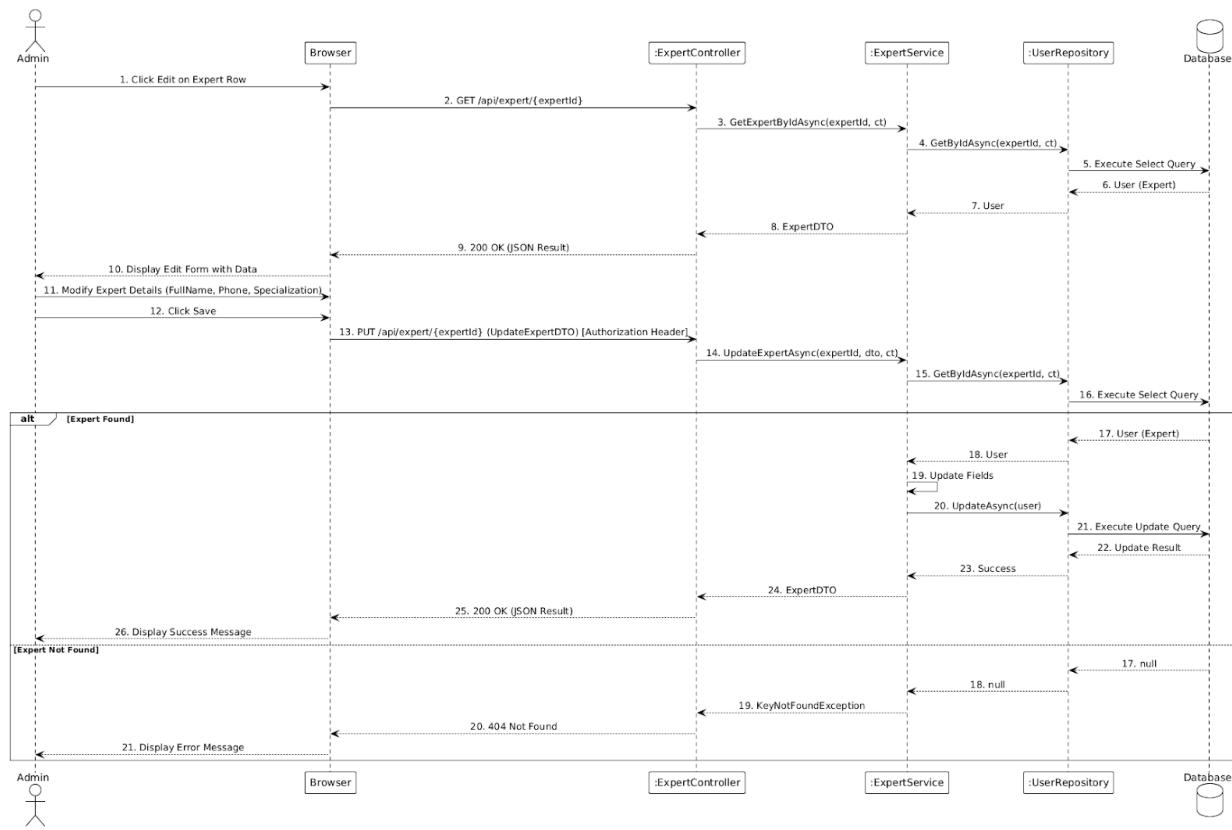
3.8.2.2 View Expert Detail



3.8.2.3 Create Expert Account

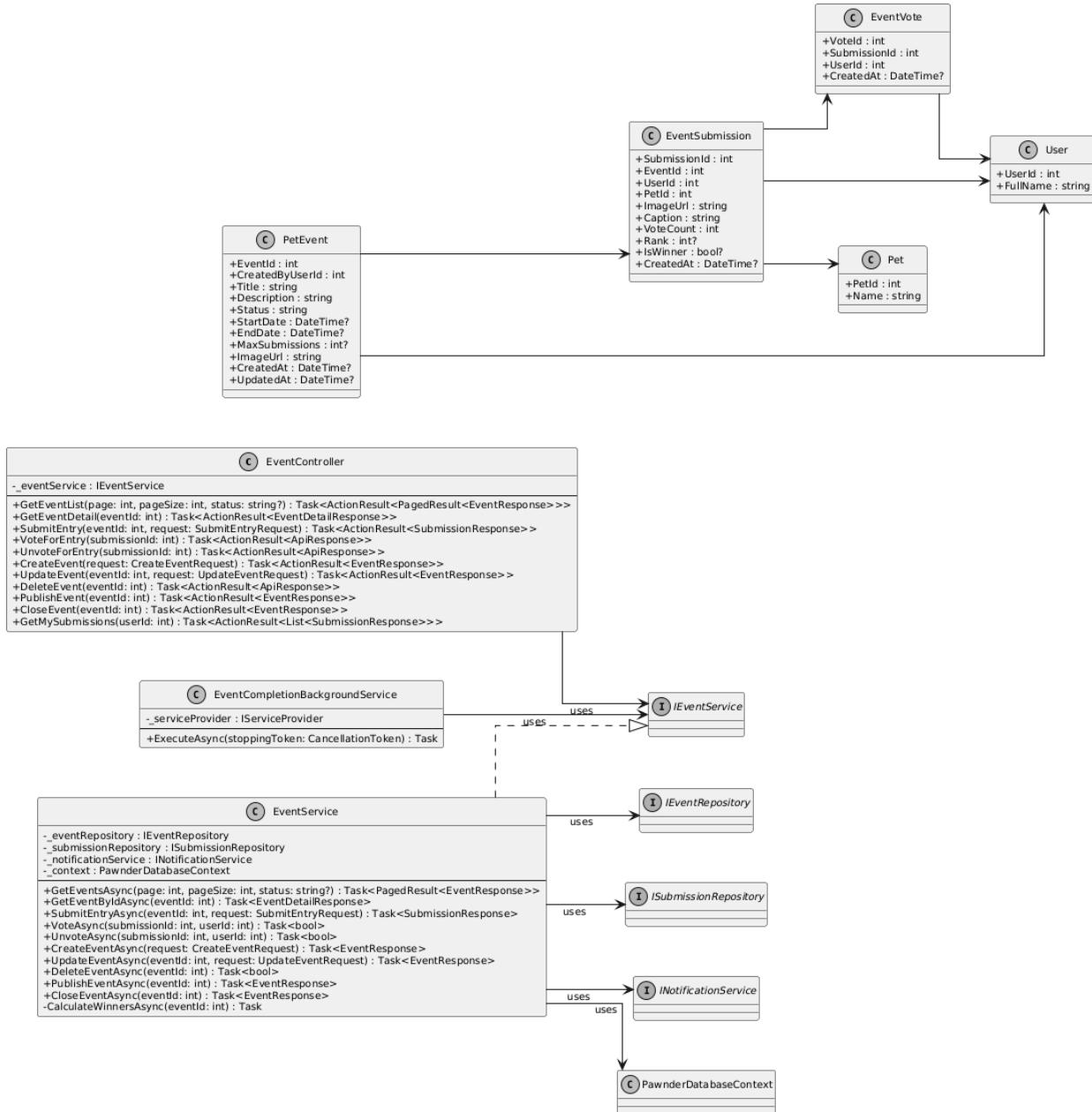


3.8.2.4 Edit Expert Account



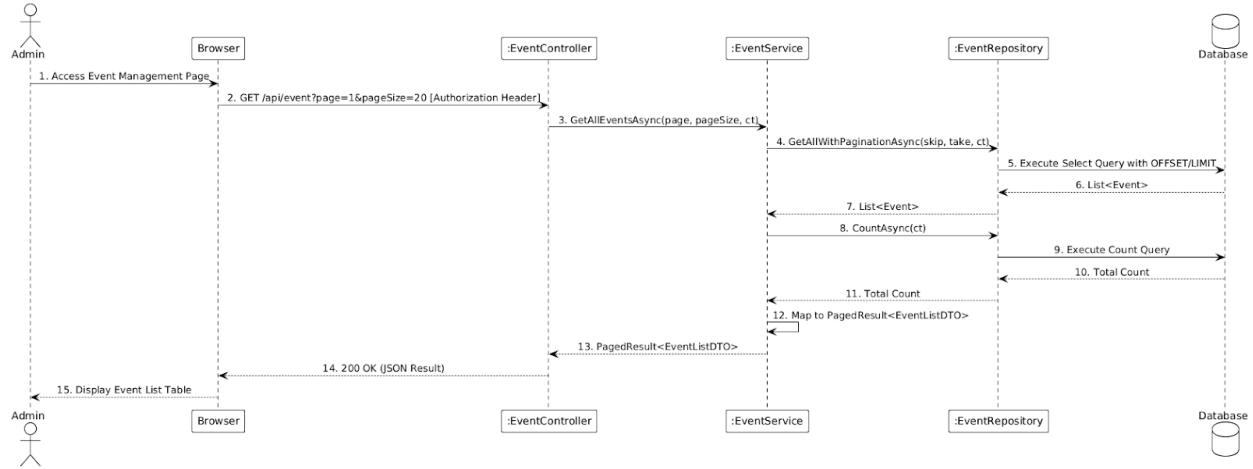
3.9 Event Management

3.9.1 Class Diagram

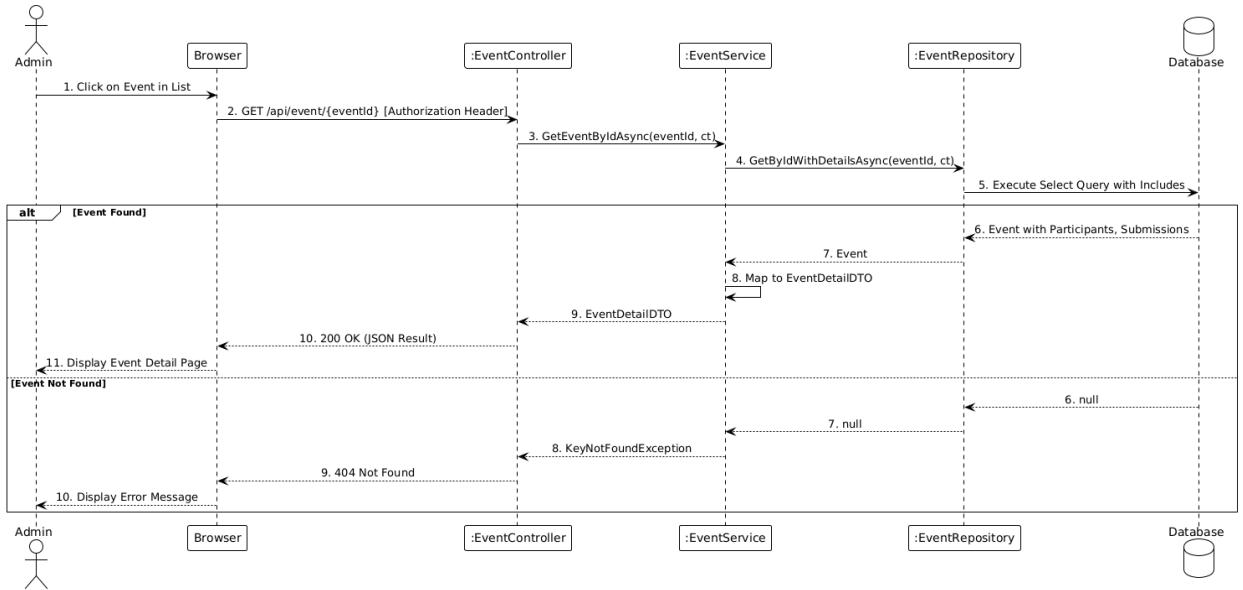


3.9.2 Sequence Diagram

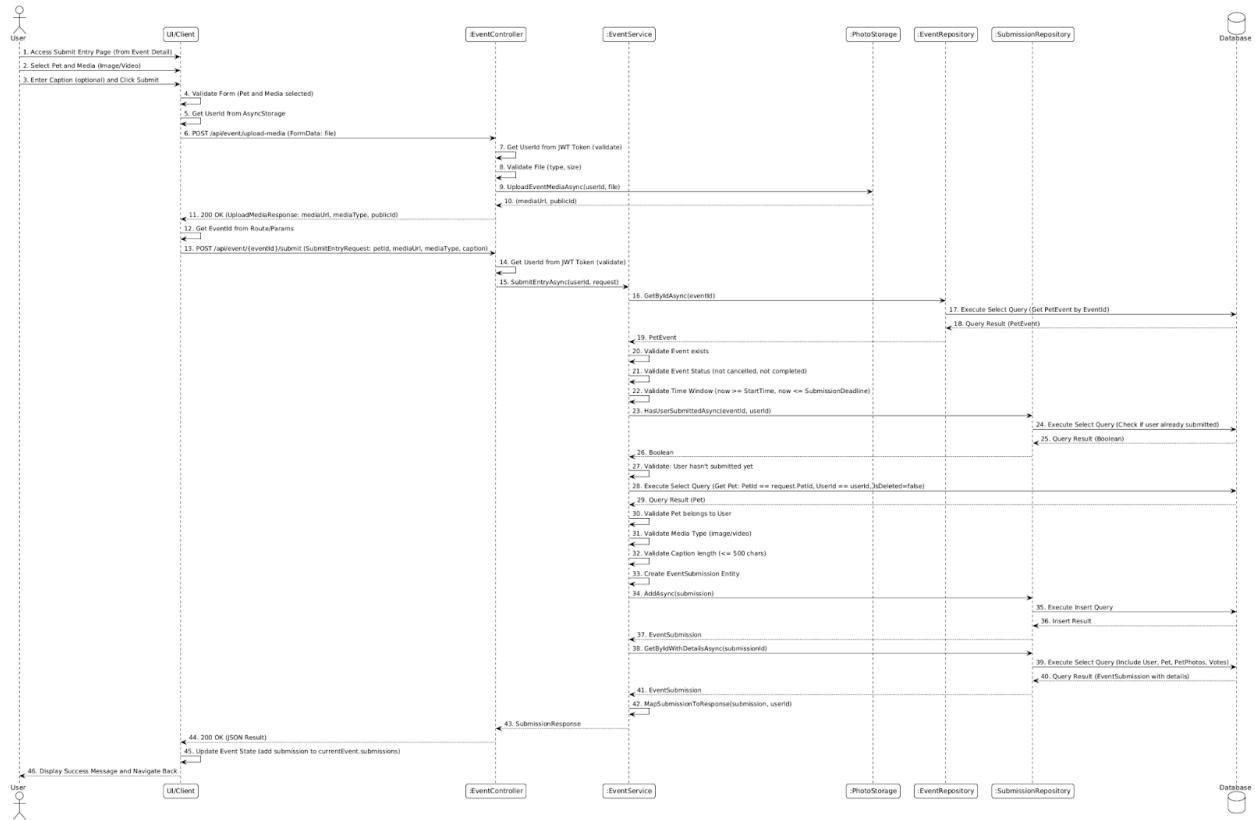
3.9.2.1 View Event List



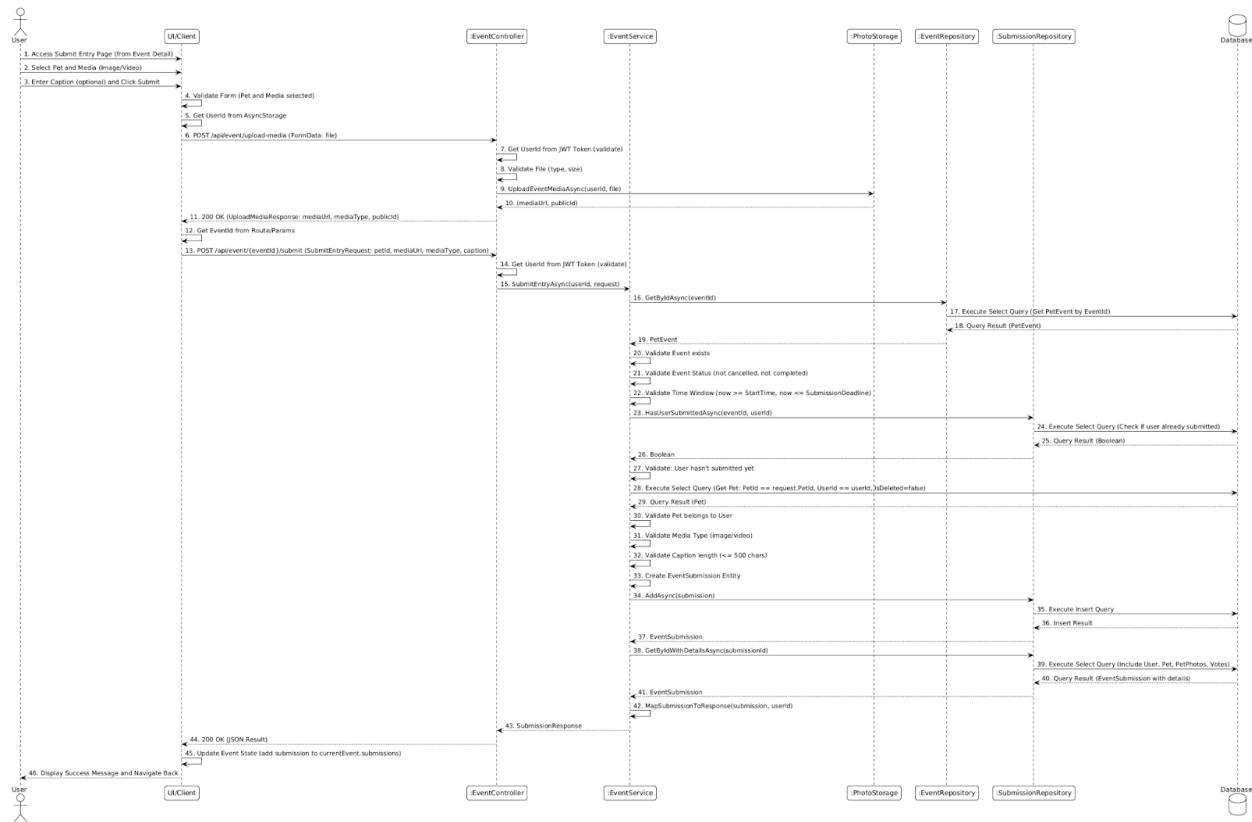
3.9.2.2 View Event Detail



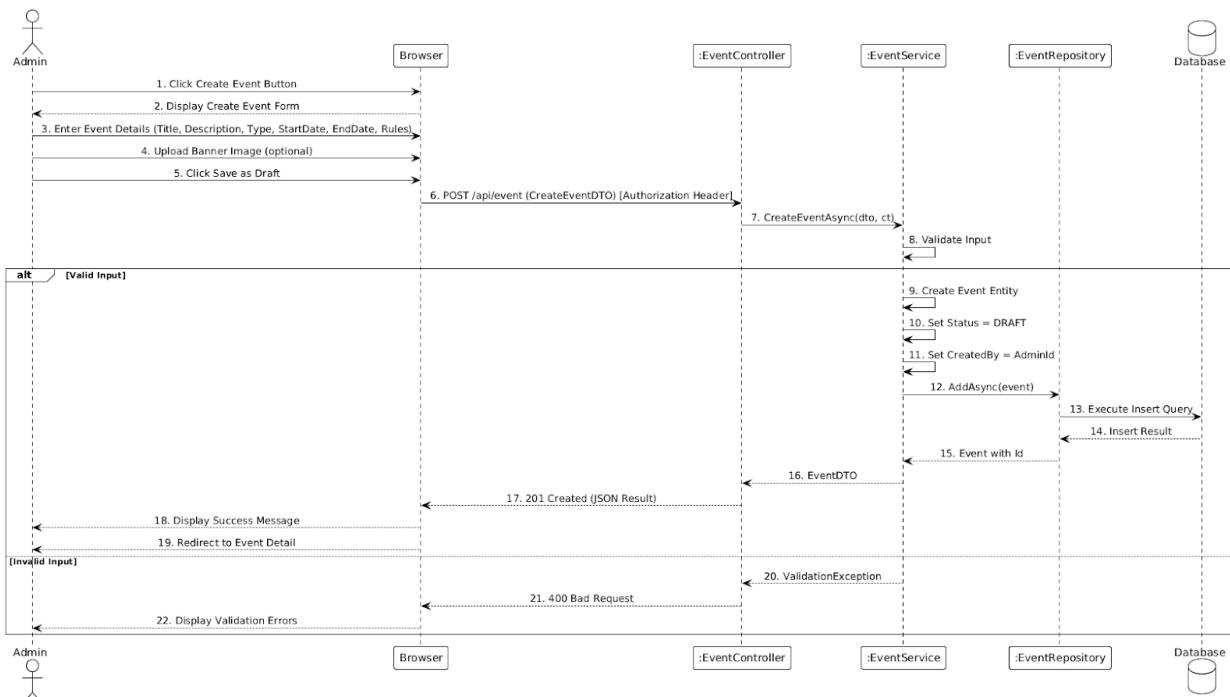
3.9.2.3 Submit Event Entry



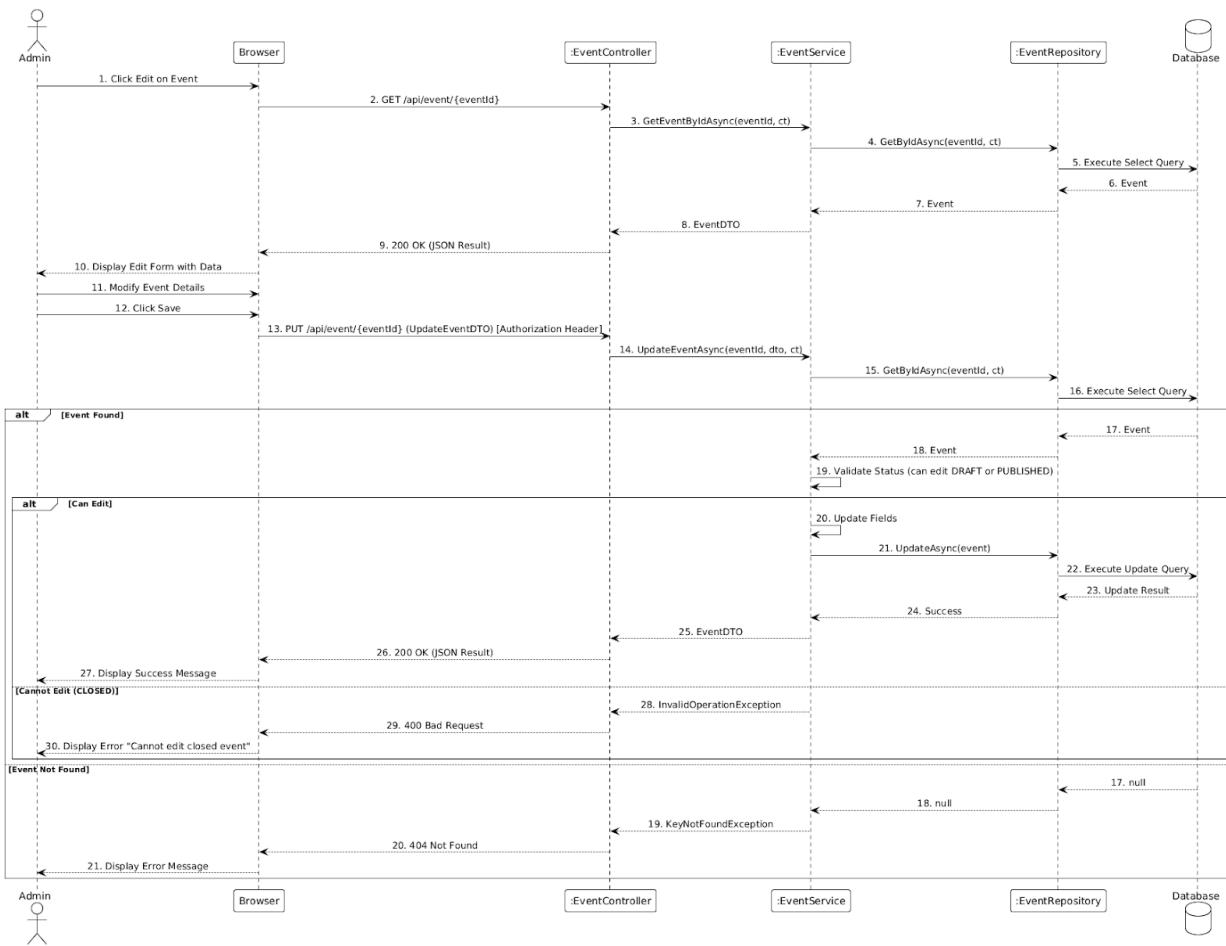
3.9.2.4 Vote / Unvote for Event



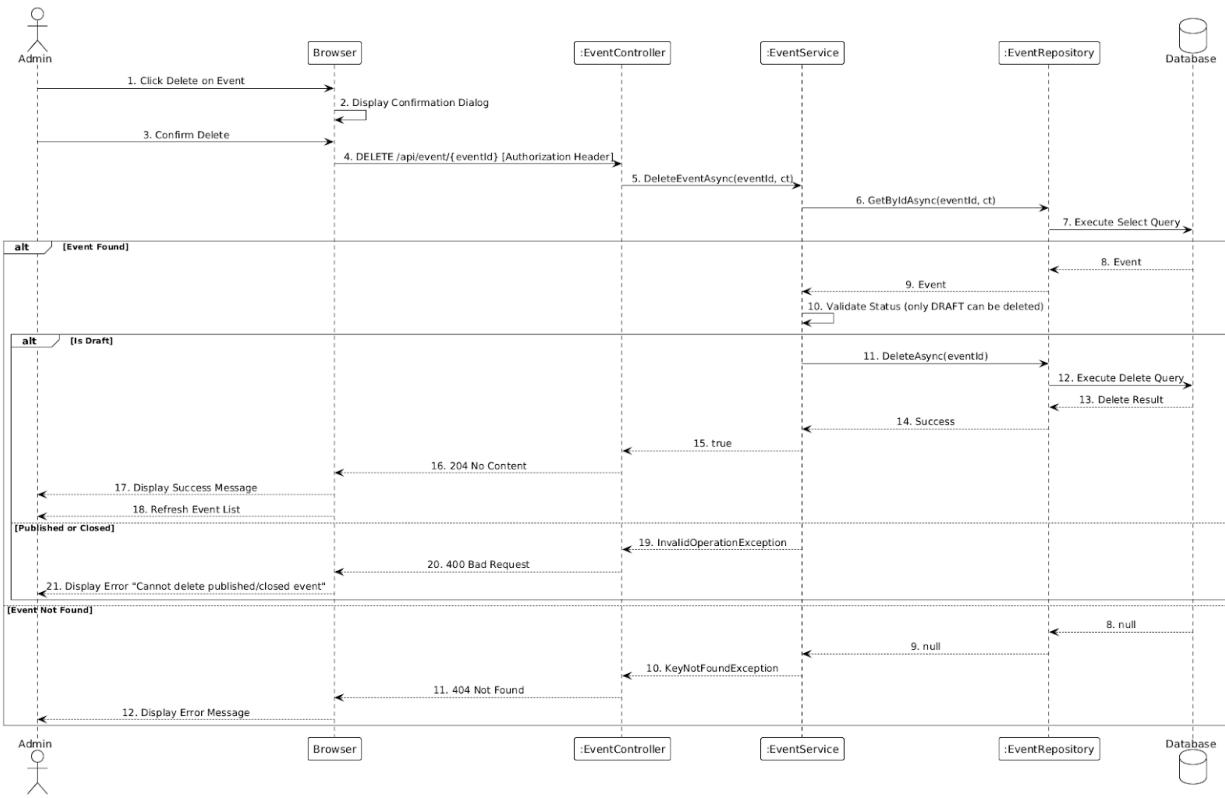
3.9.2.5 Create Event



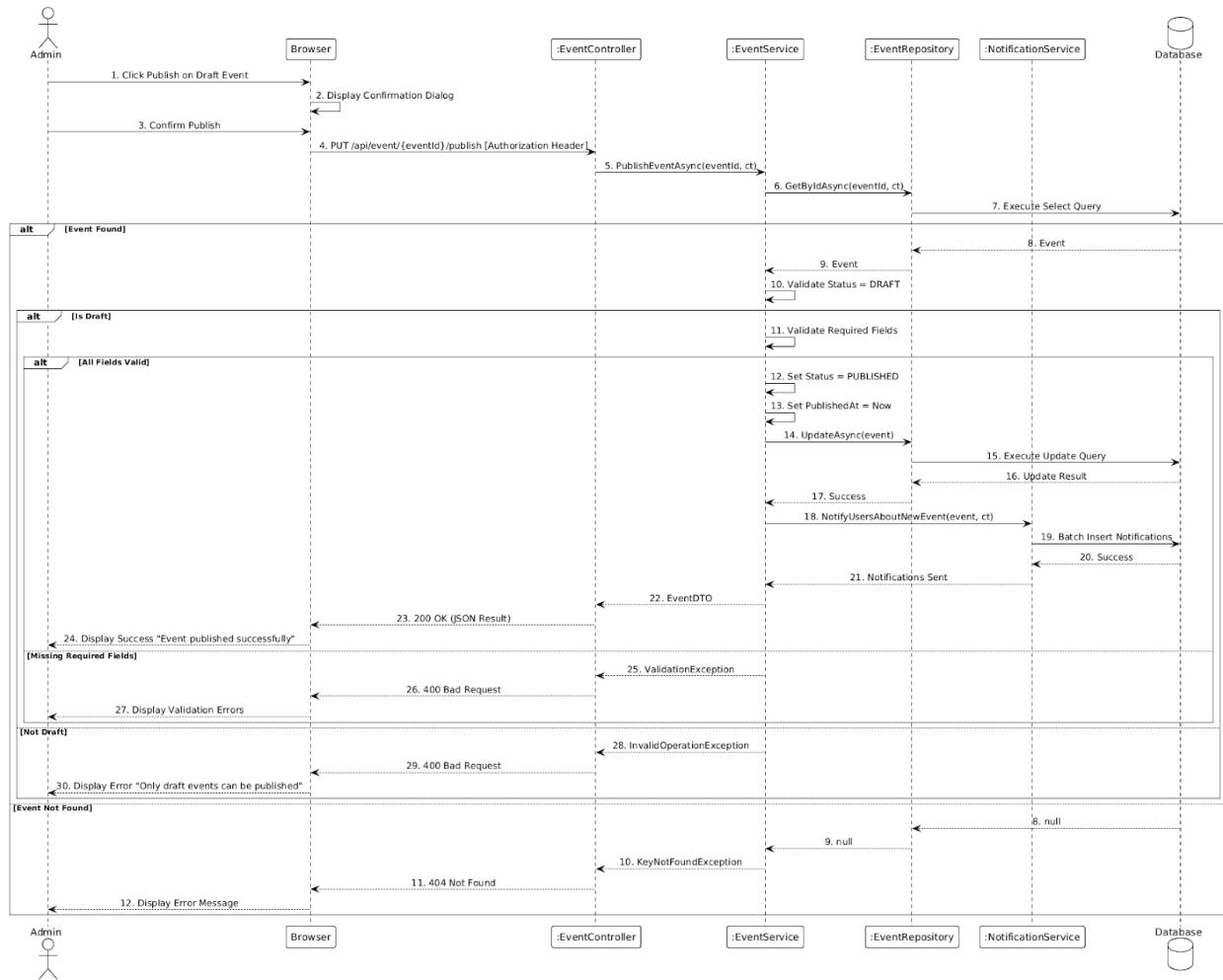
3.9.2.6 Edit Event



3.9.2.7 Delete Event

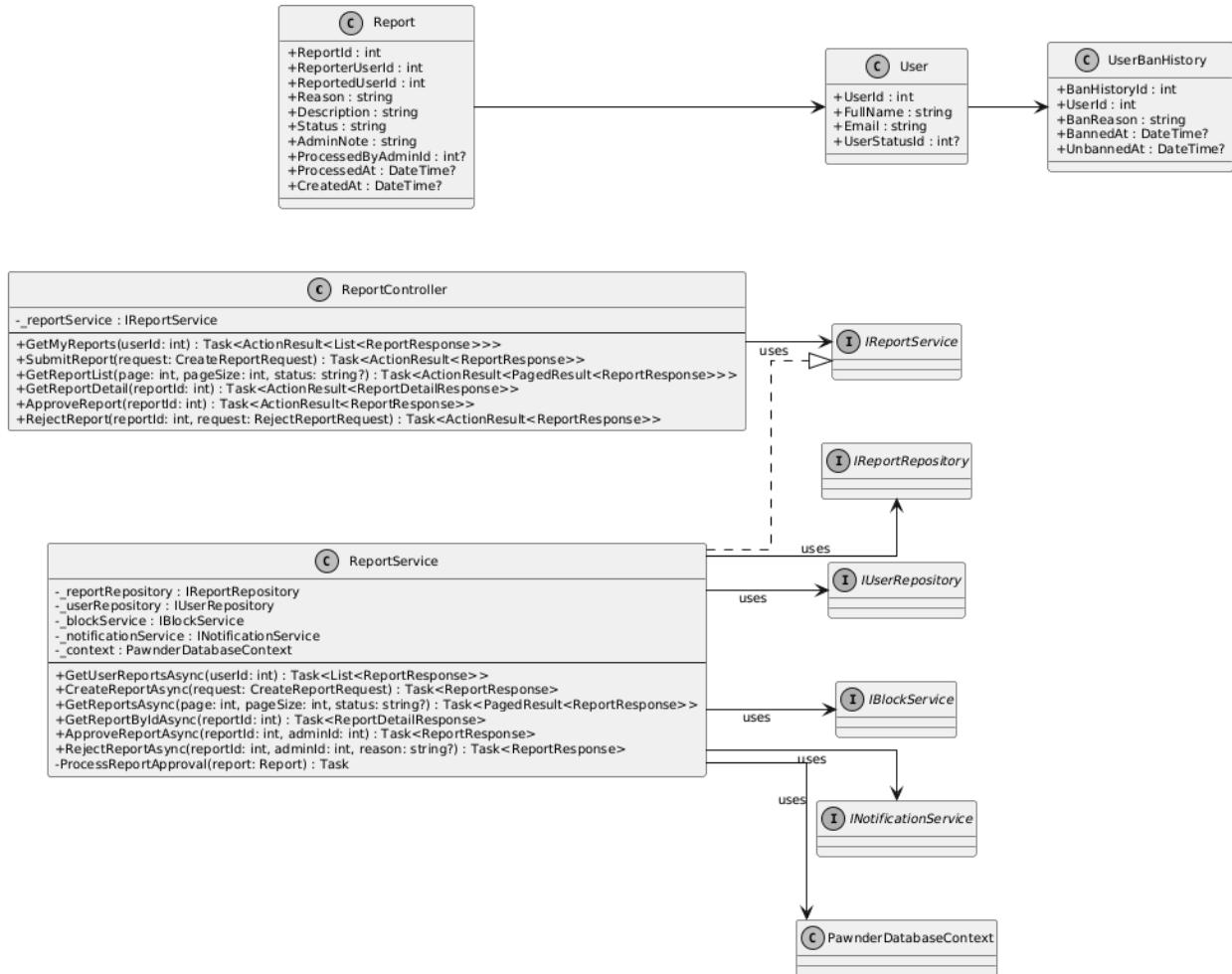


3.9.2.8 Publish Event



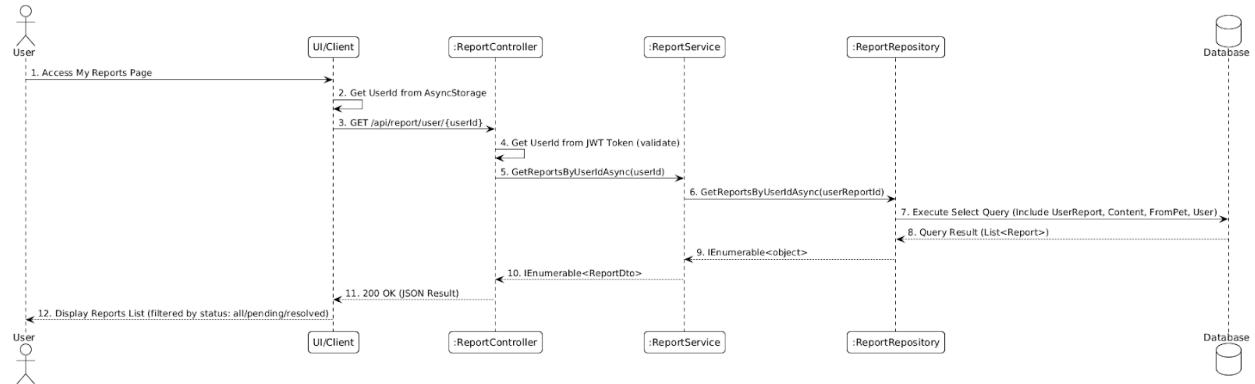
3.10 Report Management

3.10.1 Class Diagram

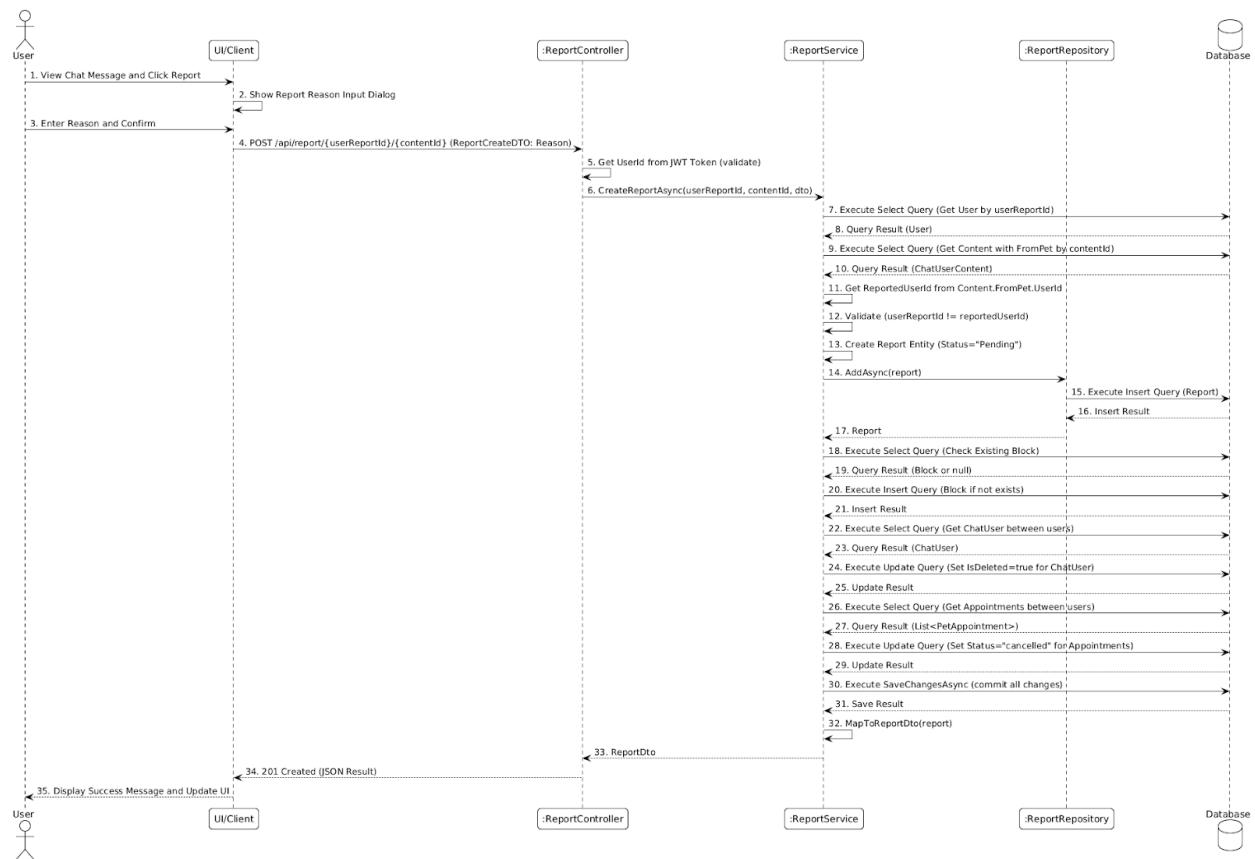


3.10.2 Sequence Diagram

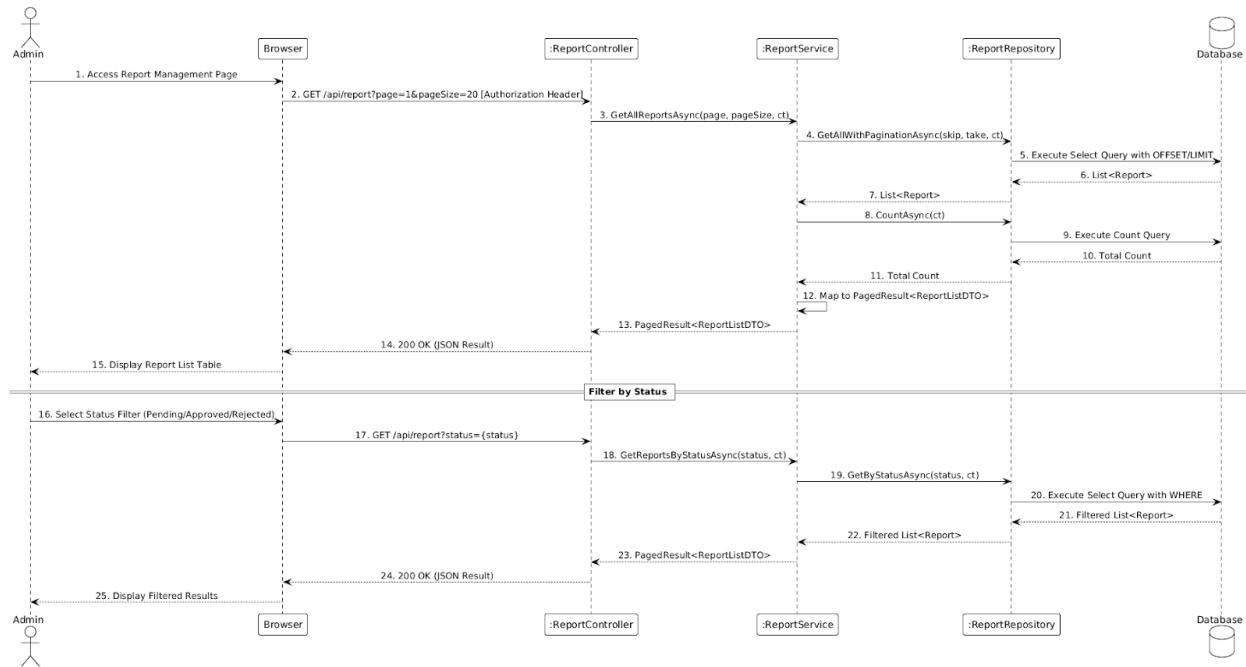
3.10.2.1 View My Report



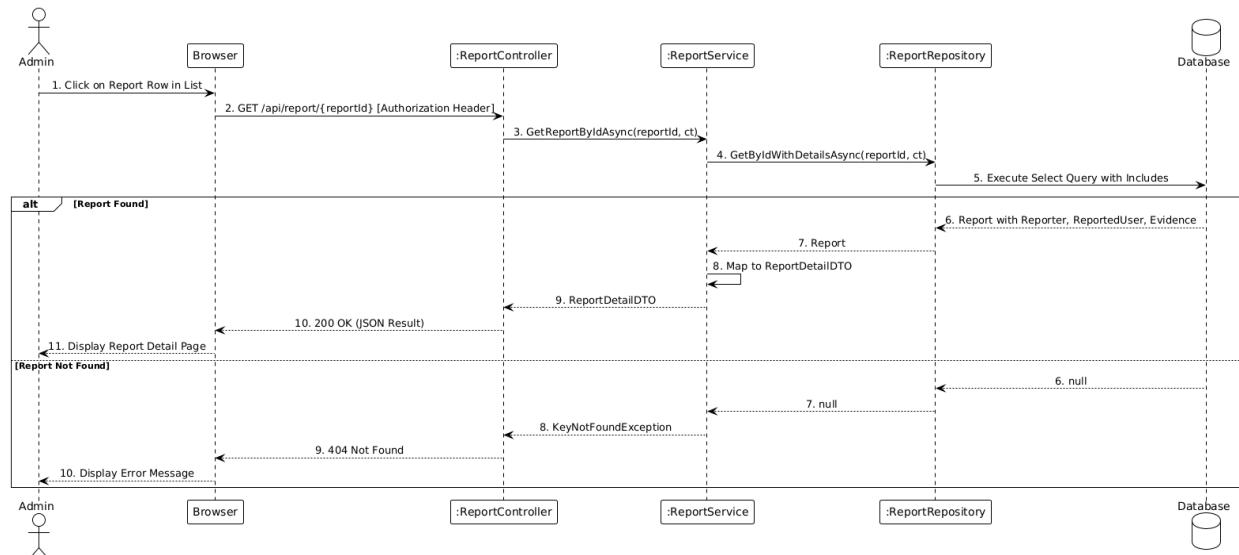
3.10.2.2 Submit Report



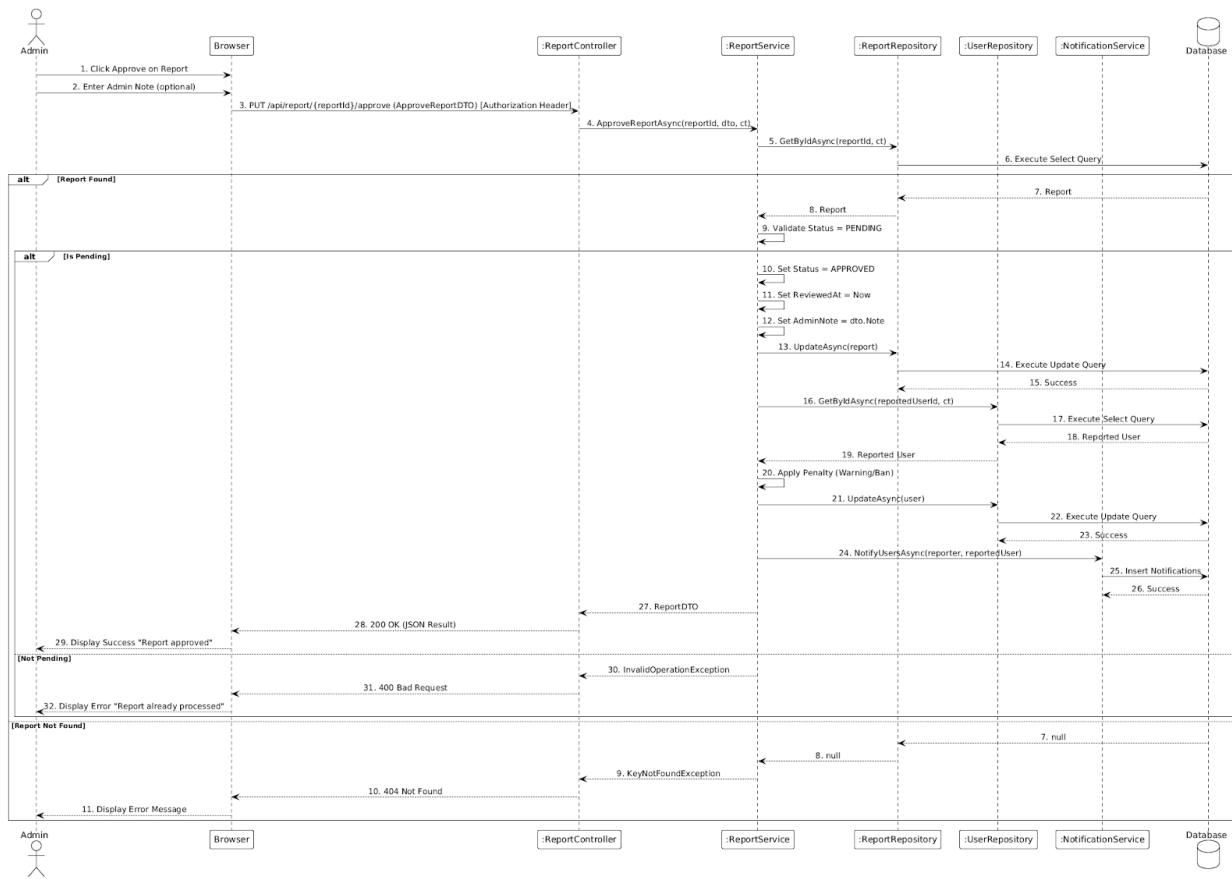
3.10.2.3 View Report List



3.10.2.4 View Report Detail

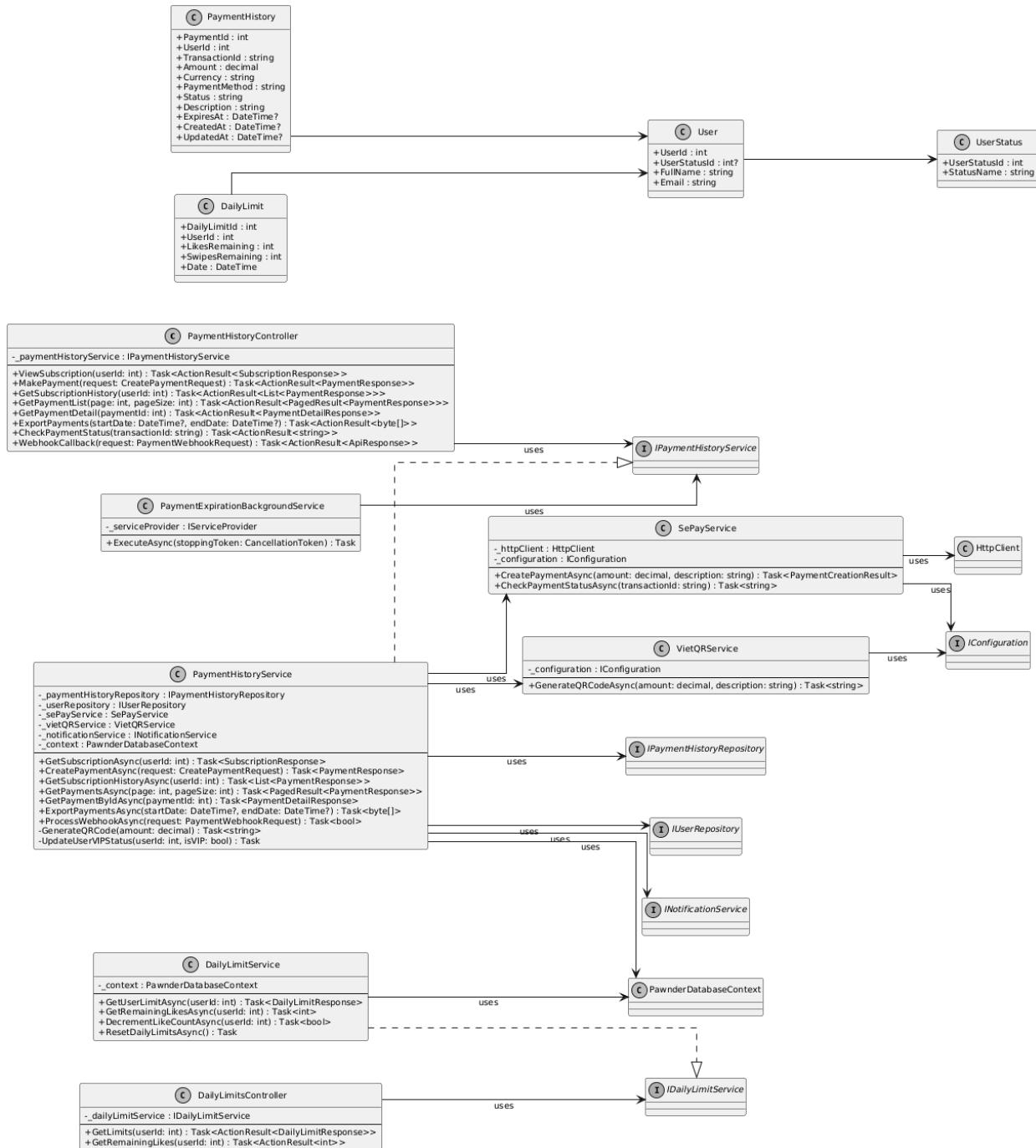


3.10.2.5 Approve Report



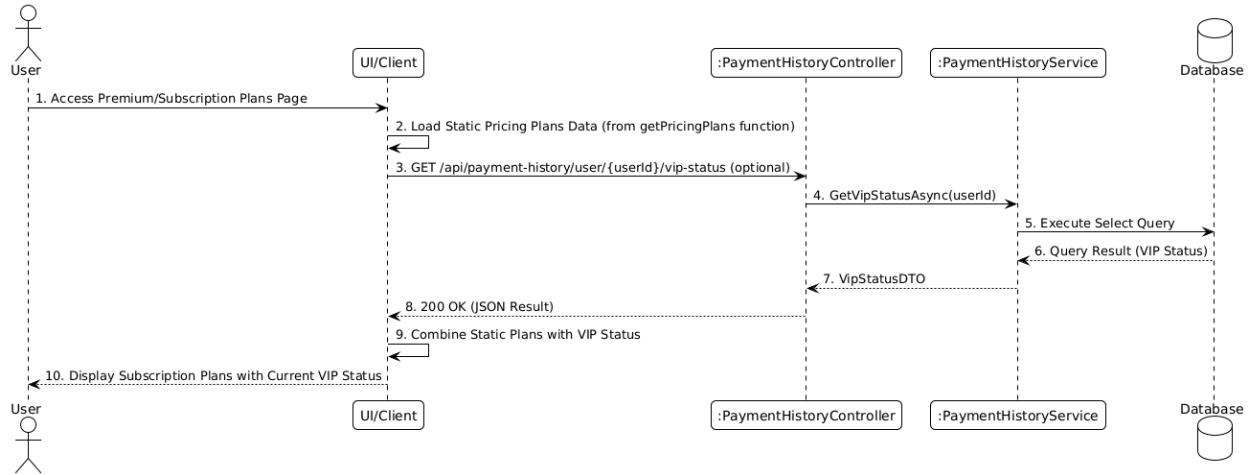
3.11 Subscription & Payment

3.11.1 Class Diagram

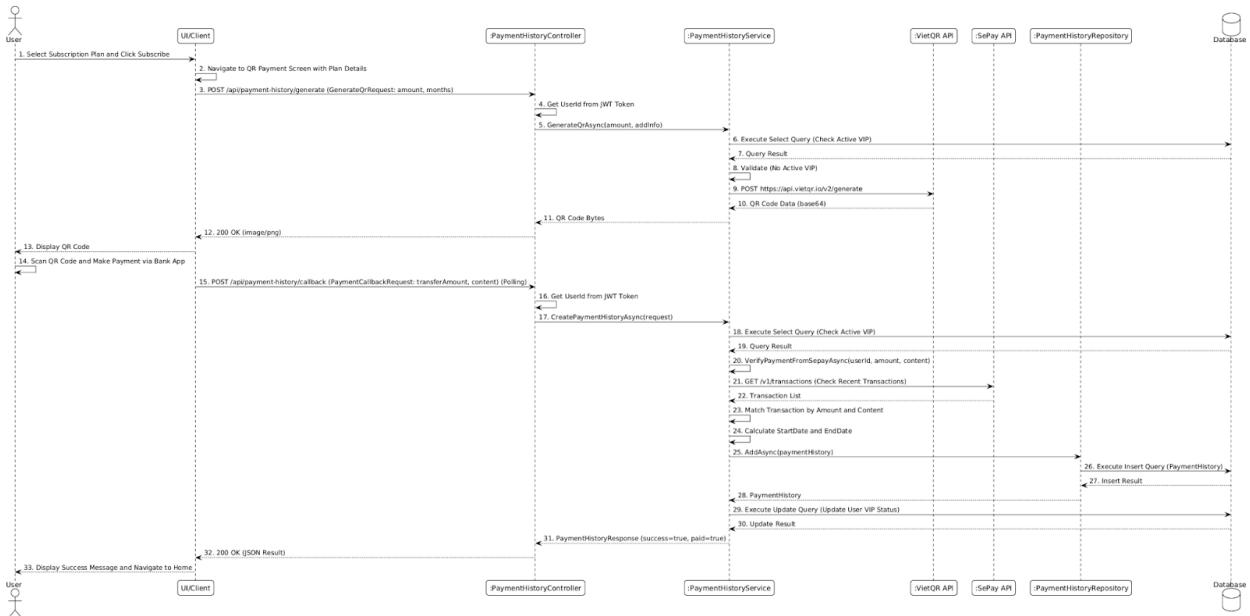


3.11.2 Sequence Diagram

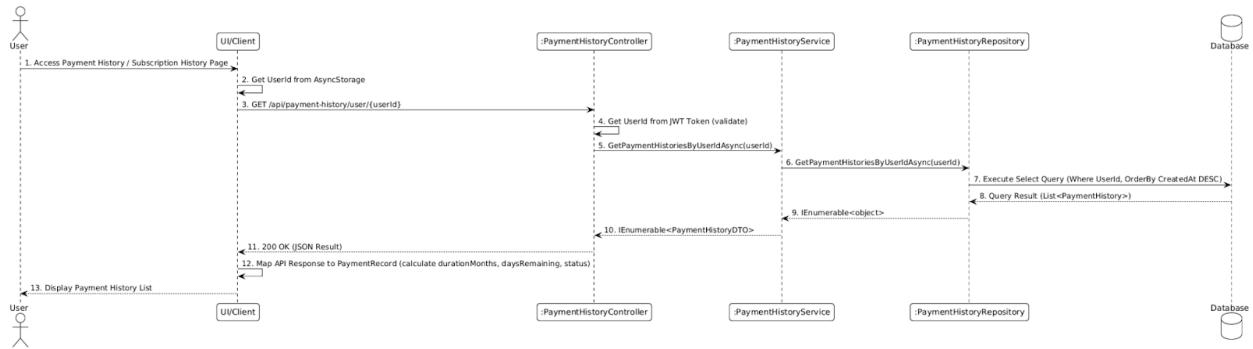
3.11.2.1 View Subscription



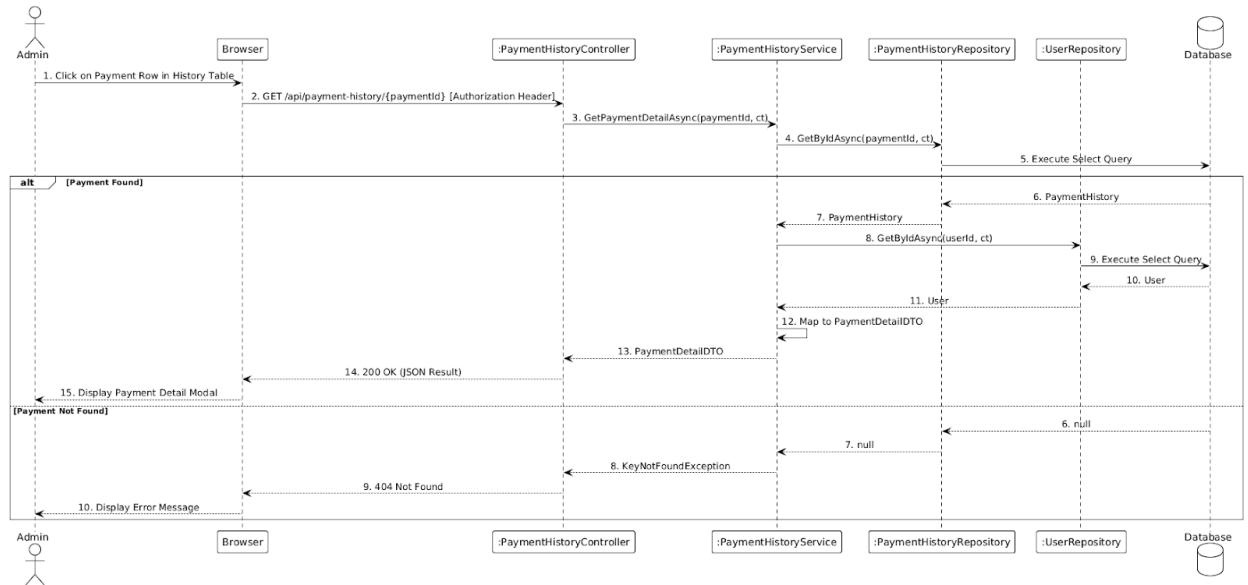
3.11.2.2 Make Payment



3.11.2.3 View Subscription History

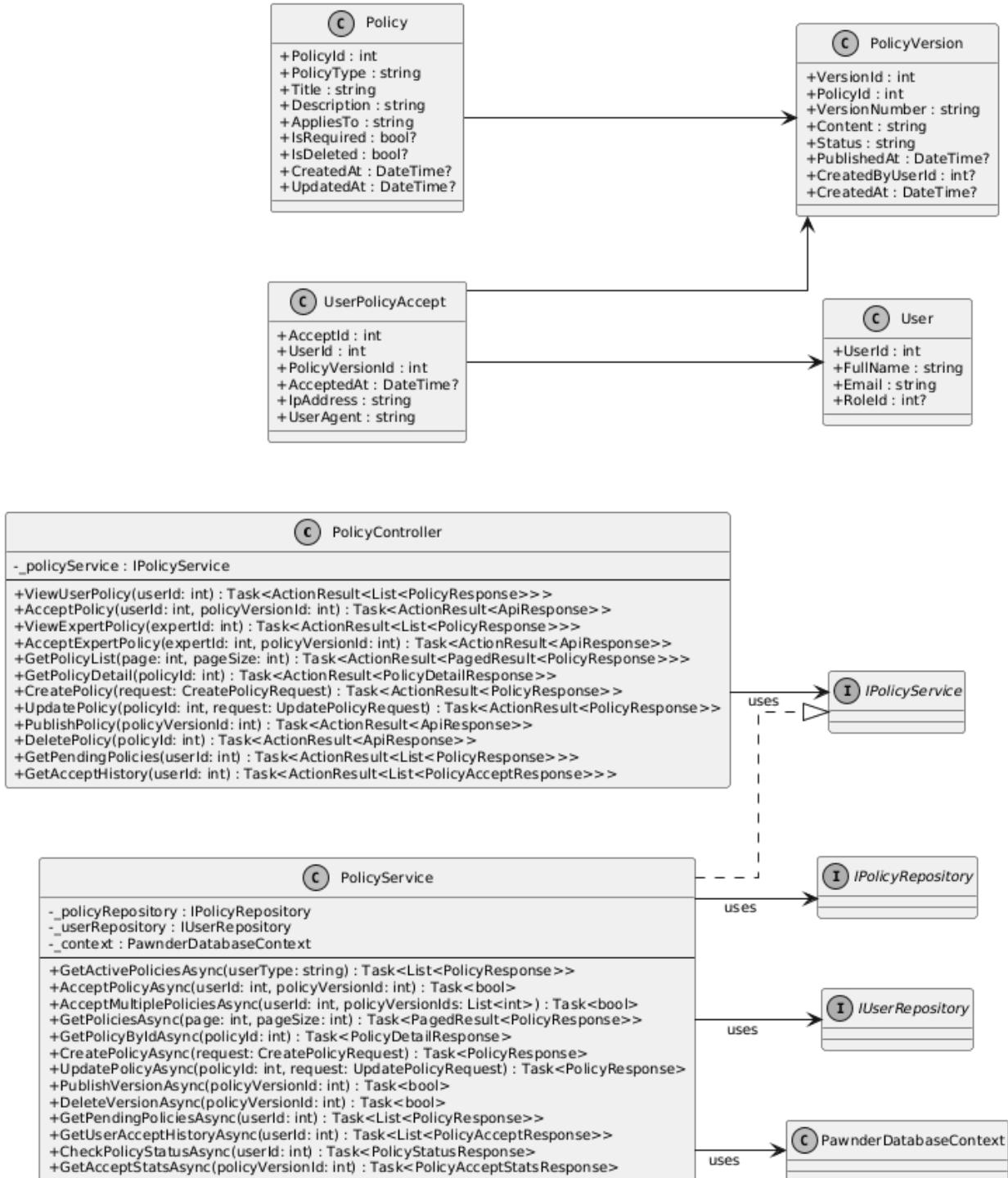


3.11.2.4 View Payment Detail



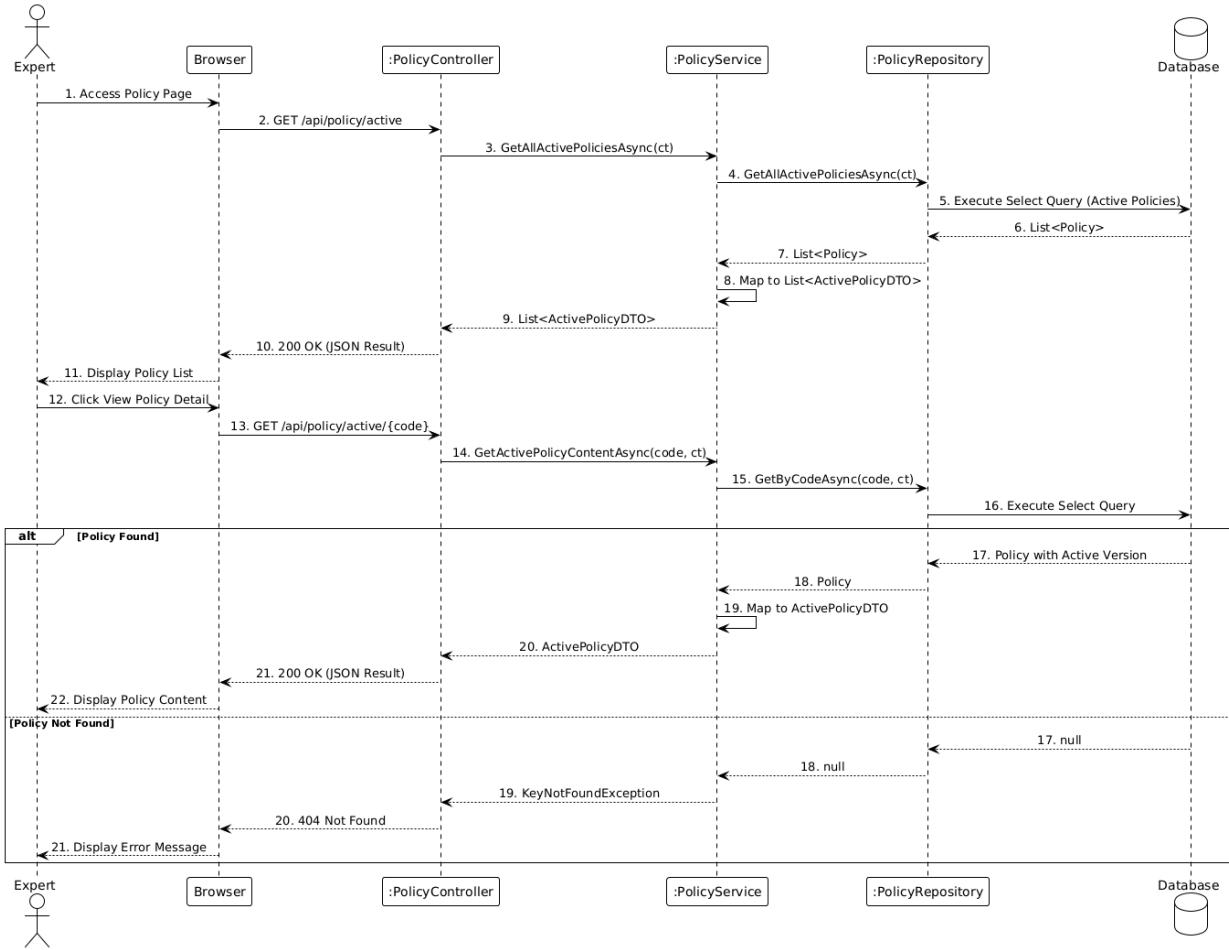
3.12 Policy Management

3.12.1 Class Diagram

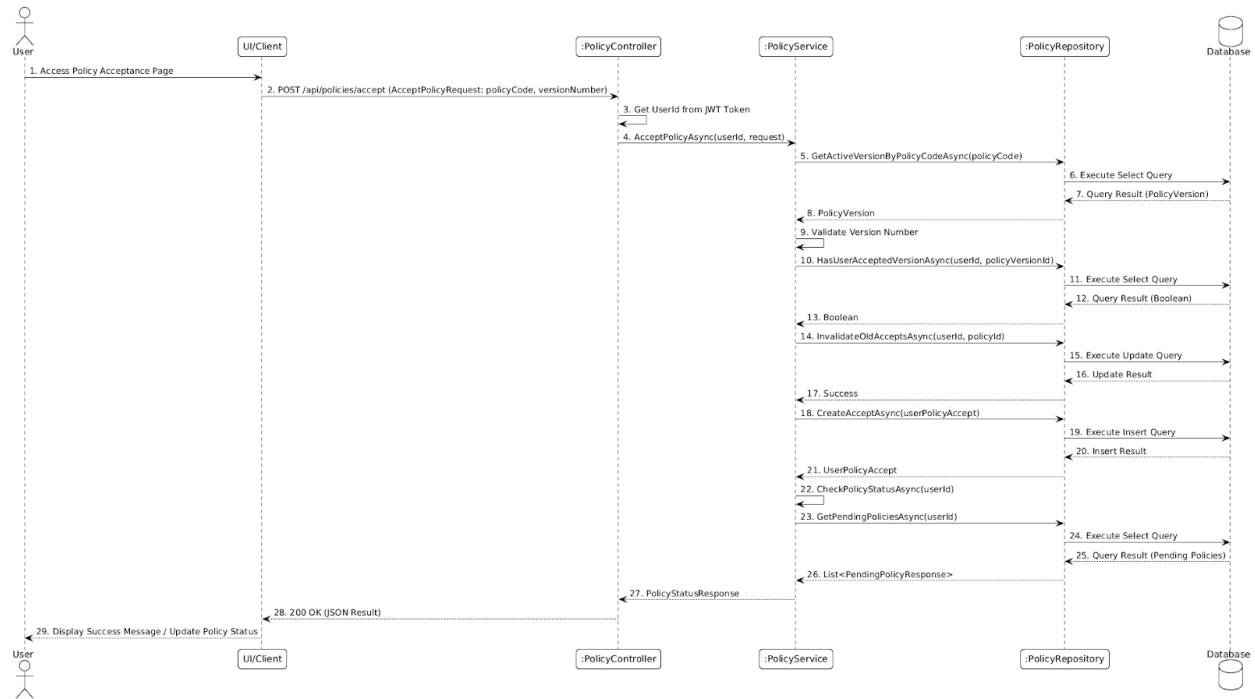


3.12.2 Sequence Diagram

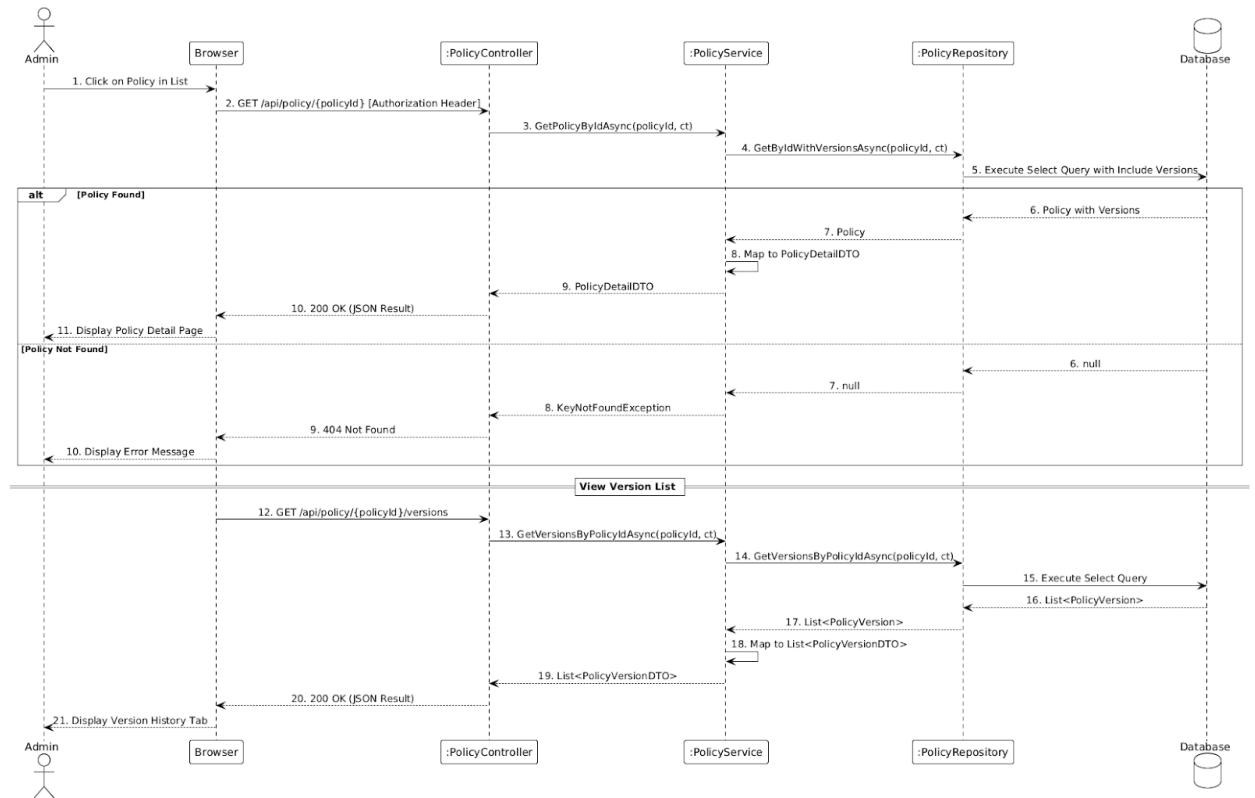
3.12.2.1 View User Policy



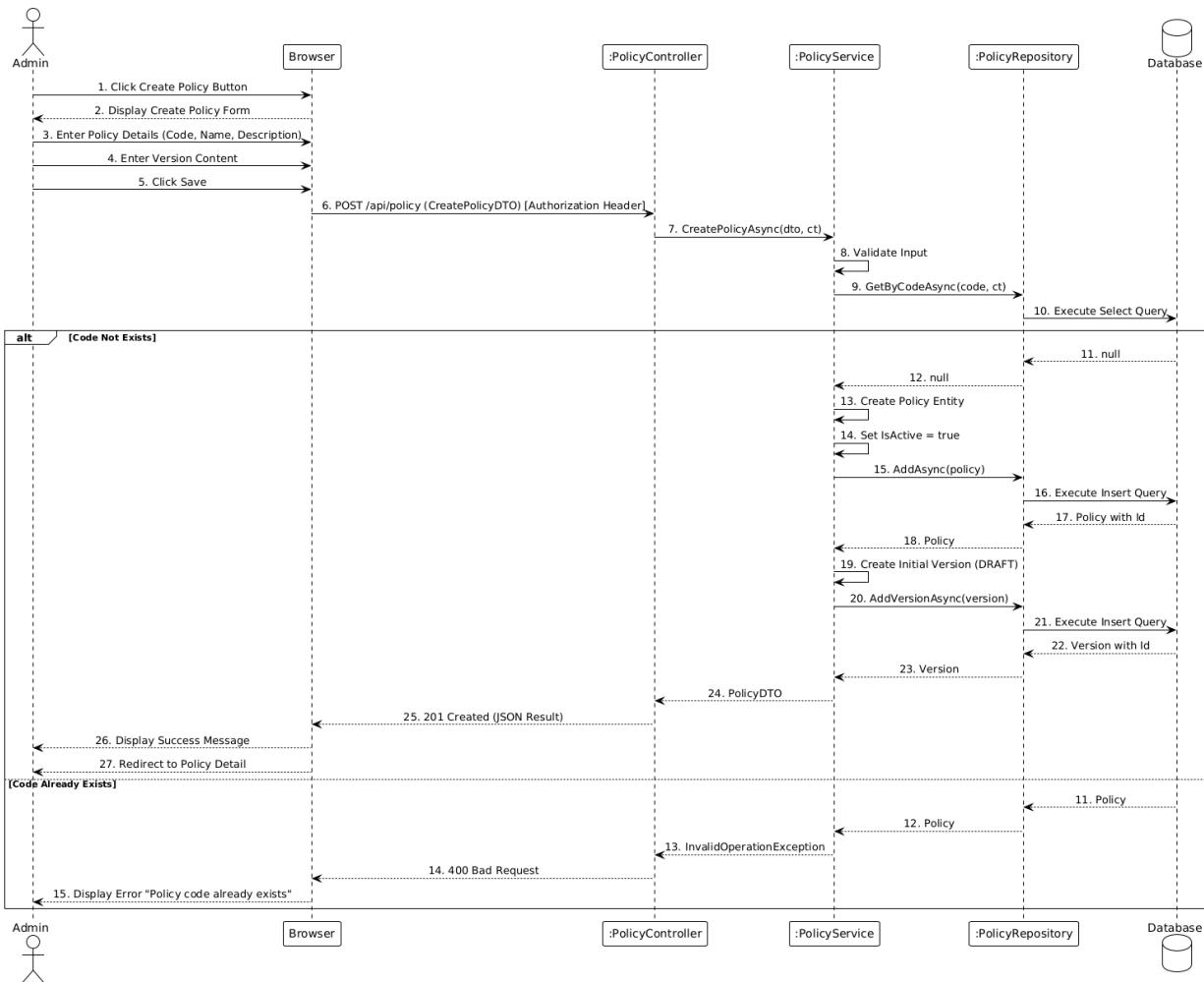
3.12.2.2 Accept Policy



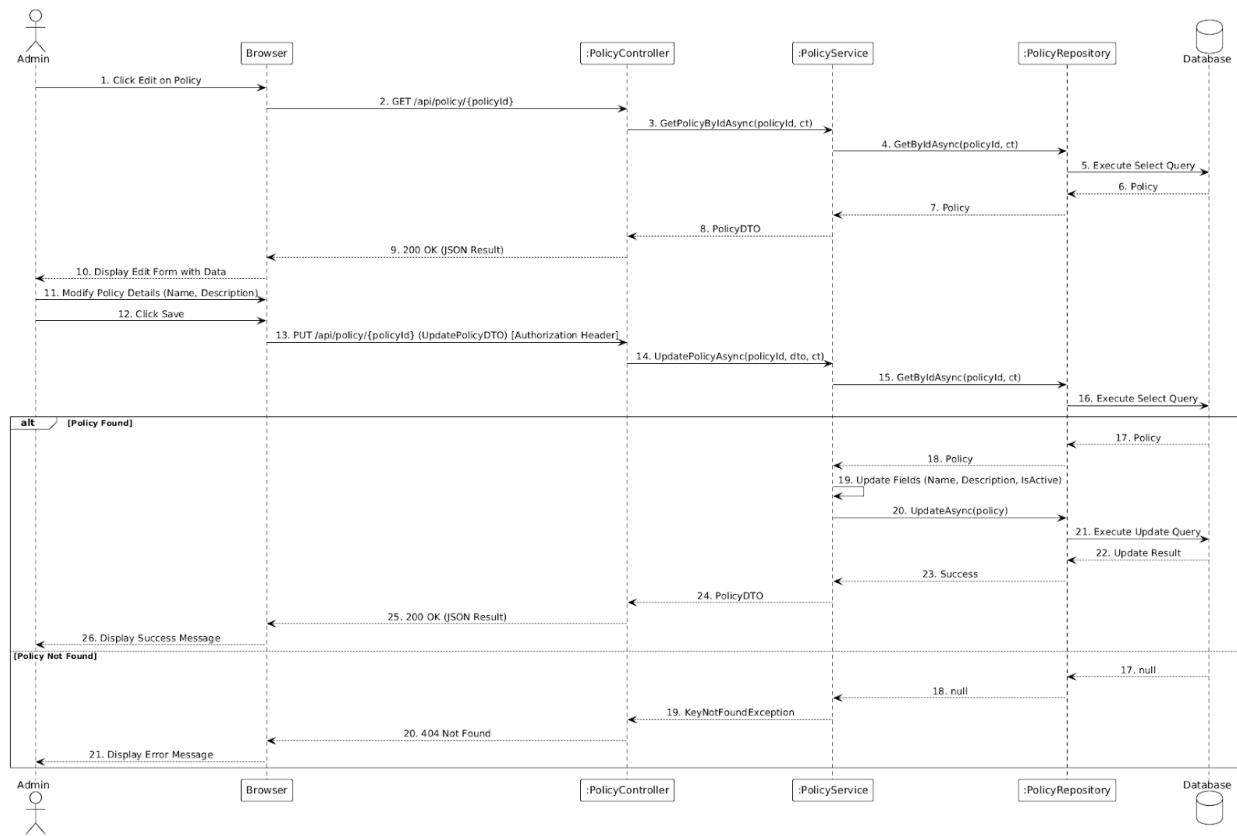
3.12.2.3 View Policy Detail



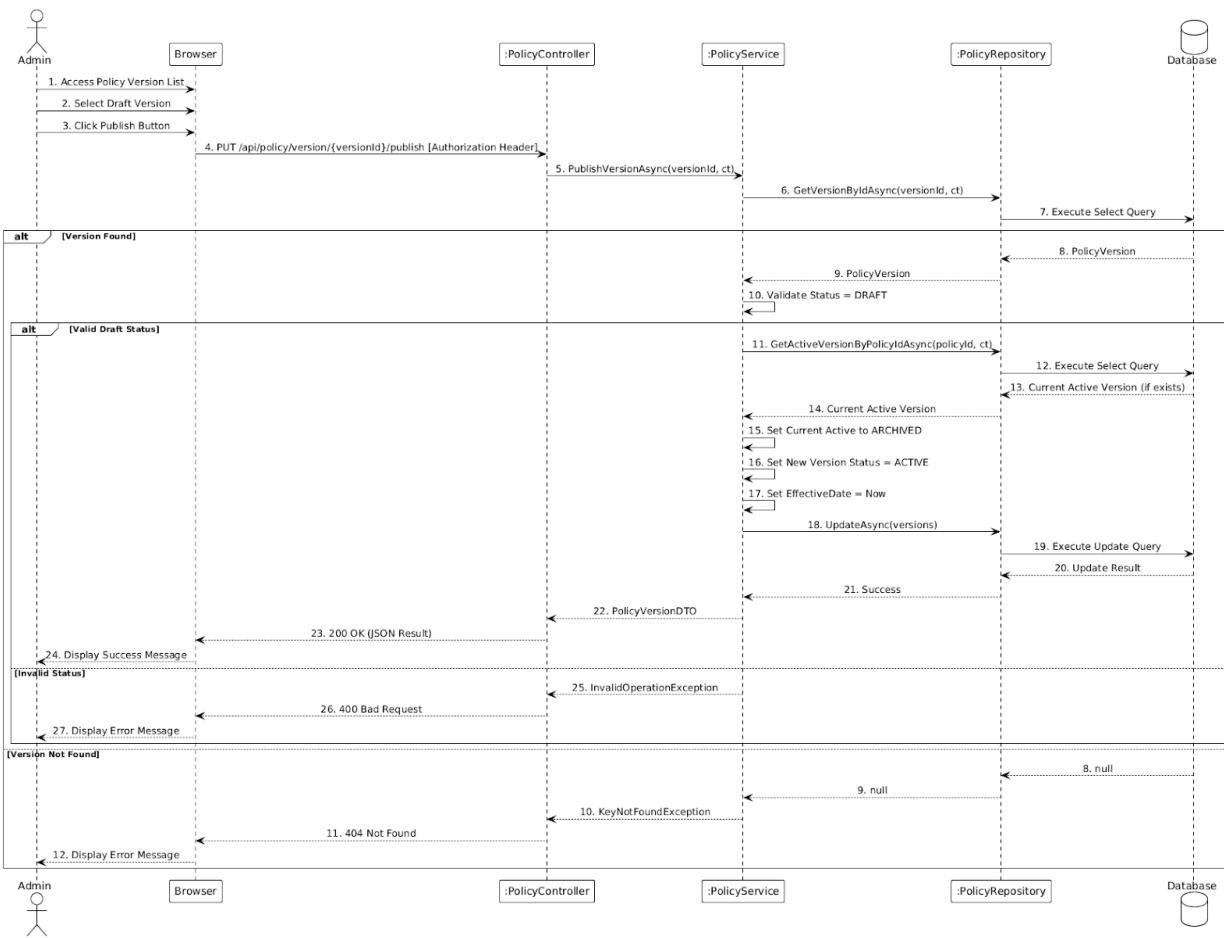
3.12.2.4 Create Policy



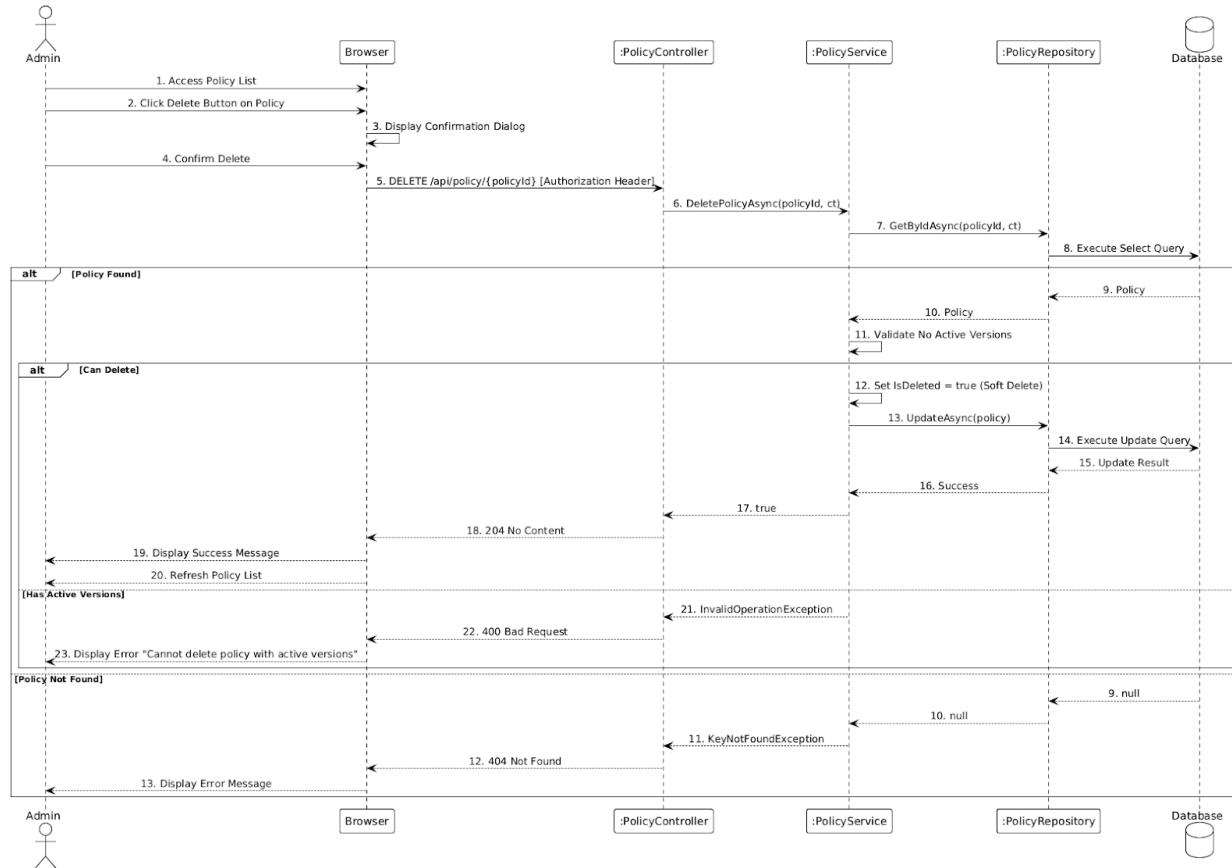
3.12.2.5 Edit Policy



3.12.2.6 Publish Policy

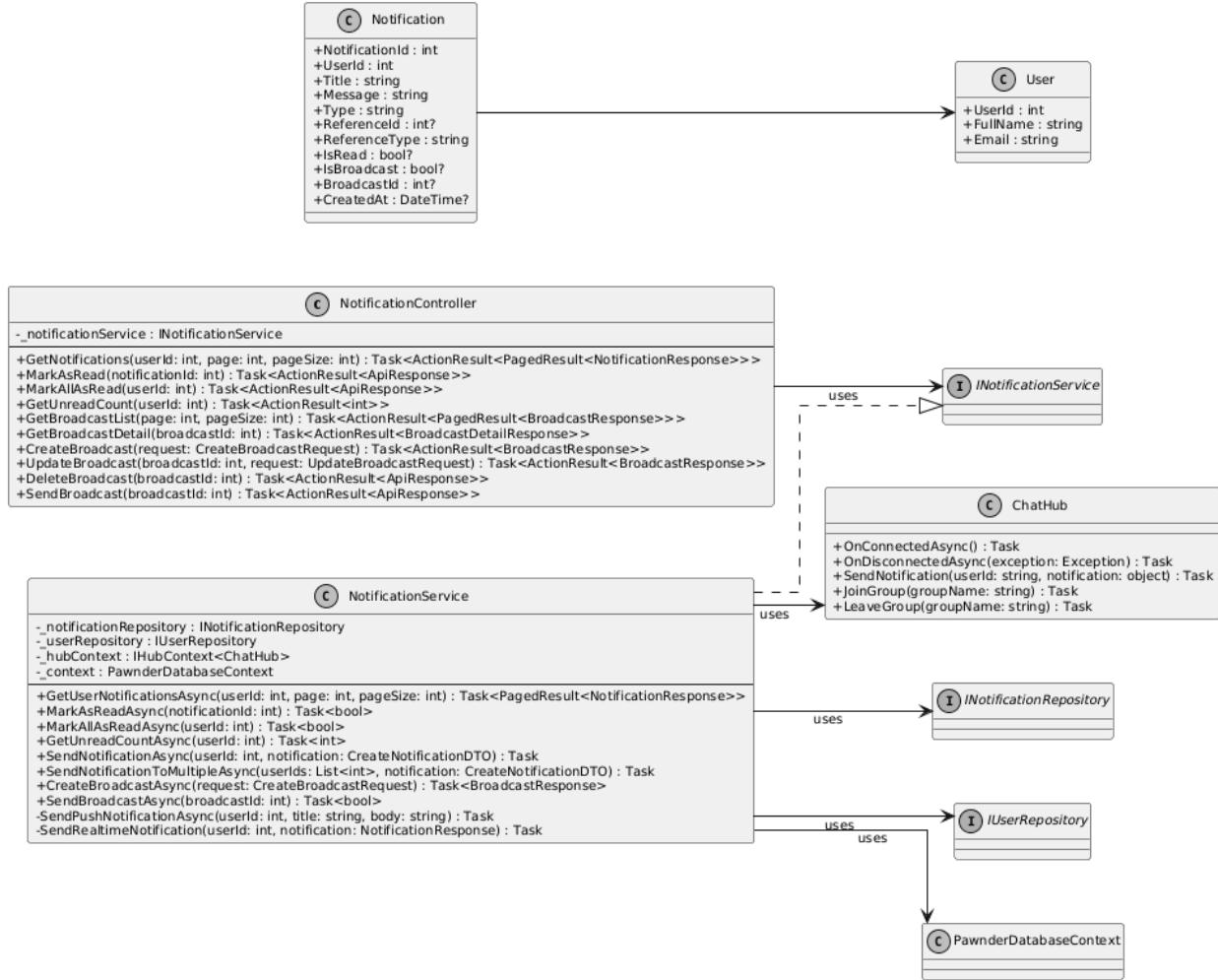


3.12.2.7 Delete Policy



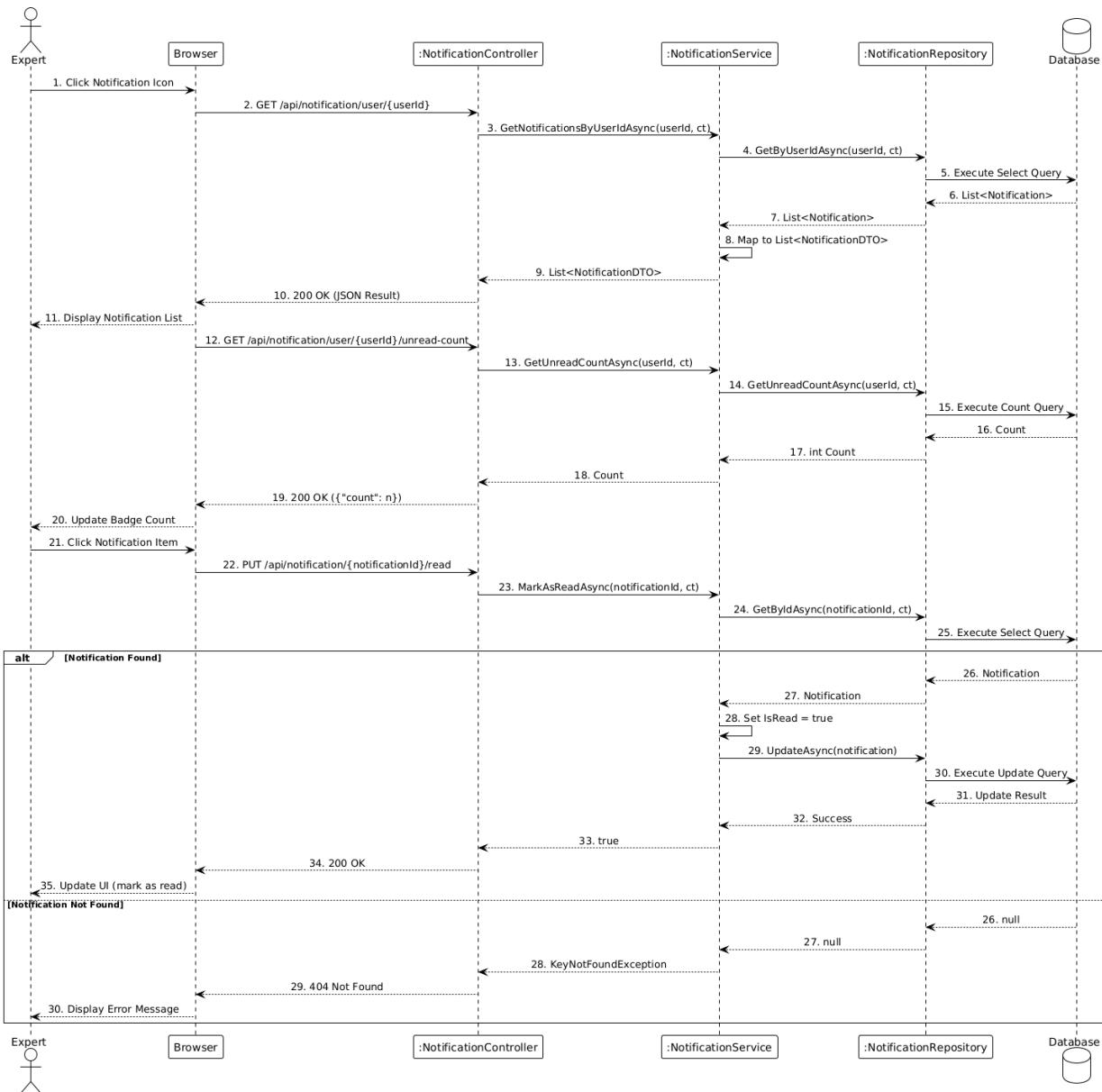
3.13 Notification & Broadcast

3.13.1 Class Diagram

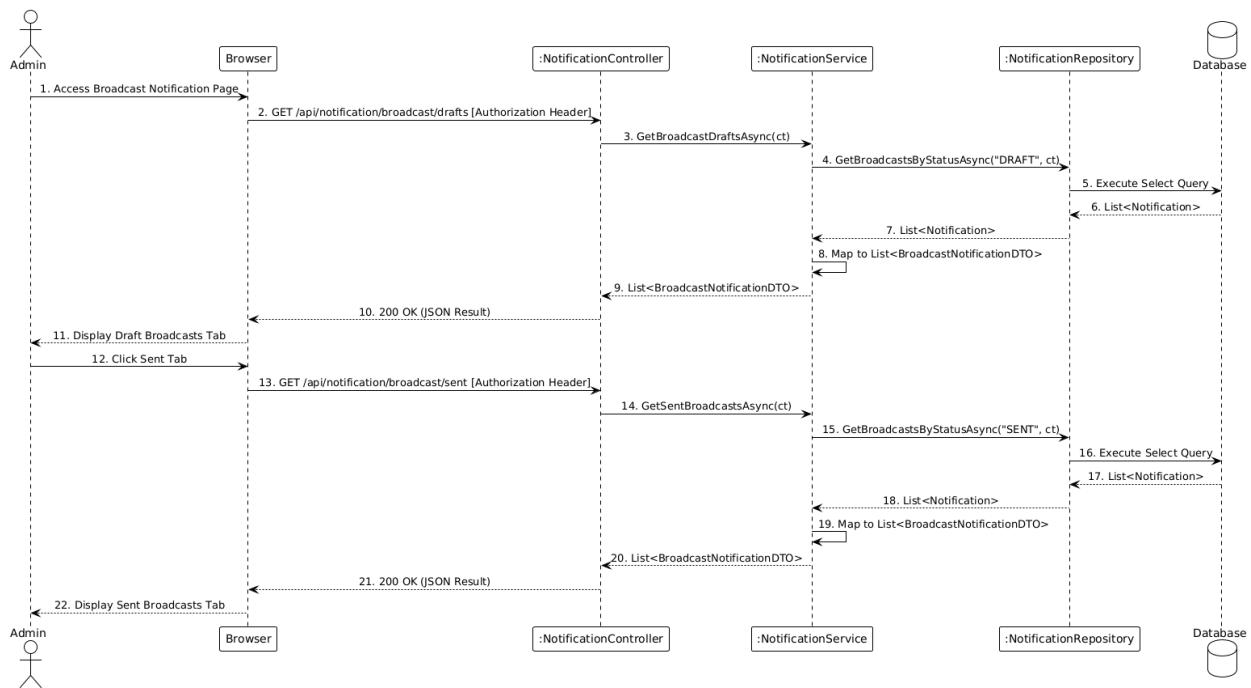


3.13.2 Sequence Diagram

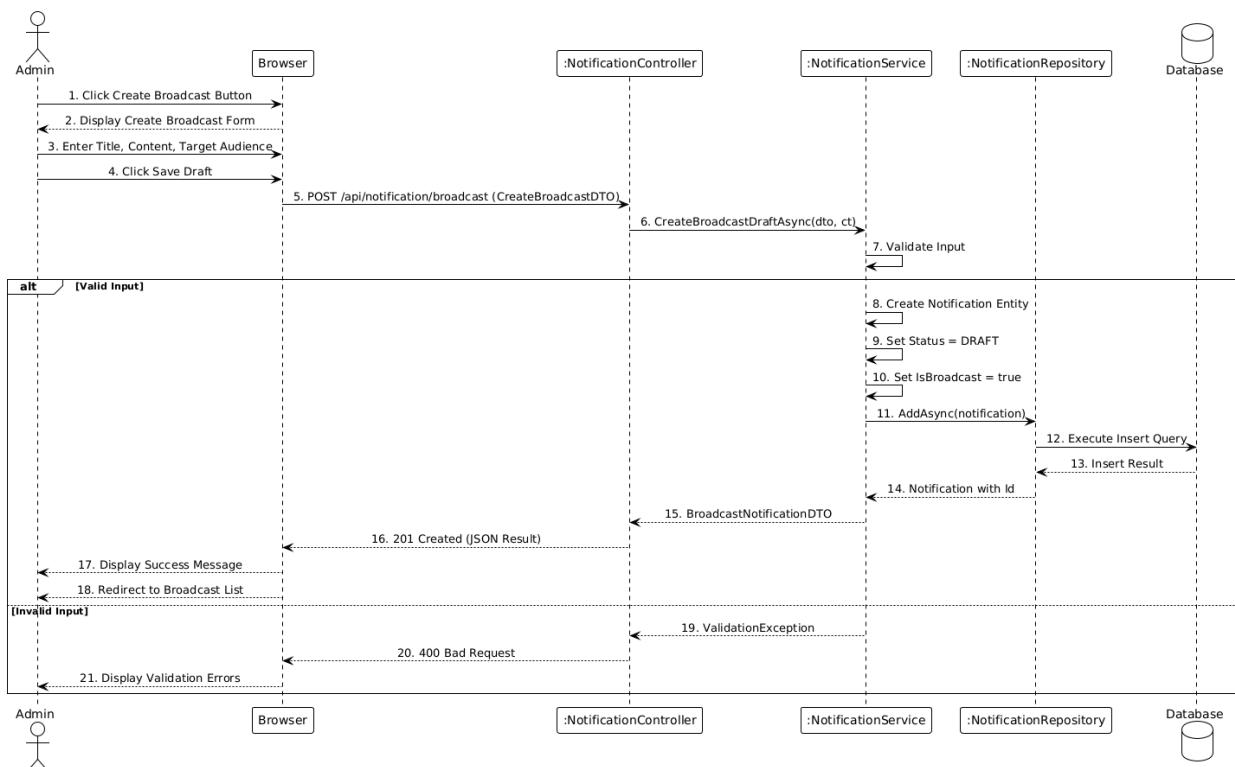
3.13.2.1 View Notification



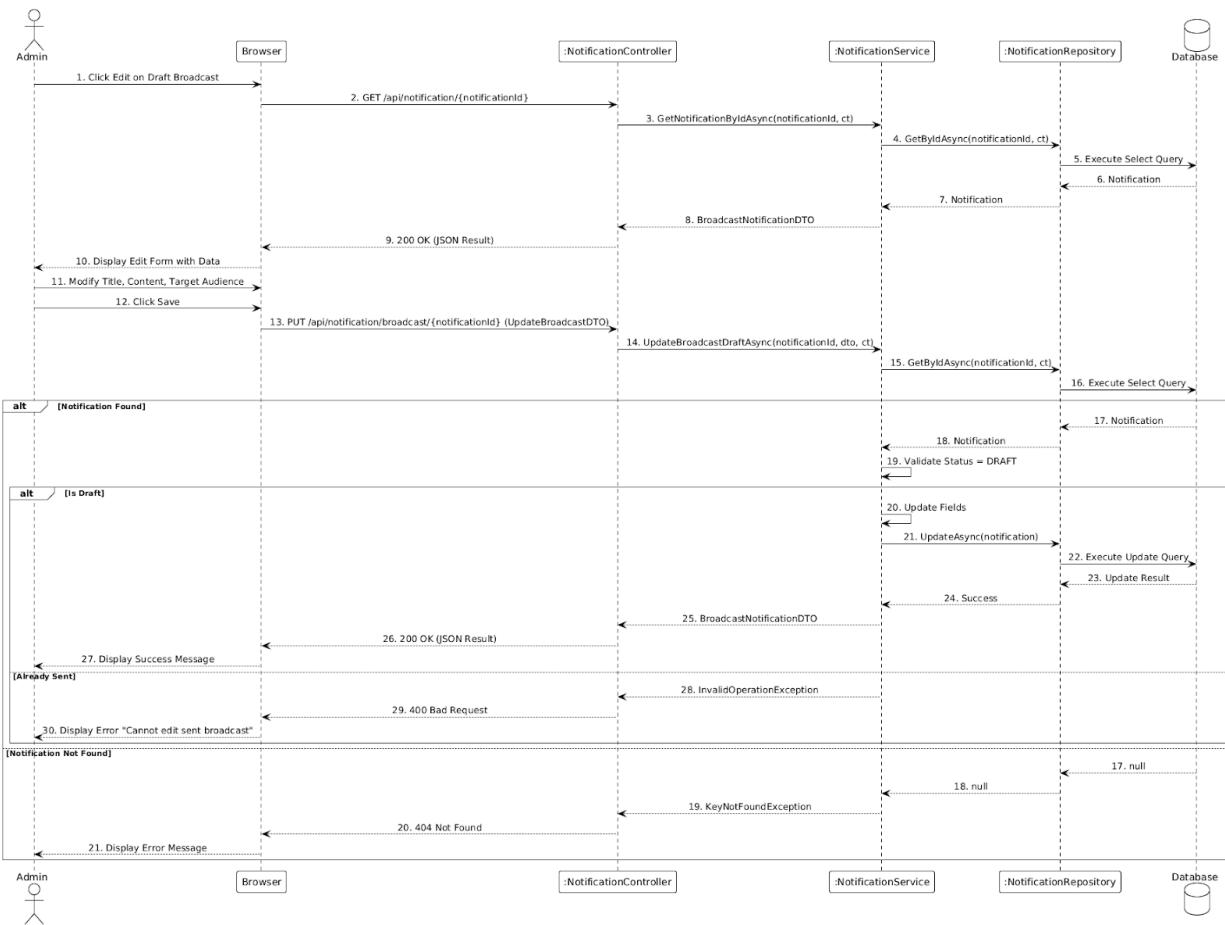
3.13.2.2 View Broadcast List



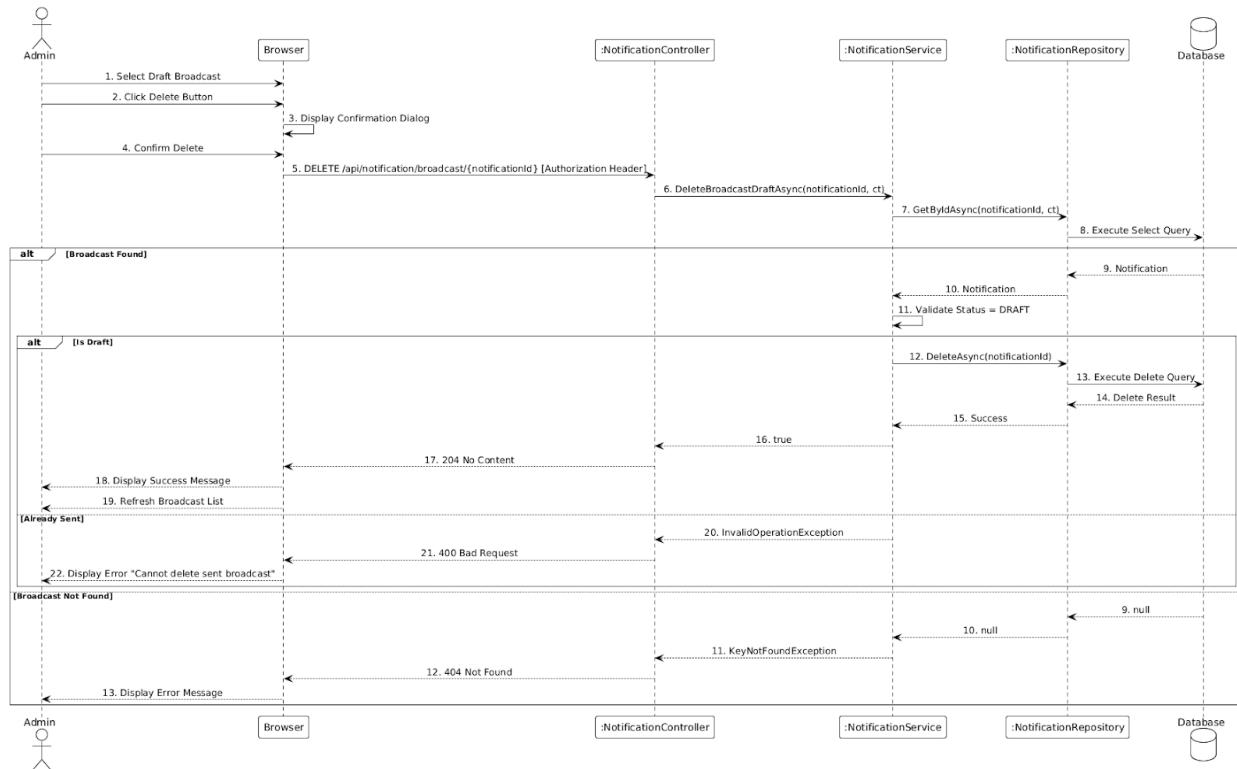
3.13.2.3 Create Broadcast



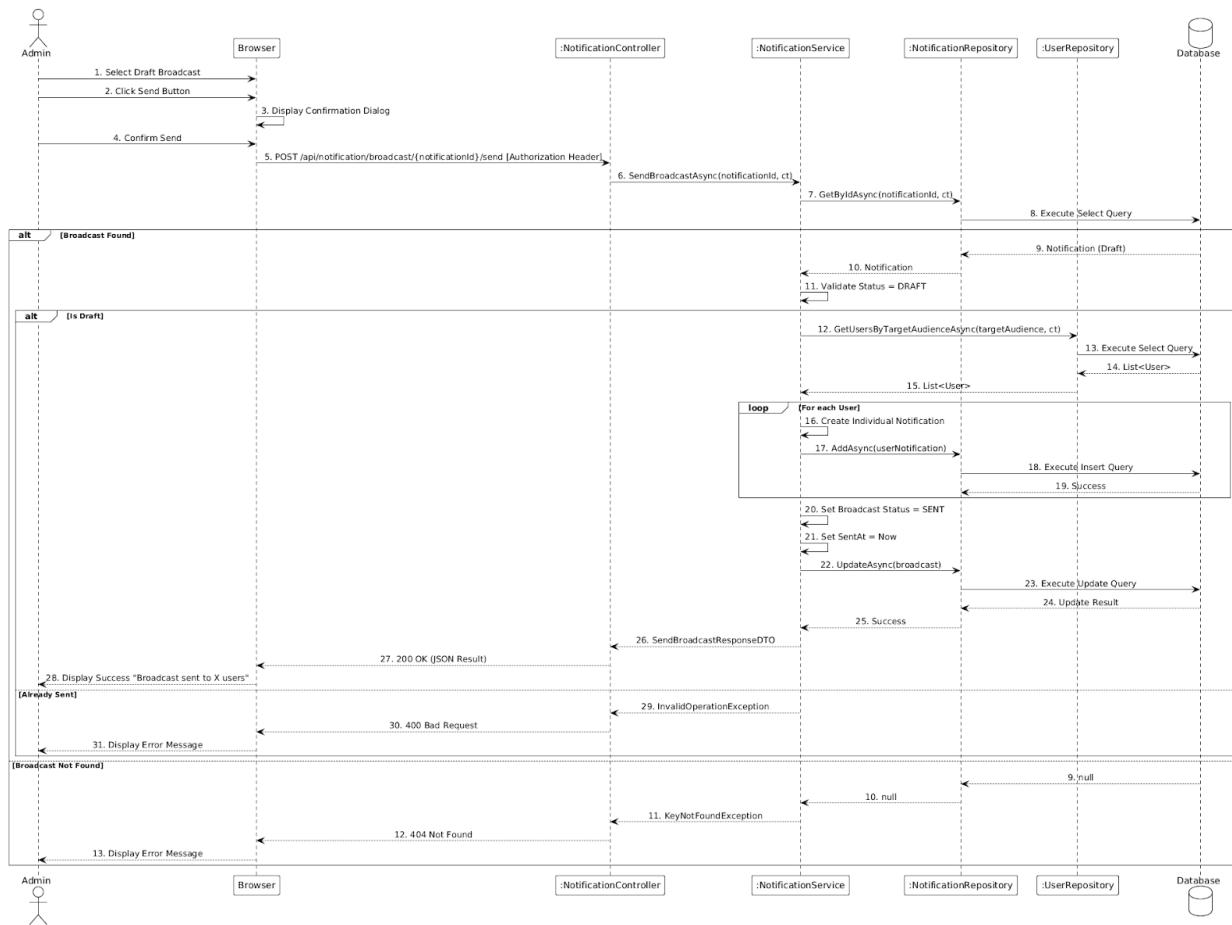
3.13.2.4 Edit Broadcast



3.13.2.5 Delete Broadcast

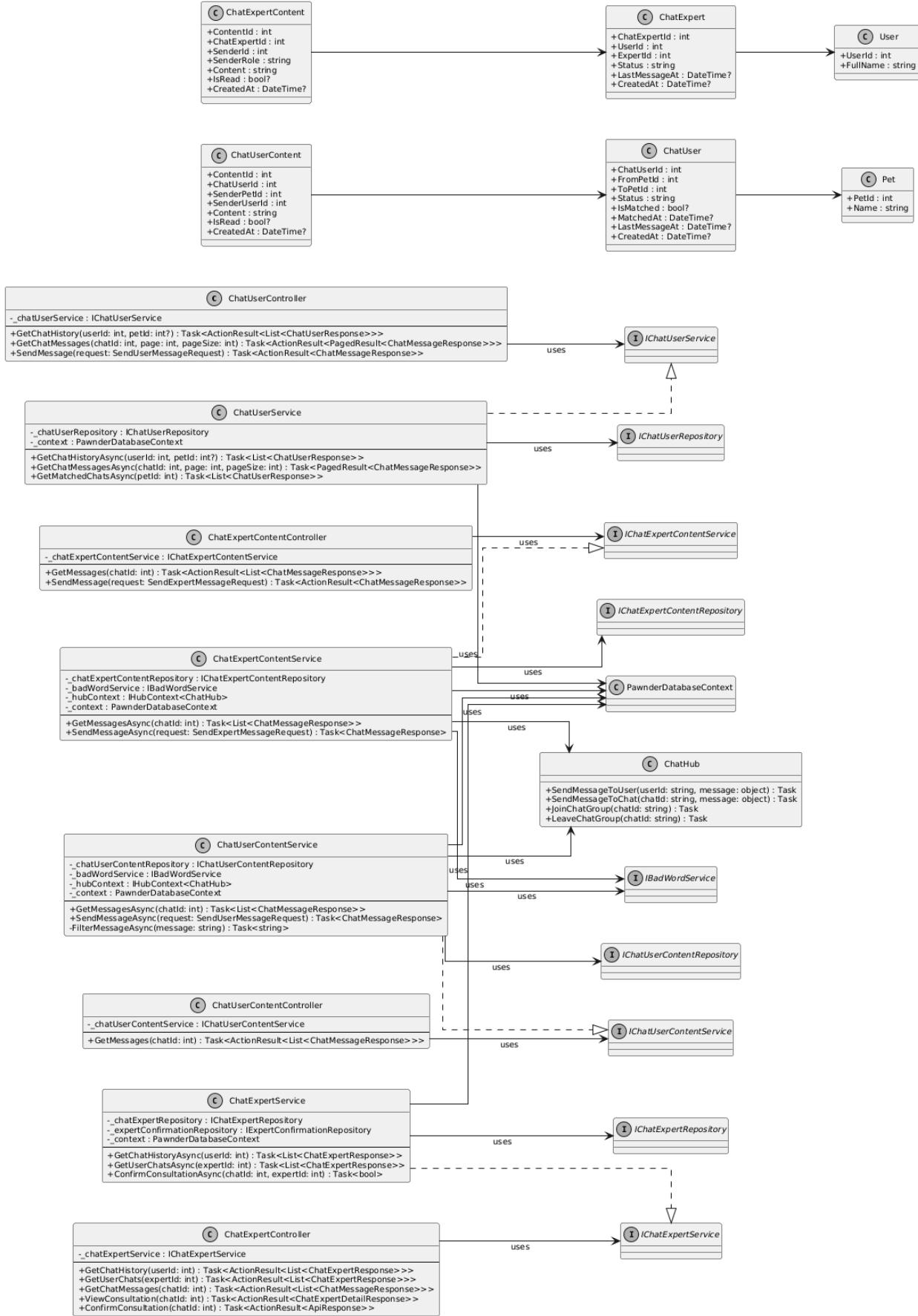


3.13.2.5 Send Broadcast



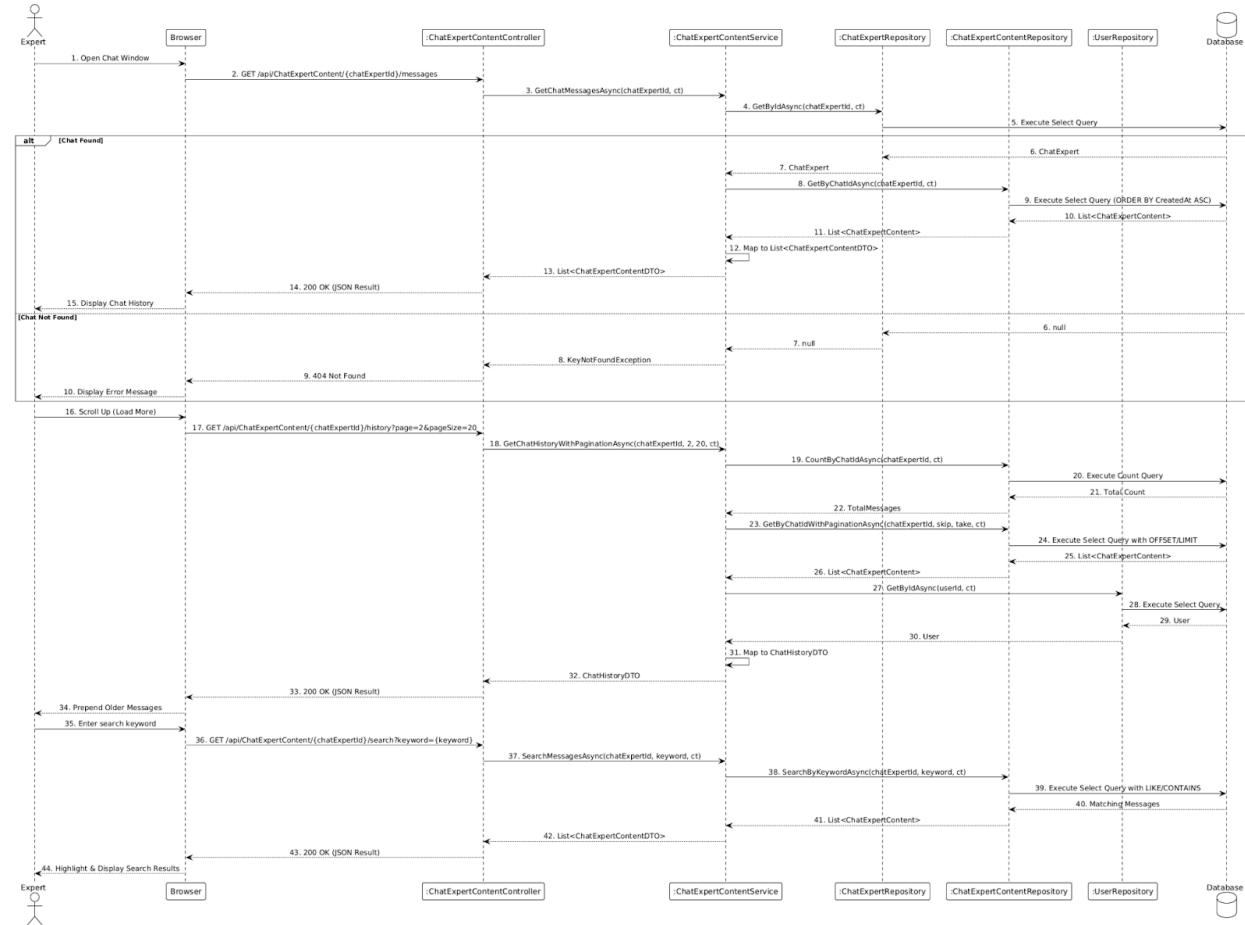
3.14 Chat & Messaging

3.14.1 Class Diagram

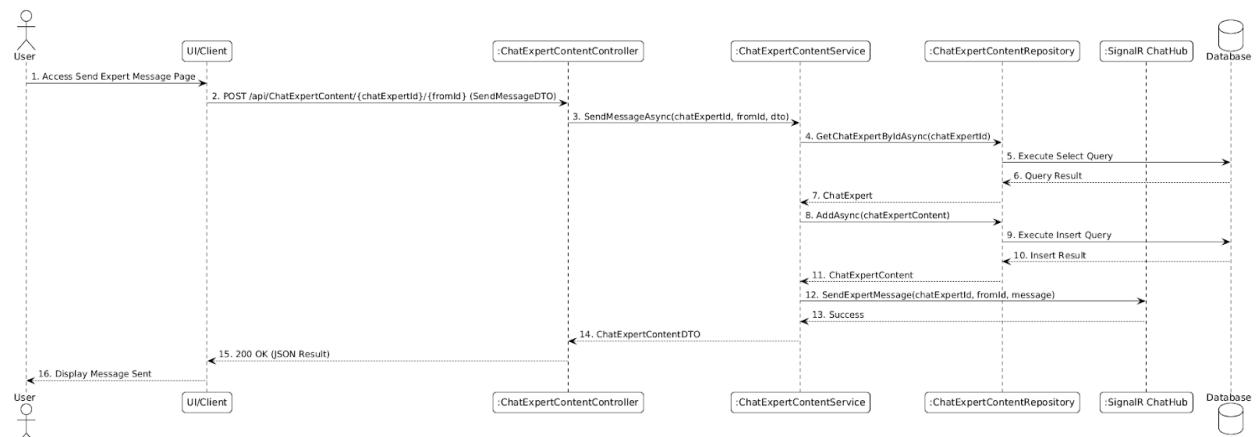


3.14.2 Sequence Diagram

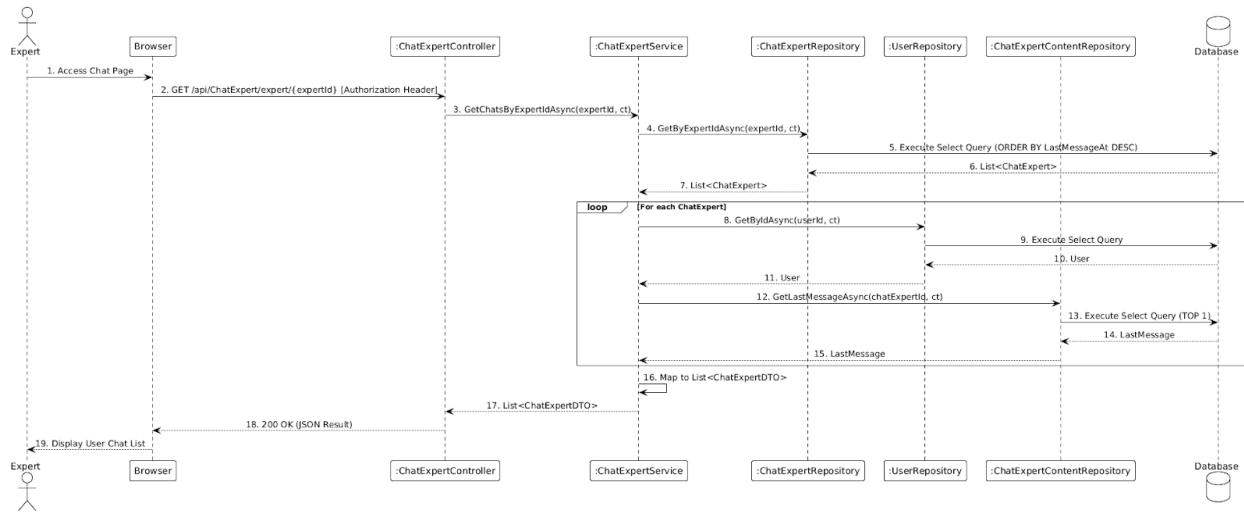
3.14.2.1 View Chat History



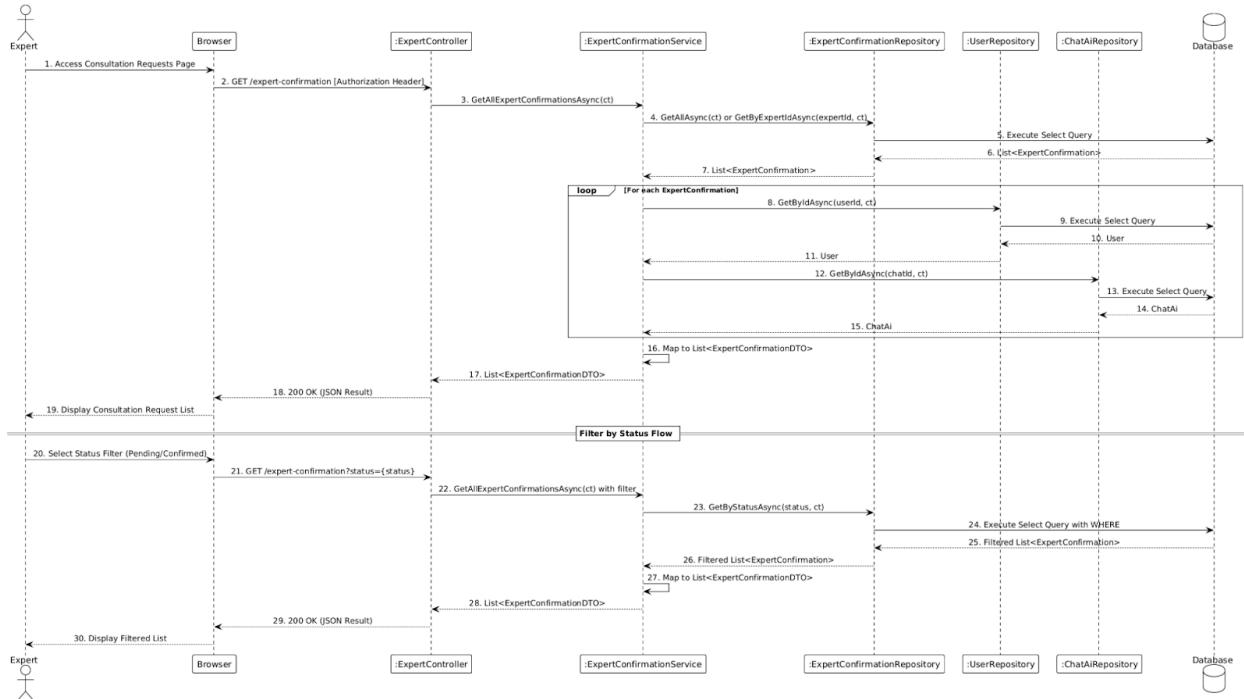
3.14.2.2 Chat with Expert



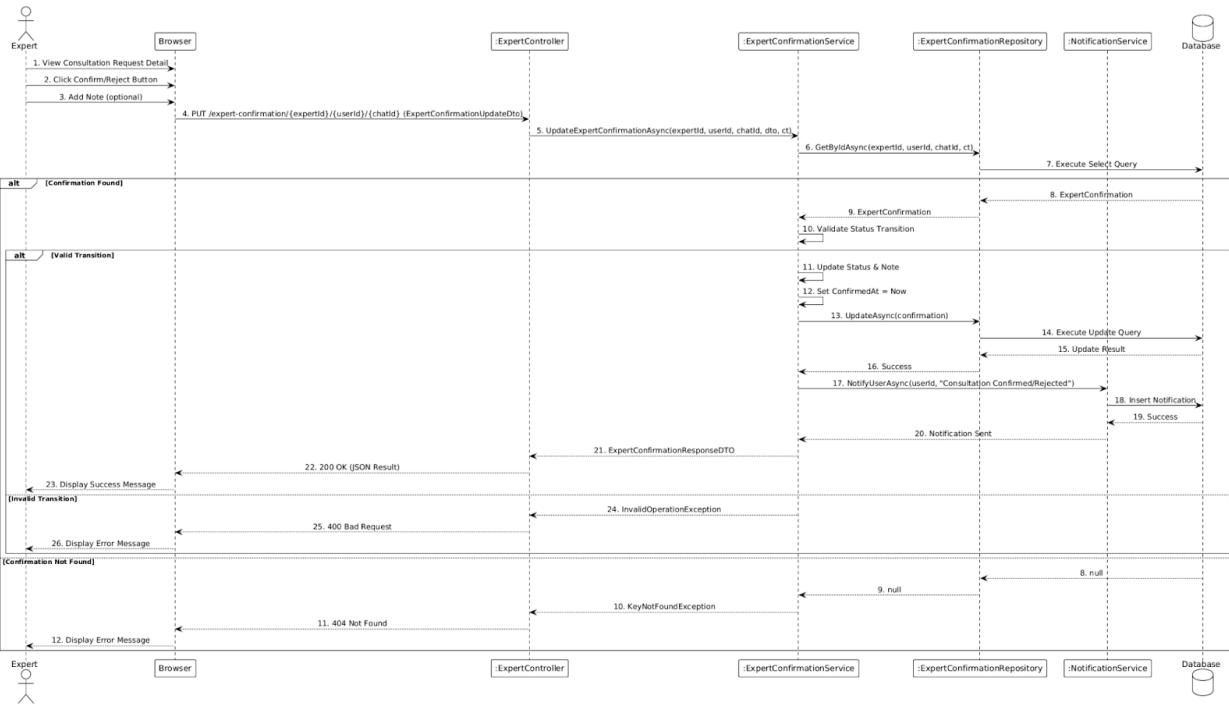
3.14.2.5 View User Chat List



3.14.2.6 View Consultation

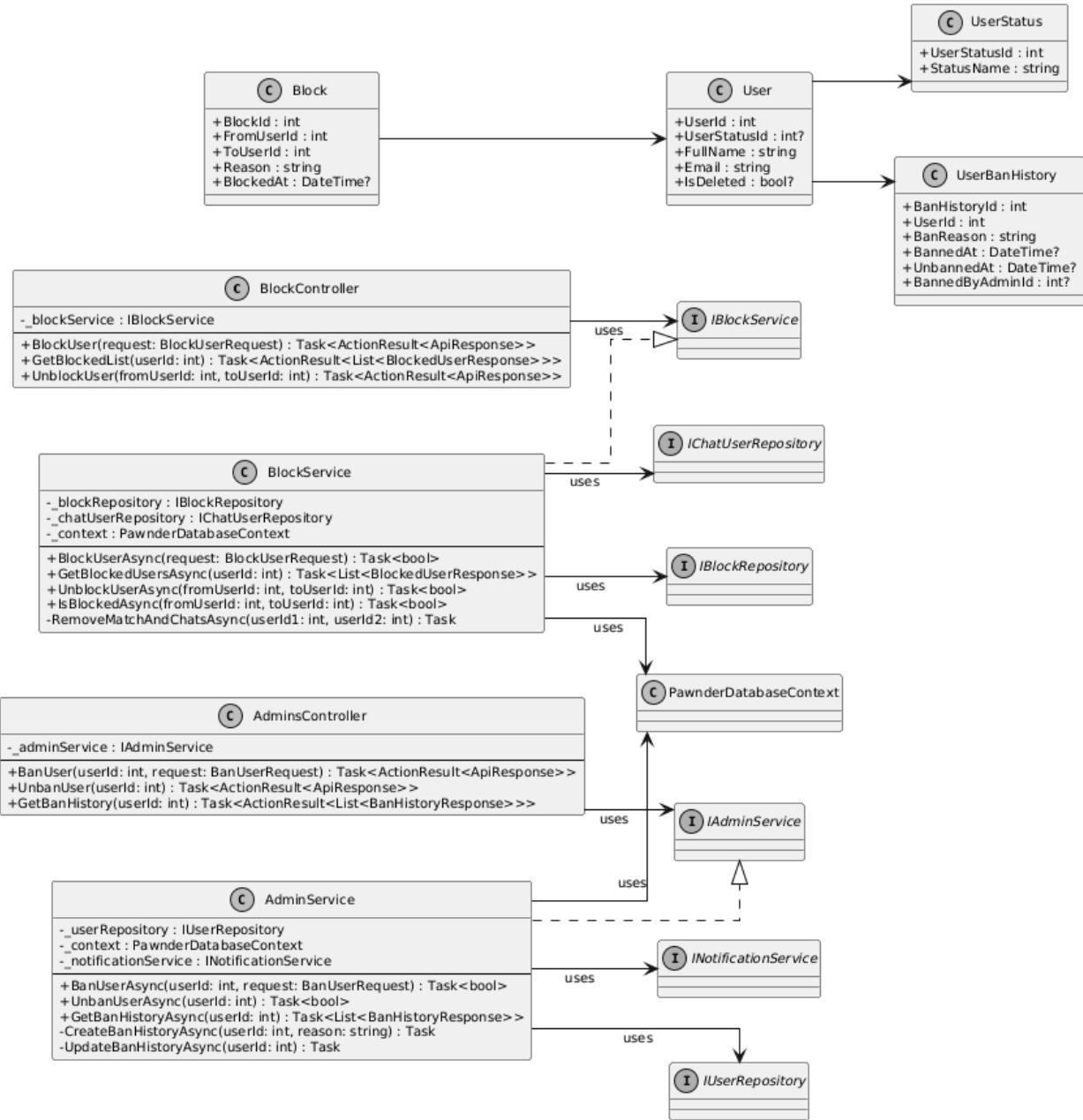


3.14.2.7 Confirm Consultation



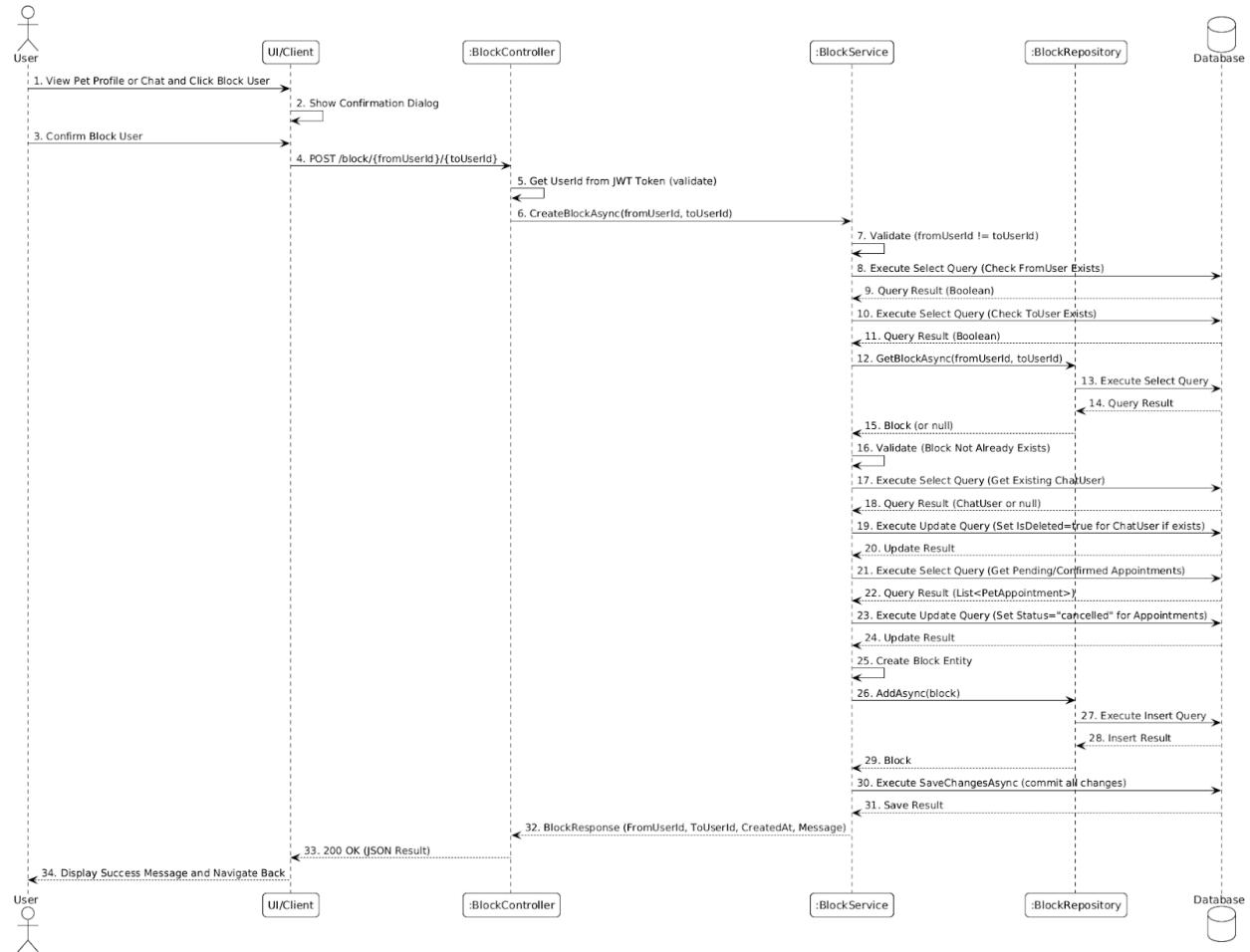
3.15 Block & Privacy

3.15.1 Class Diagram

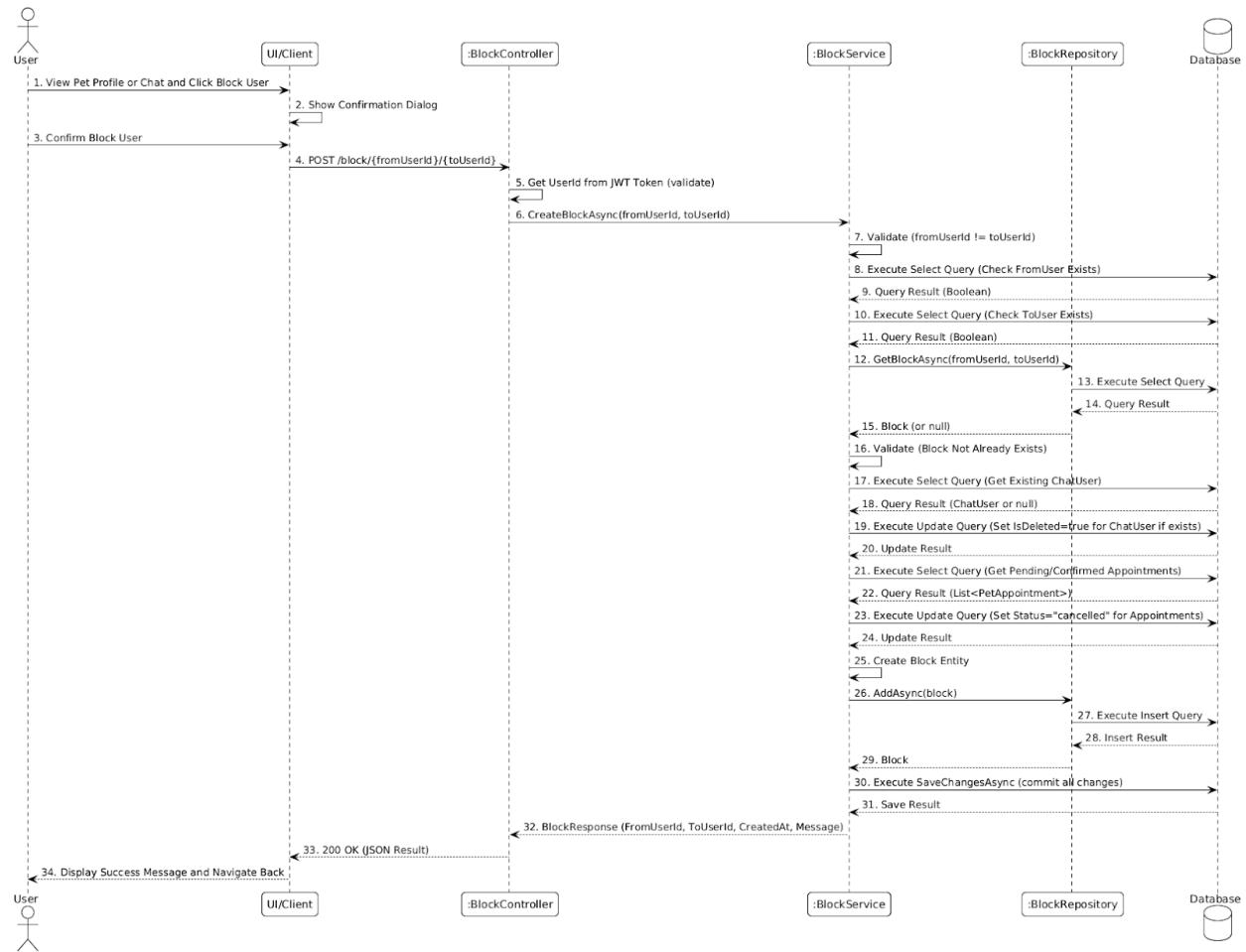


3.15.2 Sequence Diagram

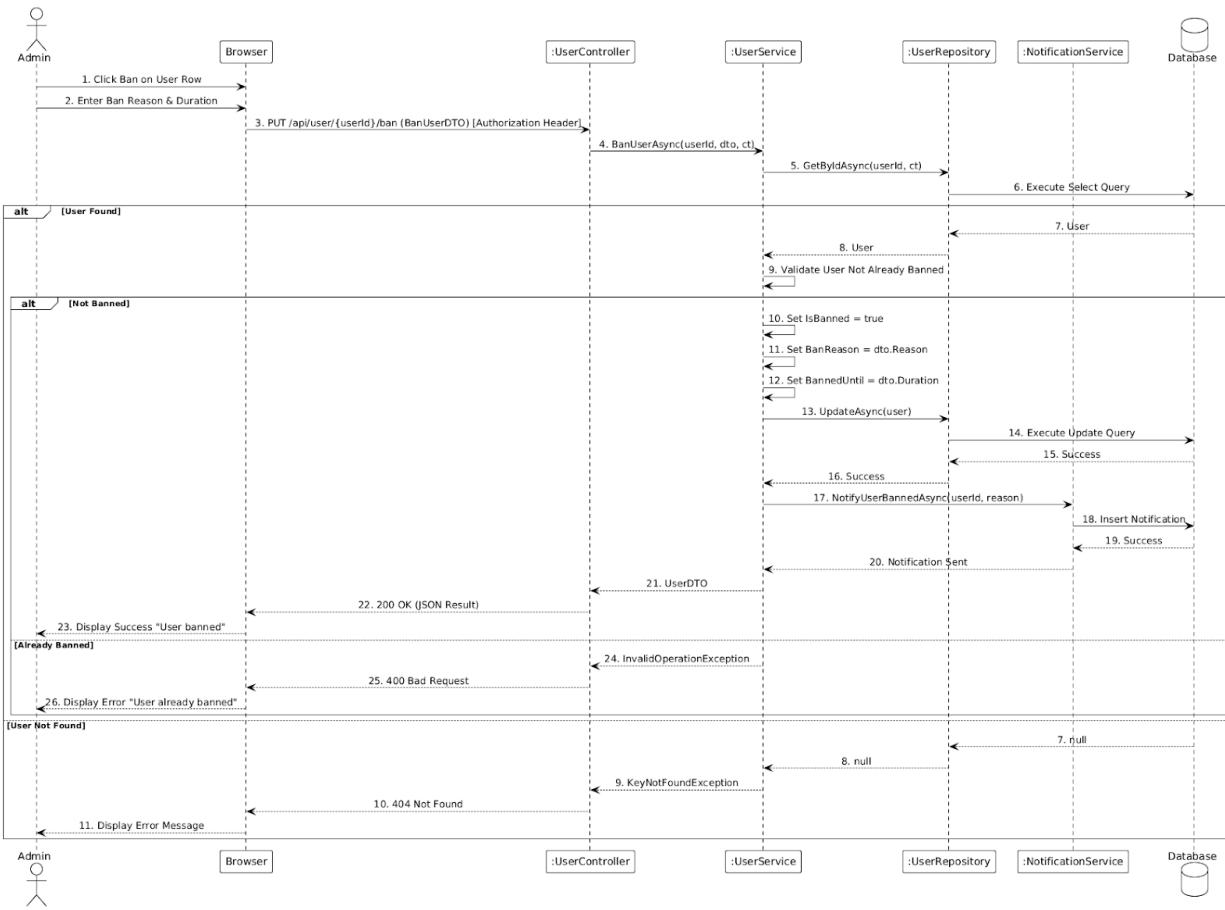
3.15.2.1 Block User



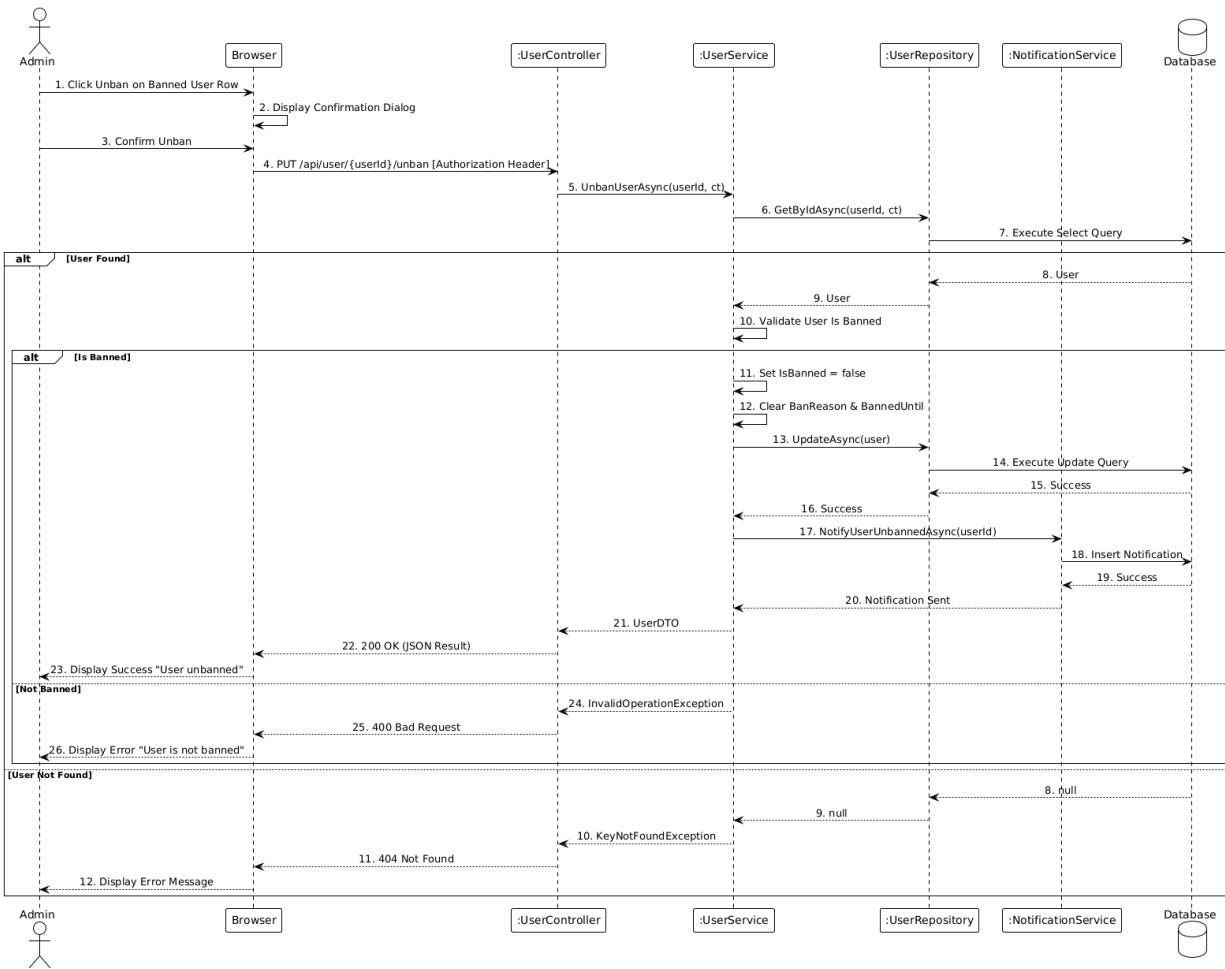
3.15.2.2 View Blocked List



3.15.2.3 Ban User

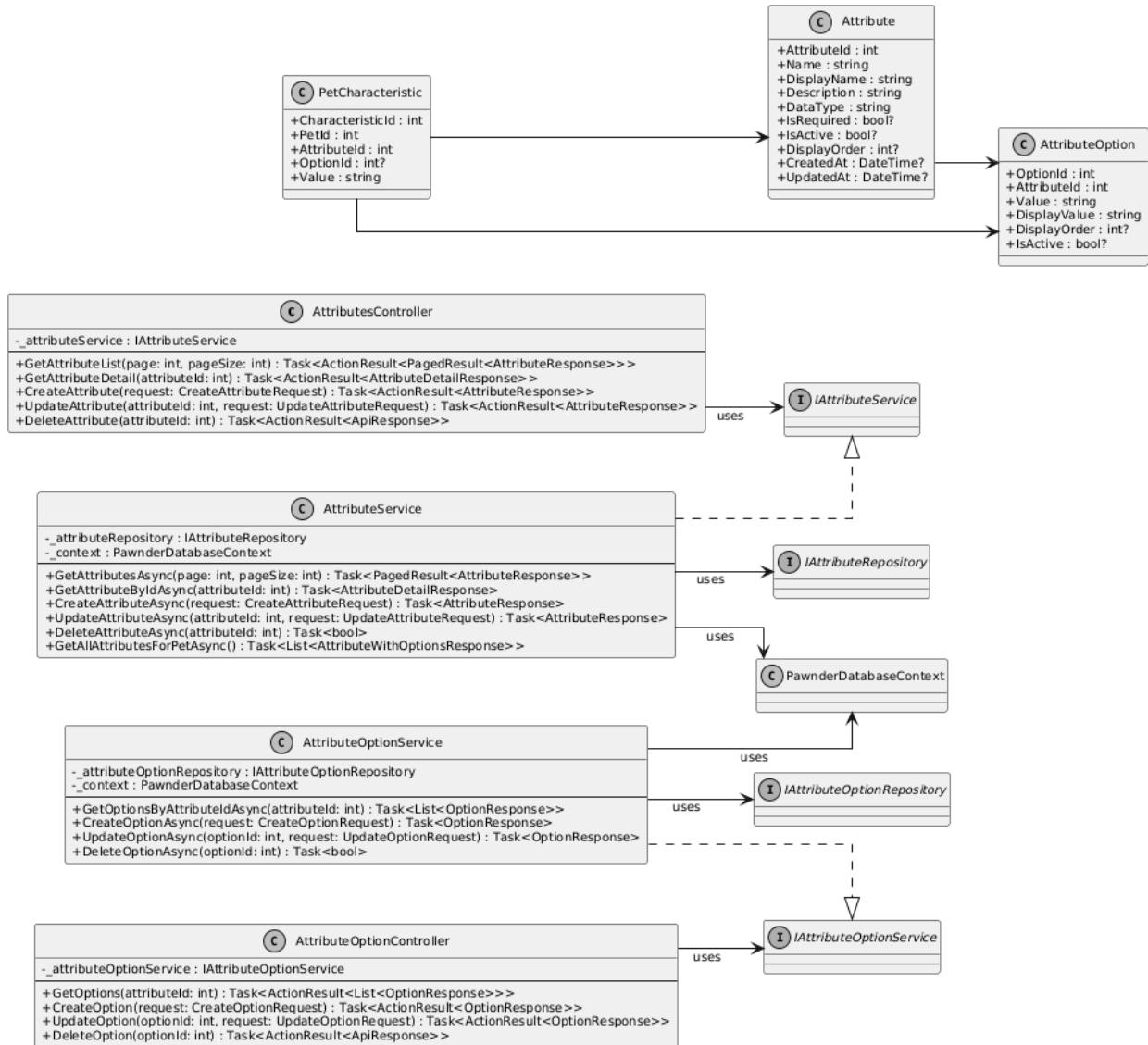


3.15.2.4 Unban User



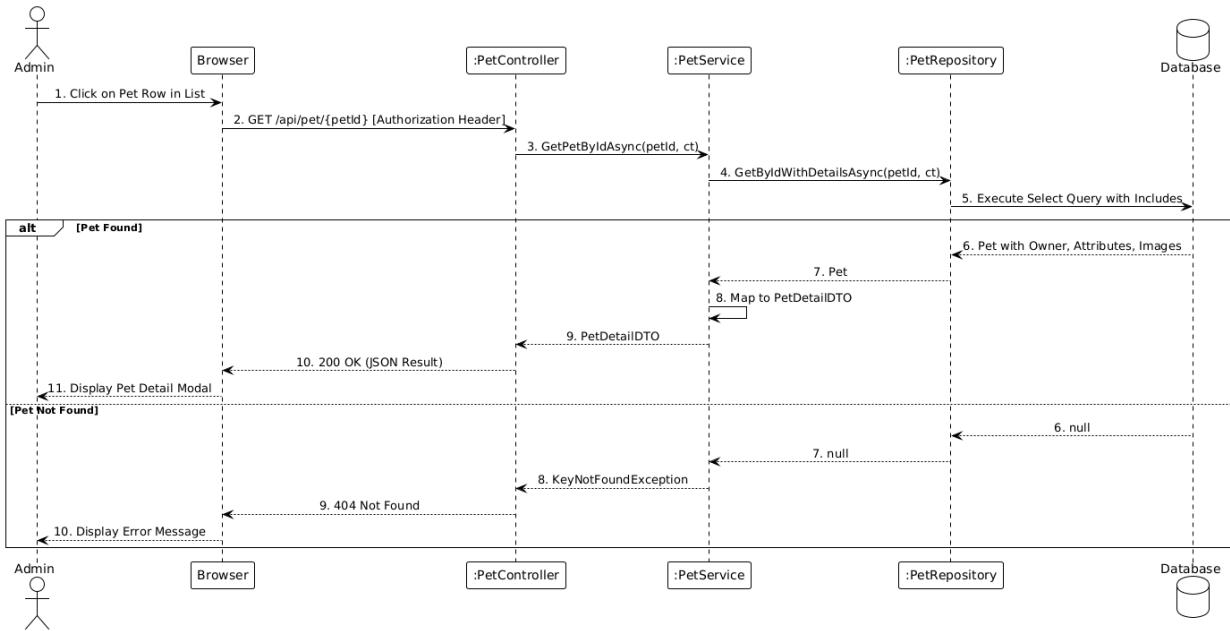
3.16 Attribute Management

3.16.1 Class Diagram

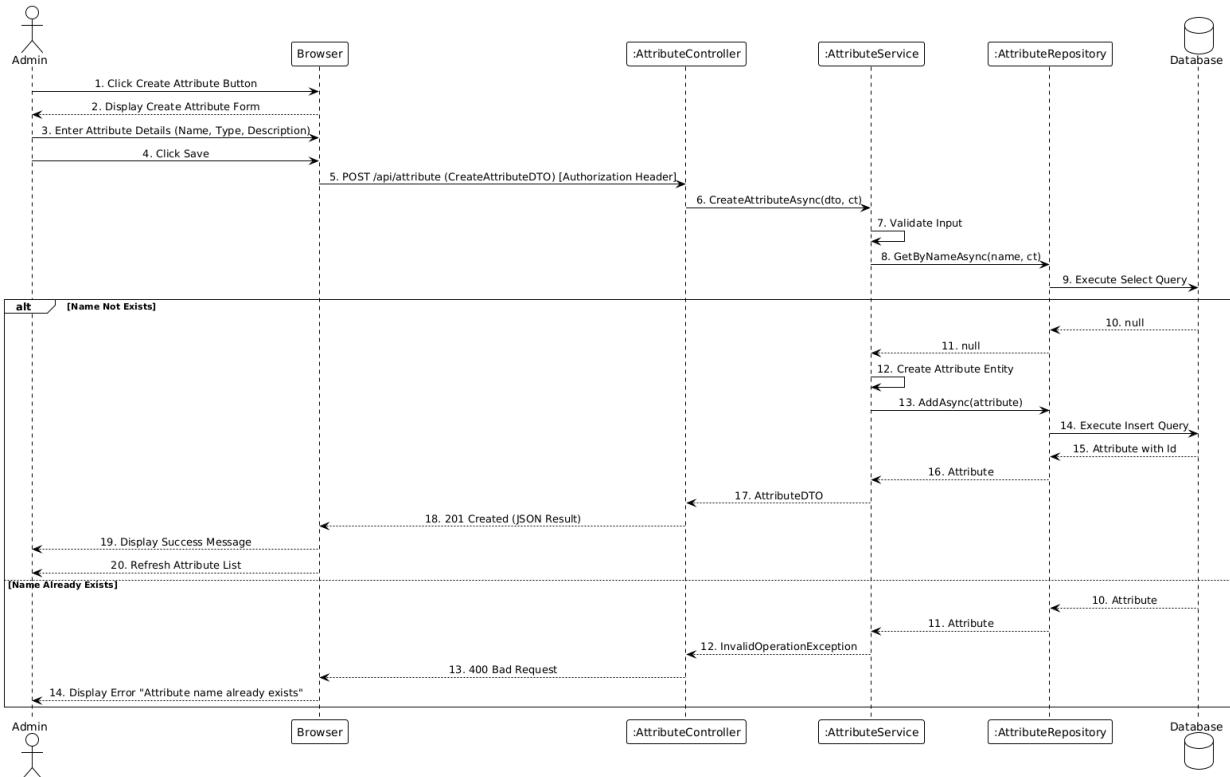


3.16.2 Sequence Diagram

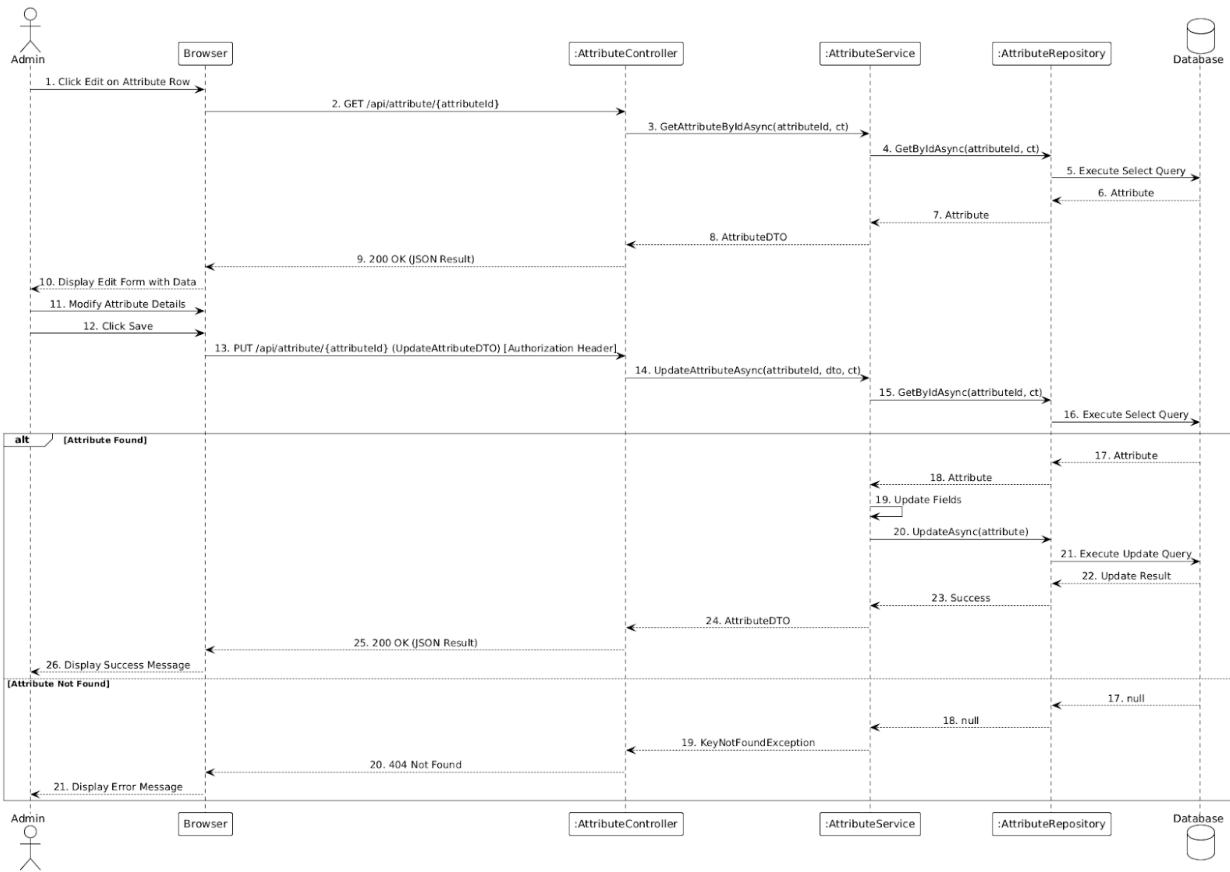
3.16.2.1 View Attribute List



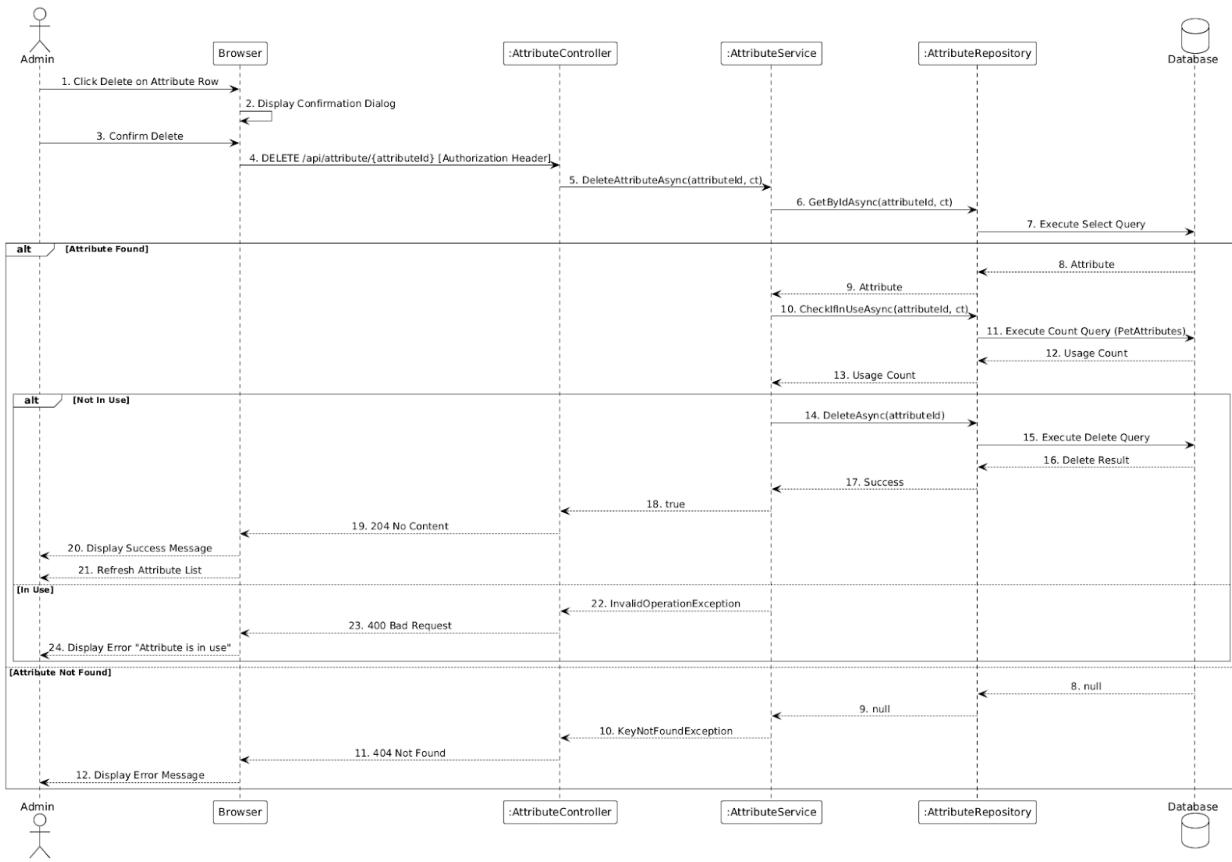
3.16.2.2 Create Attribute



3.16.2.3 Edit Attribute

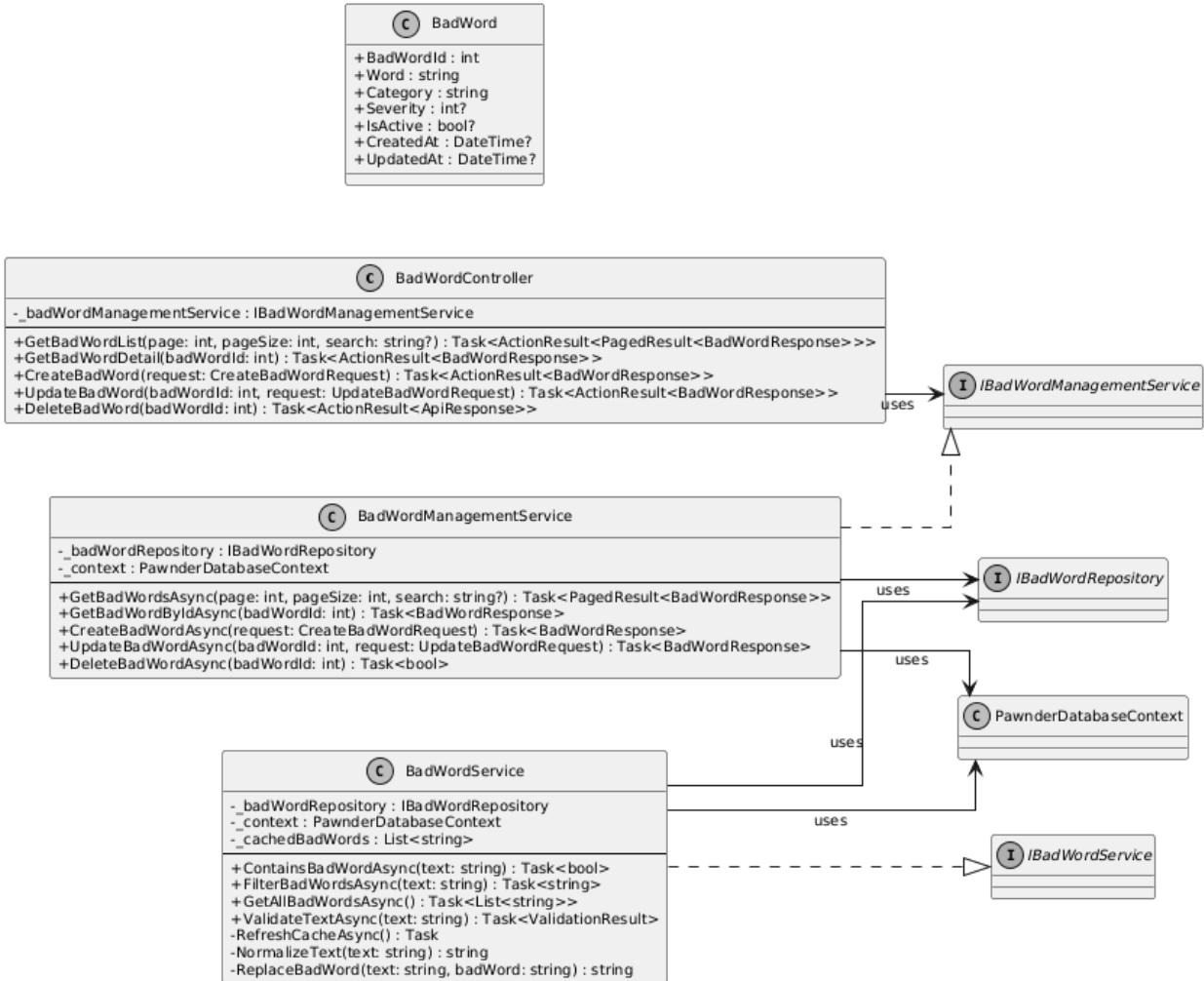


3.16.2.4 Delete Attribute



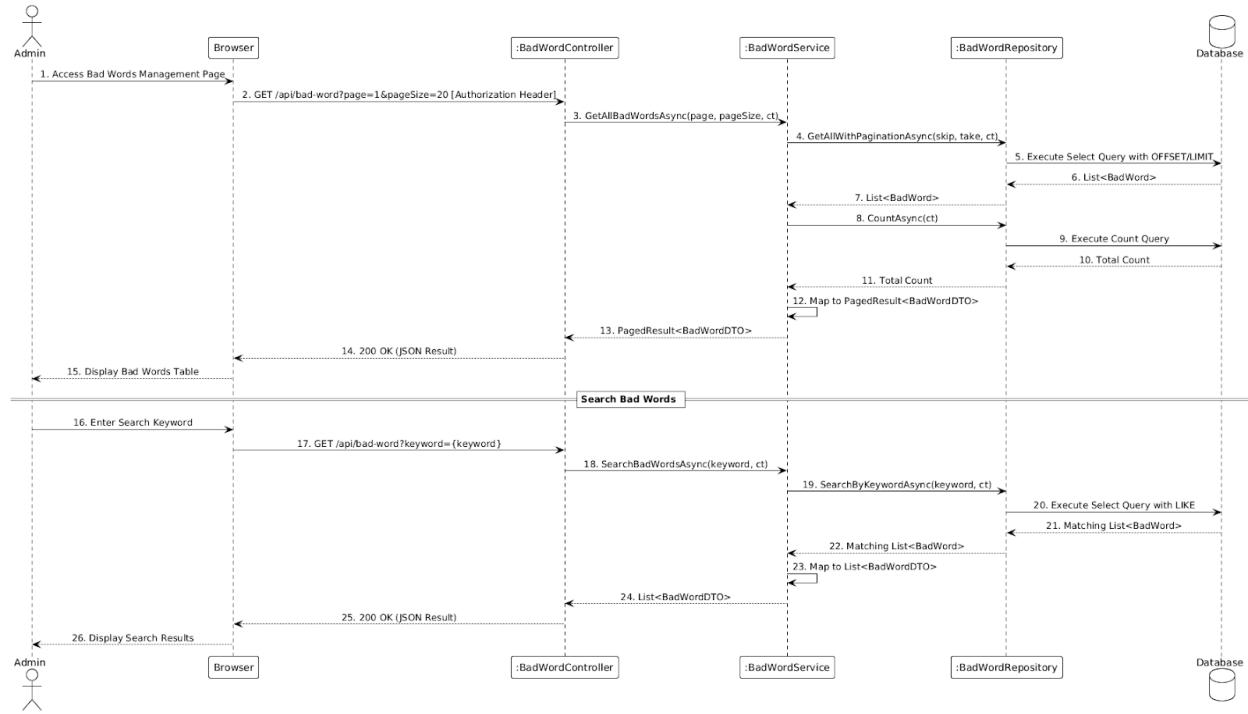
3.17 Bad Word Management

3.17.1 Class Diagram

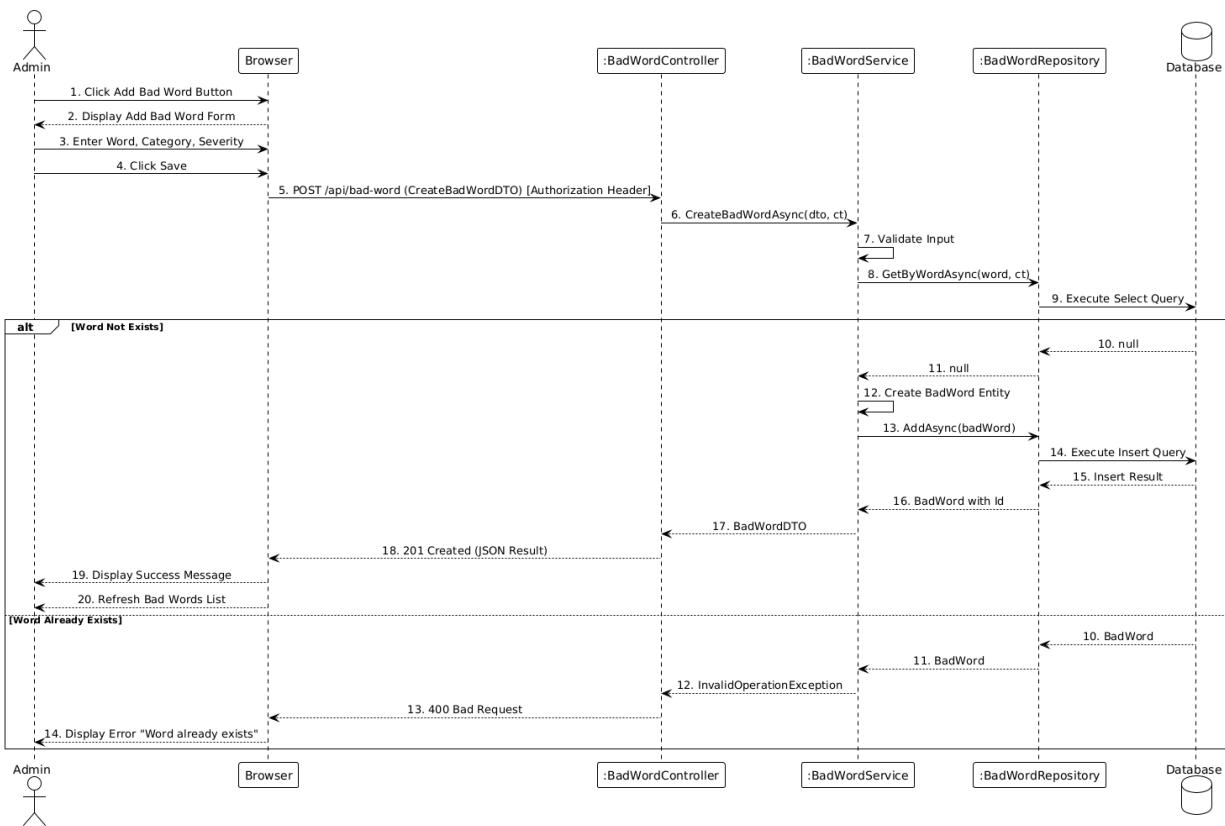


3.17.2 Sequence Diagram

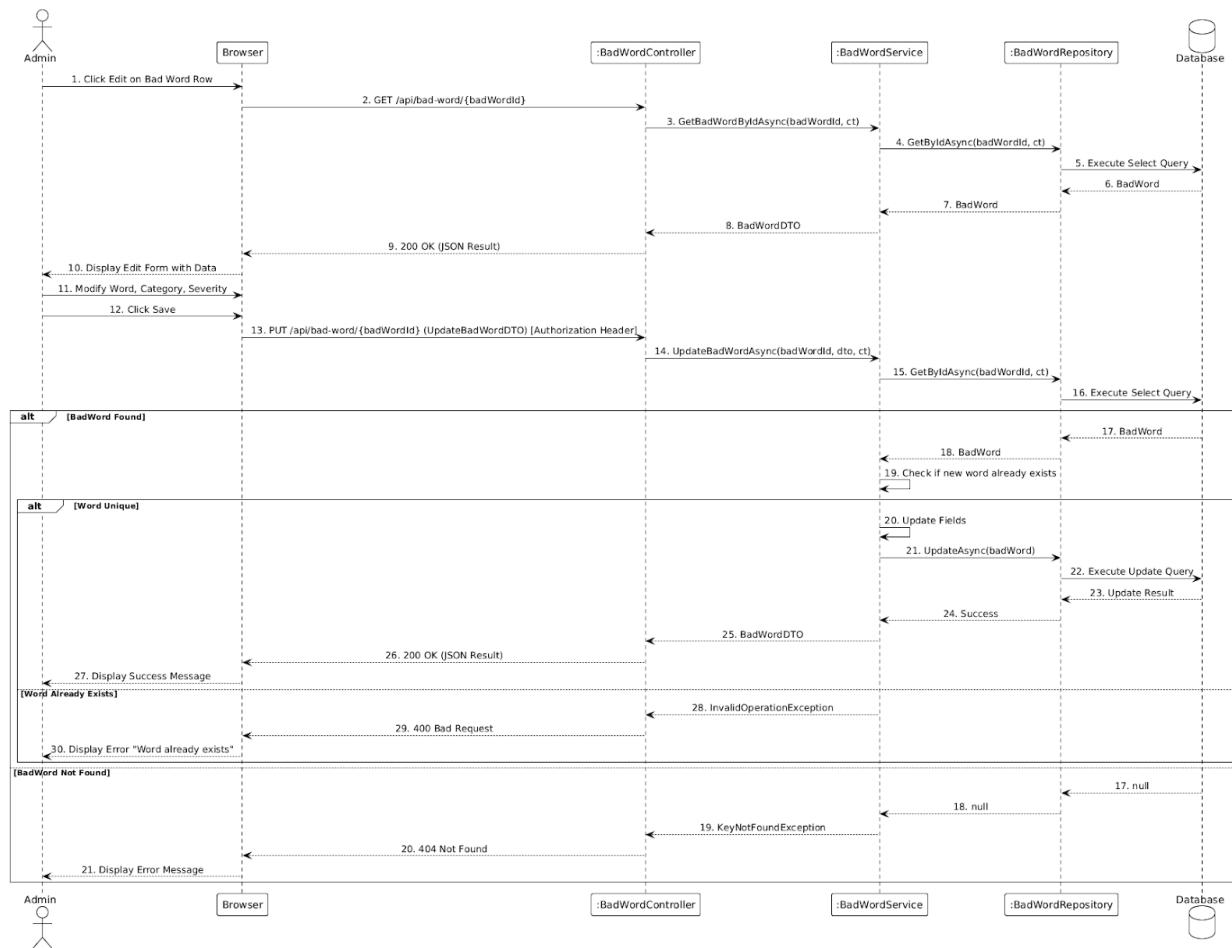
3.17.2.1 View Bad Word List



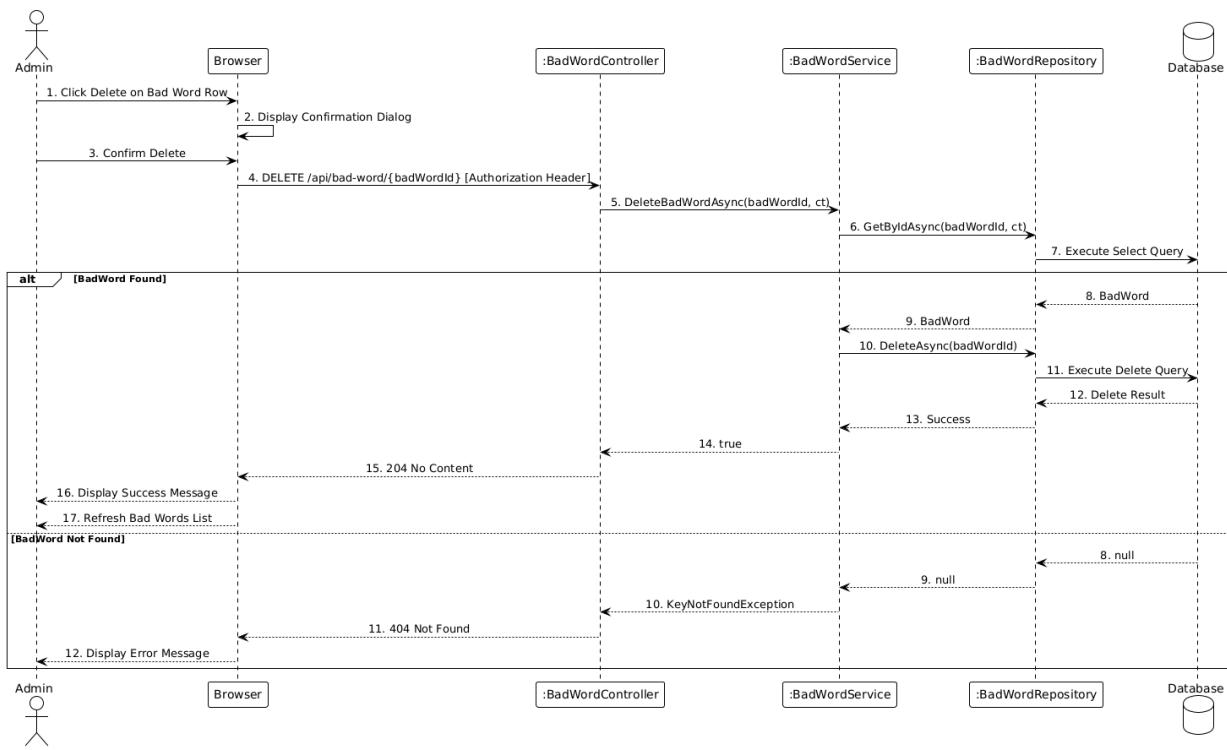
3.17.2.2 Create Bad Word



3.17.2.3 Edit Bad Word



3.17.2.4 Delete Bad Word



4. Class Specifications

4.1 Models

4.1.1 Address

No	Name	Description
Attributes		
01	AddressId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each address.</p>
02	Latitude	<p>Visibility: private</p> <p>Type: decimal?</p> <p>Purpose: Stores the geographic latitude coordinate of the address.</p>

03	Longitude	Visibility: private Type: decimal? Purpose: Stores the geographic longitude coordinate of the address.
04	FullAddress	Visibility: private Type: string Purpose: Stores the complete address string.
05	City	Visibility: private Type: string Purpose: Stores the city name.
06	District	Visibility: private Type: string? Purpose: Stores the district name.
07	Ward	Visibility: private Type: string? Purpose: Stores the ward name.
08	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the address was created.
09	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the address was last updated

4.1.2 Attributes

No	Name	Description
Attributes		
01	AttributeId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each attribute</p>
02	Name	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the attribute (e.g., Size, Color, Energy Level)</p>
03	TypeValue	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the type of value for this attribute (e.g., numeric, text, option)</p>
04	Unit	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the unit of measurement if applicable (e.g., kg, cm).</p>
05	Percent	<p>Visibility: private</p> <p>Type: decimal?</p> <p>Purpose: Stores the percentage weight for matching algorithm</p>

06	IsDeleted	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the attribute has been soft deleted</p>
07	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the attribute was created.</p>
08	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the attribute was last updated</p>
<i>Methods/Operations</i>		
09	Ward	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the ward name.</p>
09	GetAttributeOptions()	<p>Visibility: public</p> <p>Return: ICollection<AttributeOption></p> <p>Purpose: Returns the collection of predefined options for this attribute.</p> <p>Parameters: None</p>
10	GetPetCharacteristics()	<p>Visibility: public</p> <p>Return: ICollection<PetCharacteristic></p> <p>Purpose: Returns the collection of pet characteristics using this attribute.</p> <p>Parameters: None</p>

4.1.3 AttributeOption

No	Name	Description
Attributes		
01	OptionId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each attribute option</p>
02	AttributeId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the foreign key reference to the Attribute table.</p>
03	Name	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the option (e.g., Small, Medium, Large)</p>
04	IsDeleted	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the option has been soft deleted</p>
05	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the attribute was created.</p>
06	UpdatedAt	Visibility: private

		<p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the attribute was last updated</p>
Methods/Operations		
07	GetAttribute()	<p>Visibility: public</p> <p>Return: Attribute?</p> <p>Purpose: Returns the parent attribute for this option.</p>
08	GetPetCharacteristics()	<p>Visibility: public</p> <p>Return: ICollection<PetCharacteristic></p> <p>Purpose: Returns the collection of pet characteristics using this option.</p> <p>Parameters: None</p>
09	GetPetCharacteristics()	<p>Visibility: public</p> <p>Return: ICollection<PetCharacteristic></p> <p>Purpose: Returns the collection of pet characteristics using this attribute.</p> <p>Parameters: None</p>
10	GetUserPreferences()	<p>Visibility: public</p> <p>Return: ICollection<UserPreference></p> <p>Purpose: Returns the collection of user preferences using this option.</p> <p>Parameters: None</p>

4.1.4 Block

No	Name	Description
Attributes		
01	FromUserId	Visibility: private

		Type: int Purpose: Stores the user ID who initiated the block (composite primary key part 1).
02	ToUserId	Visibility: private Type: int Purpose: Stores the user ID who is blocked (composite primary key part 2)
03	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the attribute was created.
04	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the attribute was last updated
Methods/Operations		
05	GetFromUser()	Visibility: public Return: User Purpose: Returns the user who initiated the block. Parameters: None
06	GetToUser()	Visibility: public Return: User Purpose: Returns the user who is blocked. Parameters: None

4.1.5 ChaiAi

No	Name	Description
Attributes		
01	ChatAId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Serves as the primary key of the ChatAi entity, uniquely identifying each AI chat session.</p>
02	UserId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the ID of the user who owns the AI chat session (nullable).</p>
03	Title	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Represents the title of the AI chat session.</p>
04	IsDeleted	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the chat session is logically deleted.</p>
05	CreatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the AI chat session was created.</p>
06	UpdatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the AI chat session was last updated.</p>

07	ChatAicontents	<p>Visibility: public</p> <p>Type: ICollection<ChatAicontent></p> <p>Purpose: Contains the list of AI chat messages associated with this chat session.</p>
08	ExpertConfirmations	<p>Visibility: public</p> <p>Type: ICollection<ExpertConfirmation></p> <p>Purpose: Stores expert confirmation records related to this AI chat session.</p>
09	User	<p>Visibility: public</p> <p>Type: User?</p> <p>Purpose: Represents the user entity linked to this AI chat session.</p>
Methods/Operations		
07	GetChatAicontents()	<p>Visibility: public</p> <p>Return: ICollection<ChatAicontent></p> <p>Purpose: Returns the collection of messages in this AI chat session.</p> <p>Parameters: None.</p>
08	GetExpertConfirmations()	<p>Visibility: public</p> <p>Return: ICollection<ExpertConfirmation></p> <p>Purpose: Returns the collection of expert confirmations related to this chat.</p>
09	GetUser()	<p>Visibility: public</p> <p>Return: User?</p> <p>Purpose: Returns the user associated with this AI chat session.</p> <p>Parameters: None</p>

4.1.6 ChaiAiContent

No	Name	Description
Attributes		
01	ContentId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each AI chat content record.</p>
02	ChatAiid	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the AI chat session associated with this content.</p>
03	Question	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the question asked by the user in the AI chat session.</p>
04	Answer	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the answer generated by the AI for the corresponding question.</p>
05	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the chat content was created.</p>
06	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p>

		Purpose: Stores the timestamp when the chat content was last updated.
07	ChatAi	<p>Visibility: private</p> <p>Type: ChatAi?</p> <p>Purpose: Stores the AI chat session associated with this content.</p>

4.1.7 ChatExpert

No	Name	Description
Attributes		
01	ContentId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each expert chat message.</p>
02	ChatExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert chat session associated with this message.</p>
03	FromId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the sender (user or expert) of the message.</p>

04	Message	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the message content sent in the expert chat session.</p>
05	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert involved in this message (if applicable).</p>
06	UserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user involved in this message (if applicable).</p>
07	ChatAiid	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the AI chat session related to this expert chat content (if applicable).</p>
08	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the expert chat message was created.</p>

09	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the expert chat message was last updated.</p>
10	ChatExpert	<p>Visibility: private</p> <p>Type: ChatExpert?</p> <p>Purpose: Stores the expert chat session associated with this message.</p>
11	ExpertConfirmation	<p>Visibility: private</p> <p>Type: ExpertConfirmation?</p> <p>Purpose: Stores the expert confirmation related to this message (if applicable).</p>
12	From	<p>Visibility: private</p> <p>Type: User?</p> <p>Purpose: Stores the sender (user or expert) of this chat message.</p>

4.1.8 ChatExpertContent

No	Name	Description
Attributes		
01	ChatExpertId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each expert chat session.</p>
02	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert involved in the chat session.</p>
03	UserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user participating in the expert chat session.</p>
04	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the expert chat session was created.</p>
05	UpdatedAt	Visibility: private

No	Name	Description
Attributes		
01	ChatExpertId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each expert chat session.</p>
02	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert involved in the chat session.</p>
		<p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the expert chat session was last updated.</p>
06	ChatExpertContent s	<p>Visibility: private</p> <p>Type: ICollection<ChatExpertContent></p> <p>Purpose: Stores the collection of messages exchanged in this expert chat session.</p>
07	Expert	<p>Visibility: private</p> <p>Type: User?</p> <p>Purpose: Stores the expert user associated with this chat session.</p>

No	Name	Description
Attributes		
01	ChatExpertId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each expert chat session.</p>
02	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert involved in the chat session.</p>
08	User	<p>Visibility: private</p> <p>Type: User?</p> <p>Purpose: Stores the user who initiated the expert chat session.</p>

No	Name	Description
Attributes		
01	MatchId	<p>Visibility: private</p> <p>Type: int</p>

		Purpose: Stores the unique identifier for each match/chat session.
02	FromPetId	Visibility: private Type: int? Purpose: Stores the identifier of the pet who initiated the match.
03	ToPetId	Visibility: private Type: int? Purpose: Stores the identifier of the pet who received the match.
04	FromUserId	Visibility: private Type: int? Purpose: Stores the identifier of the user who initiated the match.
05	ToUserId	Visibility: private Type: int? Purpose: Stores the identifier of the user who received the match.
06	Status	Visibility: private

		Type: string? Purpose: Stores the status of the match.
07	IsDeleted	Visibility: private Type: bool? Purpose: Indicates whether the chat/match is soft deleted.
08	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the match was created.
09	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the match was last updated.
Methods/Operations		
10	GetChatUserContents()	Visibility: public Return: ICollection<ChatUserContent> Purpose: Returns the collection of messages in this chat session. Parameters: None.
11	GetFromPet()	Visibility: public Return: Pet? Purpose: Returns the pet who initiated the match. Parameters: None.

12	GetToPet()	<p>Visibility: public</p> <p>Return: Pet?</p> <p>Purpose: Returns the pet who received the match.</p> <p>Parameters: None.</p>
----	------------	--

4.1.10 ChatUserContent

No	Name	Description
Attributes		
01	ContentId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each chat content.</p>
02	MatchId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the chat/match session.</p>
03	FromUserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user who sent the message.</p>
04	FromPetId	<p>Visibility: private</p> <p>Type: int?</p>

		Purpose: Stores the identifier of the pet associated with the sender.
05	Message	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the content of the message.</p>
06	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the message was created.</p>
07	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the message was last updated.</p>
Methods/Operations		
08	GetFromUser()	<p>Visibility: public</p> <p>Return: User?</p> <p>Purpose: Returns the user who sent the message.</p> <p>Parameters: None.</p>
09	GetFromPet()	<p>Visibility: public</p> <p>Return: Pet?</p> <p>Purpose: Returns the pet associated with the sender.</p> <p>Parameters: None.</p>

10	GetMatch()	<p>Visibility: public</p> <p>Return: ChatUser?</p> <p>Purpose: Returns the chat/match session this message belongs to.</p> <p>Parameters: None.</p>
11	GetReports()	<p>Visibility: public</p> <p>Return: ICollection<Report></p> <p>Purpose: Returns the collection of reports made against this message.</p> <p>Parameters: None.</p>

4.1.11 DailyLimit

No	Name	Description
Attributes		
01	LimitId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each daily limit record.</p>
02	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the user.</p>
03	ActionType	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the type of action being limited.</p>

04	ActionDate	<p>Visibility: private</p> <p>Type: DateOnly</p> <p>Purpose: Stores the date when the action limit is tracked.</p>
05	Count	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the number of times the action has been performed on that day.</p>
06	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the limit record was created.</p>
Methods/Operations		
07	GetUser()	<p>Visibility: public</p> <p>Return: User</p> <p>Purpose: Returns the user associated with this daily limit.</p> <p>Required navigation property.</p> <p>Parameters: None.</p>

4.1.12 ExpertConfirmation

No	Name	Description
Attributes		
01	ExpertId	Visibility: private

		Type: int Purpose: Stores the identifier of the expert.
02	UserId	Visibility: private Type: int Purpose: Stores the identifier of the user requesting expert help.
03	ChatAiid	Visibility: private Type: int Purpose: Stores the identifier of the AI chat session.
04	Status	Visibility: private Type: string? Purpose: Stores the status of the confirmation.
05	Message	Visibility: private Type: string? Purpose: Stores the message or response from the expert.
06	CreatedAt	Visibility: private Type: DateTime?

		Purpose: Stores the timestamp when the confirmation was created.
07	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the confirmation was last updated.</p>
08	UserQuestion	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the question from the user to the expert.</p>
Methods/Operations		
09	GetChatAi()	<p>Visibility: public</p> <p>Return: ChatAi</p> <p>Purpose: Returns the AI chat session associated with this confirmation.</p> <p>Parameters: None.</p>
10	GetChatExpertContents()	<p>Visibility: public</p> <p>Return: ICollection<ChatExpertContent></p> <p>Purpose: Returns the collection of chat contents between user and expert.</p> <p>Parameters: None.</p>
11	GetExpert()	<p>Visibility: public</p> <p>Return: User</p> <p>Purpose: Returns the expert user. Required navigation property.</p>

		Parameters: None.
12	GetUser()	<p>Visibility: public</p> <p>Return: User</p> <p>Purpose: Returns the user who requested expert help.</p> <p>Required navigation property.</p> <p>Parameters: None.</p>

4.1.13 Notification

No	Name	Description
Attributes		
01	NotificationId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each notification.</p>
02	UserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user who receives the notification.</p>
03	Title	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the title of the notification.</p>
04	Message	Visibility: private

		Type: string? Purpose: Stores the content/message of the notification.
05	Type	Visibility: private Type: string? Purpose: Stores the type of notification.
06	IsRead	Visibility: private Type: bool Purpose: Indicates whether the notification has been read.
07	ReferenceId	Visibility: private Type: int? Purpose: Stores the reference ID for related entities.
08	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the notification was created.
09	UpdatedAt	Visibility: private Type: DateTime?

		Purpose: Stores the timestamp when the notification was last updated.
Methods/Operations		
10	GetUser()	<p>Visibility: public</p> <p>Return: User?</p> <p>Purpose: Returns the user who receives this notification.</p> <p>Parameters: None.</p>

4.1.14 PaymentHistory

No	Name	Description
Attributes		
01	HistoryId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each payment history record.</p>
02	UserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user who made the payment .</p>
03	StatusService	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the service status.</p>

04	StartDate	Visibility: private Type: DateOnly? Purpose: Stores the start date of the service subscription.
05	EndDate	Visibility: private Type: DateOnly? Purpose: Stores the end date of the service subscription.
06	Amount	Visibility: private Type: decimal? Purpose: Stores the payment amount.
07	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the payment record was created.
08	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the payment record was last updated.
Methods/Operations		
09	GetUser()	Visibility: public Return: User?

		Purpose: Returns the user who made this payment. Parameters: None.
--	--	---

4.1.15 Pet

No	Name	Description
Attributes		
01	PetId	Visibility: private Type: int Purpose: Stores the unique identifier for each pet.
02	UserId	Visibility: private Type: int? Purpose: Stores the identifier of the user who owns this pet.
03	Name	Visibility: private Type: string? Purpose: Stores the name of the pet.
04	Breed	Visibility: private Type: string? Purpose: Stores the breed of the pet.
05	Gender	Visibility: private

		Type: string? Purpose: Stores the gender of the pet.
06	Age	Visibility: private Type: int? Purpose: Stores the age of the pet.
07	IsActive	Visibility: private Type: bool? Purpose: Indicates whether the pet profile is active and visible for matching.
08	IsDeleted	Visibility: private Type: bool? Purpose: Indicates whether the pet is soft deleted.
09	Description	Visibility: private Type: string? Purpose: Stores a description or bio of the pet.
10	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the pet was created.

11	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the pet was last updated.</p>
Methods/Operations		
12	GetChatUserContents()	<p>Visibility: public</p> <p>Return: ICollection<ChatUserContent></p> <p>Purpose: Returns the collection of chat contents related to this pet.</p> <p>Parameters: None.</p>
13	GetChatUserFromPets()	<p>Visibility: public</p> <p>Return: ICollection<ChatUser></p> <p>Purpose: Returns the collection of chats where this pet initiated the conversation.</p> <p>Parameters: None.</p>
14	GetChatUserToPets()	<p>Visibility: public</p> <p>Return: ICollection<ChatUser></p> <p>Purpose: Returns the collection of chats where this pet received the conversation.</p> <p>Parameters: None.</p>
15	GetPetCharacteristics()	<p>Visibility: public</p> <p>Return: ICollection<PetCharacteristic></p> <p>Purpose: Returns the collection of characteristics of this pet.</p> <p>Parameters: None.</p>
16	GetPetPhotos()	<p>Visibility: public</p> <p>Return: ICollection<PetPhoto></p> <p>Purpose: Returns the collection of photos of this pet.</p>

		Parameters: None.
17	GetUser()	<p>Visibility: public</p> <p>Return: User?</p> <p>Purpose: Returns the user who owns this pet.</p> <p>Parameters: None.</p>

4.1.16 PetCharacteristic

No	Name	Description
Attributes		
01	PetId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the pet.</p>
02	Attributeld	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the attribute.</p>
03	OptionId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the selected option ID for this characteristic .</p>
04	Value	<p>Visibility: private</p> <p>Type: int?</p>

		Purpose: Stores the numeric value for this characteristic.
05	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the characteristic was created.</p>
06	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the characteristic was last updated.</p>
Methods/Operations		
07	GetAttribute()	<p>Visibility: public</p> <p>Return: Attribute</p> <p>Purpose: Returns the attribute associated with this characteristic.</p>
08	GetOption()	<p>Visibility: public</p> <p>Return: AttributeOption?</p> <p>Purpose: Returns the selected option for this characteristic.</p> <p>Parameters: None.</p>
09	GetPet()	<p>Visibility: public</p> <p>Return: Pet</p> <p>Purpose: Returns the pet that has this characteristic.</p> <p>Parameters: None.</p>

4.1.17 Pet Photo

No	Name	Description
Attributes		
01	Photoid	<p>Visibility: private Type: int Purpose: Stores the unique identifier for each pet photo.</p>
02	PetId	<p>Visibility: private Type: int Purpose: Stores the identifier of the pet this photo belongs to.</p>
03	ImageUrl	<p>Visibility: private Type: string Purpose: Stores the URL of the pet photo.</p>
04	PublicId	<p>Visibility: private Type: string? Purpose: Stores the public ID from cloud storage for image management.</p>
05	IsPrimary	<p>Visibility: private Type: bool</p>

		Purpose: Indicates whether this is the primary/main photo of the pet.
06	SortOrder	Visibility: private Type: int Purpose: Stores the display order of the photo.
07	IsDeleted	Visibility: private Type: bool Purpose: Indicates whether the photo is soft deleted.
08	CreatedAt	Visibility: private Type: DateTime Purpose: Stores the timestamp when the photo was uploaded.
09	UpdatedAt	Visibility: private Type: DateTime Purpose: Stores the timestamp when the photo was last updated.
Methods/Operations		
10	GetPet()	Visibility: public Return: Pet Purpose: Returns the pet that this photo belongs to. Parameters: None.

4.1.18 Report

No	Name	Description
Attributes		
01	ReportId	Visibility: private Type: int Purpose: Stores the unique identifier for each report.
02	UserReportId	Visibility: private Type: int? Purpose: Stores the identifier of the user who made the report.
03	ContentId	Visibility: private Type: int? Purpose: Stores the identifier of the reported content.
04	Reason	Visibility: private Type: string? Purpose: Stores the reason for the report.
05	Status	Visibility: private Type: string? Purpose: Stores the current status of the report.

06	Resolution	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the resolution details after the report is reviewed.</p>
07	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the report was created.</p>
08	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the report was last updated.</p>
Methods/Operations		
09	GetContent()	<p>Visibility: public</p> <p>Return: ChatUserContent?</p> <p>Purpose: Returns the reported content.</p> <p>Parameters: None.</p>
10	GetUserReport()	<p>Visibility: public</p> <p>Return: User?</p> <p>Purpose: Returns the user who made the report.</p> <p>Parameters: None.</p>

4.1.19 Role

No	Name	Description
Attributes		

01	RoleId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each role.</p>
02	RoleName	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the role.</p>
03	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the role was created.</p>
04	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the role was last updated.</p>
Methods/Operations		
05	GetUsers()	<p>Visibility: public</p> <p>Return: ICollection<User></p> <p>Purpose: Returns the collection of users assigned to this role.</p> <p>Parameters: None.</p>

4.1.20 User

No	Name	Description
Attributes		

01	UserId	Visibility: private Type: int Purpose: Stores the unique identifier for each user.
02	RoleId	Visibility: private Type: int? Purpose: Stores the role identifier of the user.
03	UserStatusId	Visibility: private Type: int? Purpose: Stores the status identifier of the user.
04	AddressId	Visibility: private Type: int? Purpose: Stores the address identifier of the user.
05	FullName	Visibility: private Type: string? Purpose: Stores the full name of the user.
06	Gender	Visibility: private Type: string? Purpose: Stores the gender of the user.

07	Email	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the email address of the user.</p>
08	PasswordHash	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the hashed password of the user.</p>
09	ProviderLogin	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the login provider.</p>
10	TokenJwt	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the JWT token for authentication.</p>
11	IsDeleted	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the user account is soft deleted.</p>
12	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the user was created.</p>

13	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the user was last updated.</p>
14	IsProfileComplete	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the user has completed their profile setup.</p>
Methods/Operations		
15	GetAddress()	<p>Visibility: public</p> <p>Return: Address?</p> <p>Purpose: Returns the address of the user.</p> <p>Parameters: None.</p>
16	GetBlockFromUsers()	<p>Visibility: public</p> <p>Return: ICollection<Block></p> <p>Purpose: Returns the collection of blocks initiated by this user.</p> <p>Parameters: None.</p>
17	GetBlockToUsers()	<p>Visibility: public</p> <p>Return: ICollection<Block></p> <p>Purpose: Returns the collection of blocks received by this user.</p> <p>Parameters: None.</p>
18	GetChatAis()	<p>Visibility: public</p> <p>Return: ICollection<ChatAi></p> <p>Purpose: Returns the collection of AI chat sessions of this user.</p> <p>Parameters: None.</p>

19	GetChatUserContents()	<p>Visibility: public</p> <p>Return: ICollection<ChatUserContent></p> <p>Purpose: Returns the collection of chat contents sent by this user.</p> <p>Parameters: None.</p>
20	GetChatExpertContents()	<p>Visibility: public</p> <p>Return: ICollection<ChatExpertContent></p> <p>Purpose: Returns the collection of expert chat contents of this user.</p> <p>Parameters: None.</p>
21	GetChatExpertExperts()	<p>Visibility: public</p> <p>Return: ICollection<ChatExpert></p> <p>Purpose: Returns the collection of expert chats where the user is the expert.</p> <p>Parameters: None.</p>
22	GetChatExpertUsers()	<p>Visibility: public</p> <p>Return: ICollection<ChatExpert></p> <p>Purpose: Returns the collection of expert chats where the user is the client.</p> <p>Parameters: None.</p>
23	GetDailyLimits()	<p>Visibility: public</p> <p>Return: ICollection<DailyLimit></p> <p>Purpose: Returns the collection of daily limits for this user.</p> <p>Parameters: None.</p>
24	GetExpertConfirmationExperts()	<p>Visibility: public</p> <p>Return: ICollection<ExpertConfirmation></p> <p>Purpose: Returns the collection of confirmations where the user is the expert.</p>

		Parameters: None.
25	GetExpertConfirmationUsers()	Visibility: public Return: ICollection<ExpertConfirmation> Purpose: Returns the collection of confirmations where the user requested an expert. Parameters: None.
26	GetNotifications()	Visibility: public Return: ICollection<Notification> Purpose: Returns the collection of notifications for this user. Parameters: None.
27	GetPaymentHistories()	Visibility: public Return: ICollection<PaymentHistory> Purpose: Returns the collection of payment histories of this user. Parameters: None.
28	GetPets()	Visibility: public Return: ICollection<Pet> Purpose: Returns the collection of pets owned by this user. Parameters: None.
29	GetReports()	Visibility: public Return: ICollection<Report> Purpose: Returns the collection of reports made by this user. Parameters: None.
30	GetRole()	Visibility: public Return: Role? Purpose: Returns the role assigned to this user Parameters: None.

31	GetUserBanHistories()	<p>Visibility: public</p> <p>Return: ICollection<UserBanHistory></p> <p>Purpose: Returns the collection of ban histories for this user.</p> <p>Parameters: None.</p>
32	GetUserPreferences()	<p>Visibility: public</p> <p>Return: ICollection<UserPreference></p> <p>Purpose: Returns the collection of preferences set by this user.</p> <p>Parameters: None.</p>
33	GetUserStatus()	<p>Visibility: public</p> <p>Return: UserStatus?</p> <p>Purpose: Returns the status of this user.</p> <p>Parameters: None.</p>

4.1.21 UserBanHistory

No	Name	Description
Attributes		
01	BanId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for each ban record (primary key).</p>
02	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the user who is banned.</p>
03	BanStart	Visibility: private

		Type: DateTime Purpose: Stores the timestamp when the ban starts. Required field.
04	BanEnd	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the ban ends.
05	BanReason	Visibility: private Type: string? Purpose: Stores the reason for the ban.
06	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the ban record was created.
07	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the ban record was last updated.
08	IsActive	Visibility: private Type: bool?

		Purpose: Indicates whether the ban is currently active.
Methods/Operations		
09	GetUser()	<p>Visibility: public</p> <p>Return: User</p> <p>Purpose: Returns the user who is banned. Required navigation property.</p> <p>Parameters: None.</p>

4.1.22 UserPreference

No	Name	Description
Attributes		
01	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the user ID who owns this preference.</p>
02	Attributeld	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the attribute ID for this preference.</p>
03	OptionId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the selected option ID for this preference.</p>
04	MaxValue	Visibility: private

		Type: int? Purpose: Stores the maximum value for range-based preferences.
05	MinValue	Visibility: private Type: int? Purpose: Stores the minimum value for range-based preferences.
06	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the preference was created.
07	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the preference was last updated.
Methods/Operations		
08	GetAttribute()	Visibility: public Return: Attribute Purpose: Returns the attribute associated with this preference. Required navigation property. Parameters: None.
09	GetOption()	Visibility: public

		<p>Return: AttributeOption?</p> <p>Purpose: Returns the selected attribute option for this preference (optional).</p> <p>Parameters: None.</p>
10	GetUser()	<p>Visibility: public</p> <p>Return: User</p> <p>Purpose: Returns the user who owns this preference. Required navigation property.</p> <p>Parameters: None.</p>

4.1.23 UserStatus

No	Name	Description
Attributes		
01	UserStatusId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier for the user status</p>
02	UserStatusName	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the user status</p>
03	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p>

		Purpose: Stores the timestamp when the user status was created.
04	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the user status was last updated.</p>
Methods/Operations		
05	GetUsers()	<p>Visibility: public</p> <p>Return: ICollection<User></p> <p>Purpose: Returns the collection of users that have this status assigned. Represents a One-to-Many relationship.</p> <p>Parameters: None.</p>

4.1.24 BadWord

No	Name	Description
Attributes		
01	BadWordId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the bad word (primary key).</p>
02	Word	<p>Visibility: private</p> <p>Type: string</p>

		Purpose: Stores the prohibited word or phrase used for content moderation.
03	IsRegex	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the word is treated as a regular expression pattern.</p>
04	Level	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Defines the severity level of the bad word for moderation and filtering logic.</p>
05	Category	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the category of the bad word (e.g., offensive, sexual, violent).</p>
06	IsActive	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the bad word is currently active and applied in the filtering system.</p>
07	CreatedAt	Visibility: private

		Type: DateTime? Purpose: Stores the timestamp when the bad word record was created.
08	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the bad word record was last updated.

4.1.25 PetAppointment

No	Name	Description
Attributes		
01	AppointmentId	Visibility: private Type: int Purpose: Stores the unique identifier of the pet appointment (primary key).
02	MatchId	Visibility: private Type: int Purpose: References the match (ChatUser) between two pets that enables the appointment.
03	InviterPetId	Visibility: private Type: int

		Purpose: Stores the ID of the pet that initiates the appointment.
04	InviteePetId	Visibility: private Type: int Purpose: Stores the ID of the pet that receives the appointment invitation.
05	InviterUserId	Visibility: private Type: int Purpose: Stores the user ID of the appointment initiator.
06	InviteeUserId	Visibility: private Type: int Purpose: Stores the user ID of the invited user.
07	AppointmentDateTime	Visibility: private Type: DateTime Purpose: Stores the scheduled date and time of the pet appointment.
08	LocationId	Visibility: private Type: int?

		Purpose: References the location where the appointment takes place.
09	ActivityType	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Defines the type of activity (e.g., walk, cafe, playdate).</p>
10	Status	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the current status of the appointment (pending, confirmed, rejected, cancelled, on_going, completed, no_show).</p>
11	CurrentDecisionUserld	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Identifies the user who currently has permission to accept, decline, or counter-offer the appointment.</p>
12	CounterOfferCount	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the number of counter-offers made for the appointment (maximum limit applied).</p>
13	InviterCheckedIn	Visibility: private

		Type: bool? Purpose: Indicates whether the inviter has checked in to the appointment.
14	InviteeCheckedIn	Visibility: private Type: bool? Purpose: Indicates whether the invitee has checked in to the appointment.
15	InviterCheckInTime	Visibility: private Type: DateTime? Purpose: Stores the check-in time of the inviter.
16	InviteeCheckInTime	Visibility: private Type: DateTime? Purpose: Stores the check-in time of the invitee.
17	CancelledBy	Visibility: private Type: int? Purpose: Stores the user ID of the person who cancelled the appointment.
18	CancelReason	Visibility: private Type: string?

		Purpose: Stores the reason for appointment cancellation.
19	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the appointment was created.</p>
20	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the appointment was last updated.</p>

4.1.26 PetAppointmentLocation

No	Name	Description
Attributes		
01	LocationId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the appointment location (primary key).</p>
02	Name	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the pet-friendly location.</p>

03	Address	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the full address of the location.</p>
04	Latitude	<p>Visibility: private</p> <p>Type: decimal</p> <p>Purpose: Stores the latitude coordinate of the location for map positioning.</p>
05	Longitude	<p>Visibility: private</p> <p>Type: decimal</p> <p>Purpose: Stores the longitude coordinate of the location for map positioning.</p>
06	City	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the city/province where the location is located.</p>
07	District	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the district/area where the location is located.</p>
08	IsPetFriendly	Visibility: private

		Type: bool? Purpose: Indicates whether the location is verified or marked as pet-friendly.
09	PlaceType	Visibility: private Type: string? Purpose: Stores the type of location (e.g., park, pet_cafe, vet_clinic, custom).
10	GooglePlaceId	Visibility: private Type: string? Purpose: Stores the Google Place ID for integrating with Google Maps/Places API.
11	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the location record was created.
12	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the location record was last updated.

4.1.27 PetEvent

No	Name	Description
Attributes		
01	EventId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the pet event (primary key).</p>
02	Title	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the title of the pet photo/video contest event.</p>
03	Description	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the detailed description of the event.</p>
04	CoverImageUrl	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the URL of the event cover image.</p>
05	StartTime	<p>Visibility: private</p> <p>Type: DateTime</p> <p>Purpose: Stores the start time of the event.</p>

06	SubmissionDeadline	<p>Visibility: private</p> <p>Type: DateTime</p> <p>Purpose: Stores the deadline for submitting contest entries.</p>
07	EndTime	<p>Visibility: private</p> <p>Type: DateTime</p> <p>Purpose: Stores the end time of the event, including voting closure and result calculation.</p>
08	Status	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the current status of the event (upcoming, active, submission_closed, voting_ended, completed, cancelled).</p>
09	PrizeDescription	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the description of prizes awarded to winners.</p>
10	PrizePoints	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the reward points given to winners.</p>
11	CreatedBy	<p>Visibility: private</p>

		Type: int Purpose: Stores the user ID of the event creator (admin or authorized user).
12	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the event was created.
13	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the event was last updated.

4.1.28 EventSubmission

No	Name	Description
Attributes		
01	SubmissionId	Visibility: private Type: int Purpose: Stores the unique identifier of the event submission (primary key).
02	EventId	Visibility: private Type: int

		Purpose: References the event (PetEvent) that this submission belongs to.
03	UserId	Visibility: private Type: int Purpose: Stores the ID of the user who submitted the entry.
04	PetId	Visibility: private Type: int Purpose: Stores the ID of the pet featured in the submission.
05	MediaUrl	Visibility: private Type: string Purpose: Stores the URL of the submitted media (image/video).
06	MediaType	Visibility: private Type: string Purpose: Stores the media type of the submission (image, video).
07	ThumbnailUrl	Visibility: private Type: string?

		Purpose: Stores the thumbnail URL for previewing video/image submissions.
08	Caption	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the caption or description provided by the user for the submission.</p>
09	VoteCount	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the total number of votes for the submission (denormalized for faster queries and ranking).</p>
10	Rank	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the final ranking position after results are calculated (1, 2, 3...).</p>
11	IsWinner	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the submission is a winning entry.</p>
12	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p>

		Purpose: Stores the timestamp when the submission was created.
13	IsDeleted	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the submission is soft-deleted (used to hide content without removing data permanently).</p>

4.1.29 EventVote

No	Name	Description
Attributes		
01	Voteld	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the vote record (primary key).</p>
02	SubmissionId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: References the event submission that is being voted for.</p>
03	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the ID of the user who casts the vote.</p>

04	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the vote was created.</p>
----	-----------	---

4.1.30 Policy

No	Name	Description
Attributes		
01	PolicyId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the policy (primary key).</p>
02	PolicyCode	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the unique policy code (e.g., TERMS_OF_SERVICE, PRIVACY_POLICY).</p>
03	PolicyName	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the display name of the policy.</p>
04	Description	<p>Visibility: private</p> <p>Type: string?</p>

		Purpose: Stores a short description of the policy.
05	DisplayOrder	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Defines the display order of the policy in the user interface.</p>
06	RequireConsent	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether user consent is required for this policy.</p>
07	IsActive	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the policy is currently active.</p>
08	IsDeleted	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the policy is soft-deleted.</p>
09	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the policy was created.</p>

10	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the policy was last updated.</p>
----	-----------	--

4.1.31 PolicyVersion

No	Name	Description
Attributes		
01	PolicyVersionId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the policy version (primary key).</p>
02	PolicyId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: References the policy that this version belongs to (foreign key to Policy).</p>
03	VersionNumber	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the version number of the policy (1, 2, 3...).</p>
04	Title	Visibility: private

		Type: string Purpose: Stores the title of this policy version.
05	Content	Visibility: private Type: string Purpose: Stores the full policy content (HTML or Markdown).
06	ChangeLog	Visibility: private Type: string? Purpose: Stores a summary of changes compared to the previous version.
07	Status	Visibility: private Type: string Purpose: Stores the publication status of the version (DRAFT, ACTIVE, INACTIVE).
08	PublishedAt	Visibility: private Type: DateTime? Purpose: Stores the publish timestamp when the version becomes ACTIVE.
09	DeactivatedAt	Visibility: private Type: DateTime?

		Purpose: Stores the timestamp when the version becomes INACTIVE (no longer effective).
10	CreatedByUserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the ID of the admin user who created this policy version.</p>
11	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the version record was created.</p>
12	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the version record was last updated.</p>

4.1.32 UserPolicyAccept

No	Name	Description
Attributes		
01	AcceptId	<p>Visibility: private</p> <p>Type: long</p>

		Purpose: Stores the unique identifier of the user policy acceptance record (primary key).
02	UserId	Visibility: private Type: int Purpose: Stores the ID of the user who accepted the policy.
03	PolicyVersionId	Visibility: private Type: int Purpose: References the specific policy version that the user accepted.
04	AcceptedAt	Visibility: private Type: DateTime Purpose: Stores the timestamp when the user accepted the policy version.
05	IsValid	Visibility: private Type: bool Purpose: Indicates whether this acceptance record is currently valid.
06	InvalidateAt	Visibility: private Type: DateTime?

		Purpose: Stores the timestamp when this acceptance record became invalid due to a newer policy version.
07	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the timestamp when the acceptance record was created.

4.2 DTO

4.2.1 AddressDTO

4.2.1.1 LocationDto

No	Name	Description
Attributes		
01	Latitude	Visibility: private Type: decimal Purpose: Stores the latitude coordinate of the location.
02	Longitude	Visibility: private Type: decimal Purpose: Stores the longitude coordinate of the location.

4.2.1.1 ManualAddressDto

No	Name	Description
Attributes		
01	City	Visibility: private Type: string? Purpose: Stores the city name of the manual address input.

02	District	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the district name of the manual address input.</p>
03	Ward	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the ward name of the manual address input.</p>

4.2.1.1 OpenStreetMapResponse

No	Name	Description
Attributes		
01	display_name	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the full formatted address returned by OpenStreetMap.</p>
02	address	<p>Visibility: private</p> <p>Type: AddressComponents?</p> <p>Purpose: Stores the detailed address components returned from OpenStreetMap.</p>

4.2.1.1 AddressComponents

No	Name	Description
Attributes		
01	city	Visibility: private

		Type: string? Purpose: Stores the city name from the address data.
02	town	Visibility: private Type: string? Purpose: Stores the town name from the address data.
03	province	Visibility: private Type: string? Purpose: Stores the province name from the address data.
04	state	Visibility: private Type: string? Purpose: Stores the state name from the address data.
05	region	Visibility: private Type: string? Purpose: Stores the region name from the address data.
06	country	Visibility: private Type: string? Purpose: Stores the country name from the address data.
07	county	Visibility: private Type: string? Purpose: Stores the county name from the address data.
08	suburb	Visibility: private

		Type: string? Purpose: Stores the suburb name from the address data.
09	quarter	Visibility: private Type: string? Purpose: Stores the quarter name from the address data.
10	neighbourhood	Visibility: private Type: string? Purpose: Stores the neighbourhood name from the address data.
11	village	Visibility: private Type: string? Purpose: Stores the village name from the address data.

4.2.2 AttributeDto

4.2.2.1 AttributeResponse

No	Name	Description
Attributes		
01	Attributeld	Visibility: private Type: int Purpose: Stores the unique identifier of the attribute.
02	Name	Visibility: private Type: string Purpose: Stores the name of the attribute.

03	TypeValue	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the data type or value type of the attribute.</p>
04	IsDeleted	<p>Visibility: private</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the attribute is soft-deleted.</p>
05	Unit	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the measurement unit of the attribute.</p>
06	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the creation timestamp of the attribute.</p>
07	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the last update timestamp of the attribute.</p>
08	optionResponses	<p>Visibility: private</p> <p>Type: List<OptionResponse></p> <p>Purpose: Stores the list of option responses related to the attribute.</p>

4.2.2.2 AttributeCreateRequest

No	Name	Description
Attributes		

01	Name	Visibility: private Type: string Purpose: Stores the name of the attribute to be created.
02	TypeValue	Visibility: private Type: string? Purpose: Stores the value type of the new attribute.
03	Unit	Visibility: private Type: string? Purpose: Stores the measurement unit of the new attribute.
04	IsDeleted	Visibility: private Type: bool? Purpose: Indicates whether the attribute is marked as soft-deleted upon creation.

4.2.2.3 AttributeUpdateRequest

No	Name	Description
Attributes		
01	Name	Visibility: private Type: string Purpose: Stores the updated name of the attribute.
02	TypeValue	Visibility: private Type: string? Purpose: Stores the updated value type of the attribute.

03	Unit	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the updated measurement unit of the attribute.</p>
----	------	---

4.2.2.4 OptionResponse

No	Name	Description
Attributes		
01	OptionId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the option.</p>
02	AttributeId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the attribute associated with this option.</p>
03	Name	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the name of the option.</p>

4.2.3 BanUserRequest

4.2.3.1 AttributeResponse

No	Name	Description
Attributes		
01	Reason	Visibility: private

		Type: string? Purpose: Stores the reason for banning the user.
02	DurationDays	Visibility: private Type: int Purpose: Stores the number of days the user is banned. A default value of 1 is applied if not provided.
03	IsPermanent	Visibility: private Type: bool Purpose: Indicates whether the ban is permanent. If true, the ban end date is undefined.

4.2.3.2 AttributeResponse

No	Name	Description
Attributes		
01	Reason	Visibility: private Type: string? Purpose: Stores the reason for unbanning the user.

4.2.4 ChatExpertContentDTO

4.2.4.1 SendMessageRequestChatExpert

No	Name	Description
Attributes		
01	Message	Visibility: private Type: string

		Purpose: Stores the message content to be sent in the expert chat.
02	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the expert receiving or sending the message.</p>
03	UserId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the user receiving or sending the message.</p>

4.2.5 ExpertConfirmationDTO

4.2.5.1 SendMessageRequestChatExpert

No	Name	Description
Attributes		
01	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the user requesting expert confirmation.</p>
02	ChatAild	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the associated AI chat session.</p>
03	ExpertId	<p>Visibility: private</p> <p>Type: int</p>

		Purpose: Stores the identifier of the assigned expert.
04	Status	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the current confirmation status (e.g., pending, approved, rejected).</p>
05	Message	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the expert's message or feedback.</p>
06	UserQuestion	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the original question submitted by the user.</p>
07	CreatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the creation timestamp of the expert confirmation request.</p>
08	UpdatedAt	<p>Visibility: private</p> <p>Type: DateTime?</p> <p>Purpose: Stores the last update timestamp of the expert confirmation request.</p>

4.2.5.2 ExpertConfirmationCreateDTO

No	Name	Description
Attributes		

01	Message	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the initial message sent when creating an expert confirmation request.</p>
02	ExpertId	<p>Visibility: private</p> <p>Type: int?</p> <p>Purpose: Stores the expert identifier if specified; otherwise auto-assigned by the system.</p>
03	UserQuestion	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the user's question requiring expert confirmation.</p>

4.2.5.3 ExpertConfirmationResponseDTO

No	Name	Description
Attributes		
01	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the user associated with the confirmation.</p>
02	ChatAId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the AI chat session.</p>
03	ExpertId	<p>Visibility: private</p>

		Type: int Purpose: Stores the identifier of the expert handling the confirmation.
04	Status	Visibility: private Type: string? Purpose: Stores the current confirmation status.
05	Message	Visibility: private Type: string? Purpose: Stores the expert's feedback message.
06	UserQuestion	Visibility: private Type: string? Purpose: Stores the original question from the user.
07	ResultMessage	Visibility: private Type: string Purpose: Stores the result message returned to the client after processing.

4.2.5.4 ExpertConfirmationUpdateDto

No	Name	Description
Attributes		
01	Status	Visibility: private Type: string?

		Purpose: Stores the updated confirmation status.
--	--	--

4.2.5.5 ReassignExpertConfirmationRequest

No	Name	Description
Attributes		
01	UserId	Visibility: private Type: int Purpose: Stores the identifier of the user associated with the confirmation request.
02	ChatAId	Visibility: private Type: int Purpose: Stores the identifier of the AI chat session.
03	FromExpertId	Visibility: private Type: int? Purpose: Stores the identifier of the originally assigned expert.
04	ToExpertId	Visibility: private Type: int Purpose: Stores the identifier of the new expert to be assigned.
05	Message	Visibility: private Type: string? Purpose: Stores the message explaining the reassignment reason.

06	KeepStatus	<p>Visibility: private</p> <p>Type: bool</p> <p>Purpose: Indicates whether the confirmation status should be preserved or reset to pending.</p>
----	------------	---

4.2.5.6 ReassignExpertConfirmationResponse

No	Name	Description
Attributes		
01	UserId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the user.</p>
02	ChatAidl	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the AI chat session.</p>
03	ExpertId	<p>Visibility: private</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the newly assigned expert.</p>
04	Status	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the current confirmation status after reassignment.</p>
05	Message	Visibility: private

		Type: string? Purpose: Stores the expert's message after reassignment.
06	UserQuestion	Visibility: private Type: string? Purpose: Stores the user's original question.
07	CreatedAt	Visibility: private Type: DateTime? Purpose: Stores the creation timestamp.
08	UpdatedAt	Visibility: private Type: DateTime? Purpose: Stores the last update timestamp.
09	ResultMessage	Visibility: private Type: string Purpose: Stores the result message after reassignment confirmation.

4.2.6 LoginRequest

4.2.6.1 LoginRequest

No	Name	Description
Attributes		
01	Email	Visibility: private Type: string

		Purpose: Stores the email address used for user authentication. Must be a valid email format and cannot be empty.
02	Password	<p>Visibility: private</p> <p>Type: string</p> <p>Purpose: Stores the password used for authentication. Must contain at least 6 characters.</p>
03	Platform	<p>Visibility: private</p> <p>Type: string?</p> <p>Purpose: Stores the platform type requesting login (e.g., "user" for mobile, "admin" for web). Defaults to "user" if not provided.</p>

4.2.7 NotificationDto

4.2.7.1 NotificationDto

No	Name	Description
Attributes		
01	NotificationId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the ID of the notification.</p>
02	Title	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the title of the notification.</p>
03	Message	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the content/message of the notification.</p>

04	CreatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the notification was created.
05	UserId	Visibility: public Type: int? Purpose: Stores the identifier of the user receiving the notification.
06	UserName	Visibility: public Type: string? Purpose: Stores the username linked to the notification.

4.2.7.2 NotificationDto_1

No	Name	Description
Attributes		
01	Title	Visibility: public Type: string? Purpose: Stores the notification title.
02	Message	Visibility: public Type: string? Purpose: Stores the message content.
03	UserId	Visibility: public Type: int?

		Purpose: Stores the ID of the user receiving the notification.
04	Type	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the notification type or category.</p>

4.2.8 OptionDto

4.2.8.1 OptionResponse

No	Name	Description
Attributes		
01	OptionId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the option.</p>
02	AttributId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the related attribute, if any.</p>
03	Name	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the option name. Cannot be null.</p>
04	IsDeleted	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Indicates whether this option is marked as deleted.</p>

4.2.9 PetCharacteristicDTO

4.2.8.1 PetCharacteristicDTO

No	Name	Description
Attributes		
01	OptionId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the identifier of the selected option for the pet characteristic.</p>
02	Value	<p>Visibility: public</p> <p>Type: double?</p> <p>Purpose: Stores the numeric value representing the pet characteristic.</p>

4.2.10 PetDTO

4.2.10.1 PetDto

No	Name	Description
Attributes		
01	PetId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the pet.</p>
02	Name	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the name of the pet.</p>
03	Breed	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the breed of the pet.</p>

04	Gender	Visibility: public Type: string? Purpose: Stores the gender of the pet.
05	Age	Visibility: public Type: int? Purpose: Stores the age of the pet.
06	IsActive	Visibility: public Type: bool? Purpose: Indicates whether the pet is active.
07	Description	Visibility: public Type: string? Purpose: Stores additional details about the pet.
08	UrlImageAvatar	Visibility: public Type: string? Purpose: Stores the URL of the pet's avatar image.

4.2.10.2 PetDto_1

No	Name	Description
Attributes		
01	PetId	Visibility: public Type: int Purpose: Stores the unique identifier of the pet.
02	Name	Visibility: publicT

		<p>Type: string?</p> <p>Purpose: Stores the name of the pet.</p>
03	Breed	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the breed of the pet.</p>
04	Gender	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the gender of the pet.</p>
05	Age	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the age of the pet.</p>
06	IsActive	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the pet is active.</p>
07	Description	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores additional information about the pet.</p>

4.2.10.3 PetDto_2

No	Name	Description
Attributes		
01	UserId	Visibility: public Type: int Purpose: Stores the identifier of the user who owns the pet.
02	Name	Visibility: public Type: string? Purpose: Stores the pet's name.
03	Breed	Visibility: public Type: string? Purpose: Stores the breed of the pet.
04	Gender	Visibility: public Type: string? Purpose: Stores the gender of the pet.

05	Age	Visibility: public Type: int? Purpose: Stores the age of the pet.
06	IsActive	Visibility: public Type: bool? Purpose: Indicates whether the pet is active.
07	Description	Visibility: public Type: string? Purpose: Stores the description or notes about the pet.

4.2.11 PetImageAnalysisDTO

4.2.11.1 PetImageAnalysisRequest

No	Name	Description
Attributes		
01	Image	Visibility: public Type: IFormFile Purpose: Stores the uploaded image used for pet characteristic analysis.

4.2.11.2 PetImageAnalysisResponse

No	Name	Description
Attributes		
01	Success	Visibility: public Type: bool Purpose: Indicates whether the analysis was successful.

02	Message	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the message describing the result of the analysis.</p>
03	Attributes	<p>Visibility: public</p> <p>Type: List<AttributeAnalysisResult> ?</p> <p>Purpose: Stores analyzed attribute results extracted from the image.</p>
04	SqlInsertScript	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores an optional generated SQL INSERT script for inserting attribute data.</p>

4.2.11.3 AttributeAnalysisResult

No	Name	Description
Attributes		
01	AttributeName	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the name of the detected attribute.</p>
02	OptionName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the detected option name of the attribute.</p>
03	Value	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the detected numeric value for the attribute.</p>

04	AttributeId	Visibility: public Type: int? Purpose: Stores the ID of the attribute (database reference).
05	OptionId	Visibility: public Type: int? Purpose: Stores the ID of the option (database reference).

4.2.11.4 PetCharacteristicInsertRequest

No	Name	Description
Attributes		
01	PetId	Visibility: public Type: int Purpose: Stores the ID of the pet to insert characteristics for.
02	Attributes	Visibility: public Type: List<AttributeAnalysisResult> Purpose: Stores a list of analyzed attribute results to insert for the pet.

4.2.12 PhotoDTO

4.2.12.1 PetPhotoResponse

No	Name	Description
Attributes		
01	Photoid	Visibility: public

		Type: int Purpose: Stores the unique identifier of the photo.
02	PetId	Visibility: public Type: int Purpose: Stores the identifier of the pet associated with the photo.
03	Url	Visibility: public Type: string Purpose: Stores the URL of the pet's photo. Cannot be null.
04	IsPrimary	Visibility: public Type: bool Purpose: Indicates whether this photo is the primary one for the pet.
05	SortOrder	Visibility: public Type: int Purpose: Stores the display order index of the photo.

4.2.12.2 ReorderPhotoRequest

No	Name	Description
Attributes		
01	Photoid	Visibility: public Type: int

		Purpose: Stores the identifier of the photo being reordered.
02	SortOrder	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the new sort/display order for the photo.</p>

4.2.13 RefreshTokenRequest

4.2.13.1 RefreshTokenRequest

No	Name	Description
Attributes		
01	RefreshToken	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the refresh token provided by the user. This field is required and cannot be empty.</p>

4.2.14 ReportDTO

4.2.14.1 ReportDto

No	Name	Description
Attributes		
01	ReportId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the report.</p>
02	Reason	<p>Visibility: public</p> <p>Type: string?</p>

		Purpose: Stores the reason why the report was created.
03	Status	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the current status of the report.</p>
04	Resolution	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores details of how the report was resolved.</p>
05	CreatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the report was created.</p>
06	UpdatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the report was last updated.</p>
07	UserReport	<p>Visibility: public</p> <p>Type: UserReportDto?</p> <p>Purpose: Stores information about the user who created the report.</p>
08	ReportedUser	<p>Visibility: public</p> <p>Type: UserReportDto?</p> <p>Purpose: Stores information about the user being reported.</p>

4.2.13.2 UserReportDto

No	Name	Description
Attributes		
01	UserId	Visibility: public Type: int Purpose: Stores the unique identifier of the user.
02	FullName	Visibility: public Type: string? Purpose: Stores the full name of the user.
03	Email	Visibility: public Type: string Purpose: Stores the email address of the user. Cannot be null.

4.2.13.3 ReportCreateDTO

No	Name	Description
Attributes		
01	Reason	Visibility: public Type: string Purpose: Stores the reason for creating the report. Required field.

4.2.13.4 ReportUpdateDTO

No	Name	Description
Attributes		
01	Status	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the updated status of the report.</p>
02	Resolution	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores updated resolution information for the report.</p>

4.2.15 UserDTO

4.2.15.1 UserResponse

No	Name	Description
Attributes		
01	UserId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the user.</p>
02	RoleId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the role ID assigned to the user.</p>
03	UserStatusId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the status ID of the user.</p>

04	AddressId	Visibility: public Type: int? Purpose: Stores the ID of the user's address.
05	FullName	Visibility: public Type: string? Purpose: Stores the full name of the user.
06	Gender	Visibility: public Type: string? Purpose: Stores the gender of the user.
07	Email	Visibility: public Type: string Purpose: Stores the user's email address. Cannot be null.
08	ProviderLogin	Visibility: public Type: string? Purpose: Stores the provider used for login (Google, Facebook, etc.).
09	isProfileComplete	Visibility: public Type: bool Purpose: Indicates whether the user's profile is fully completed.

10	IsDeleted	Visibility: public Type: bool Purpose: Indicates whether the user is marked as deleted.
11	CreatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the user was created.
12	UpdatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the user was last updated.

4.2.15.2 UserCreateRequest

No	Name	Description
Attributes		
01	RoleId	Visibility: public Type: int? Purpose: Ignored. Included only for backward compatibility.
02	UserStatusId	Visibility: public Type: int? Purpose: Stores the initial status of the user.

03	FullName	Visibility: public Type: string? Purpose: Stores the user's full name. Required.
04	Gender	Visibility: public Type: string? Purpose: Stores the user's gender.
05	Email	Visibility: public Type: string Purpose: Stores the user's email. Required and must be valid.
06	Password	Visibility: public Type: string Purpose: Stores the user's password. Required.
07	ProviderLogin	Visibility: public Type: string? Purpose: Stores the external login provider, if any.
08	isDelete	Visibility: public Type: bool Purpose: Indicates whether the user is marked as deleted.
09	isProfileComplete	Visibility: public Type: bool

		Purpose: Indicates whether the profile is completed.
--	--	--

4.2.15.3 UserUpdateRequest

No	Name	Description
Attributes		
01	AddressId	Visibility: public Type: int? Purpose: Stores the user's address ID.
02	FullName	Visibility: public Type: string? Purpose: Stores the updated full name. Required.
03	Gender	Visibility: public Type: string? Purpose: Stores the updated gender.
04	NewPassword	Visibility: public Type: string? Purpose: Stores the new password if the user requests to change it.

4.2.15.4 AdUserUpdateRequest

No	Name	Description
Attributes		

01	RoleId	Visibility: public Type: int? Purpose: Stores the user's updated role.
02	isDelete	Visibility: public Type: bool? Purpose: Indicates whether the user is deleted.
03	userStatusId	Visibility: public Type: int? Purpose: Stores the user's updated status ID.

4.2.15.5 AdUserCreateRequest

No	Name	Description
Attributes		
01	RoleId	Visibility: public Type: int? Purpose: Stores the user role.
02	UserStatusId	Visibility: public Type: int? Purpose: Stores the user status.

03	FullName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the user's full name. Required.</p>
04	Gender	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the user's gender.</p>
05	Email	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the email. Required and must be valid.</p>
06	Password	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the user's password. Required.</p>
07	isDelete	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the account is deleted.</p>
08	IsProfileComplete	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Indicates whether the profile is completed.</p>

4.2.15.6 ResetPasswordRequest

No	Name	Description
Attributes		
01	Email	Visibility: public Type: string Purpose: Stores the email used for password reset. Required.
02	NewPassword	Visibility: public Type: string Purpose: Stores the new password. Required.

4.2.15.7 ChangePasswordRequest

No	Name	Description
Attributes		
01	CurrentPassword	Visibility: public Type: string Purpose: Stores the current password for verification.
02	NewPassword	Visibility: public Type: string Purpose: Stores the new password. Must meet length requirements.

4.2.16 UserPreferenceDTO

4.2.16.1 UserPreferenceResponse

No	Name	Description
Attributes		
01	AttributeId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the attribute.</p>
02	AttributeName	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the name of the attribute. Cannot be null.</p>
03	TypeValue	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the type of the attribute (e.g., range, option).</p>
04	Unit	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the unit of measurement for numeric attributes.</p>
05	OptionId	<p>Visibility: public</p> <p>Type: int?</p>

		Purpose: Stores the selected option ID for option-based attributes.
06	OptionName	Visibility: public Type: string? Purpose: Stores the name of the selected option.
07	MaxValue	Visibility: public Type: int? Purpose: Stores the maximum accepted value for range attributes.
08	MinValue	Visibility: public Type: int? Purpose: Stores the minimum accepted value for range attributes.
09	CreatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the preference was created.
10	UpdatedAt	Visibility: public Type: DateTime?

		Purpose: Stores the timestamp when the preference was last updated.
--	--	---

4.2.16.2 UserPreferenceUpsertRequest

No	Name	Description
Attributes		
01	OptionId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the selected option ID when updating or inserting a preference.</p>
02	MinValue	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the minimum value for range-type preferences.</p>
03	MaxValue	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the maximum value for range-type preferences.</p>

4.2.16.3 UserPreferenceBatchRequest

No	Name	Description
Attributes		
01	AttributId	Visibility: public

		Type: int Purpose: Stores the identifier of the attribute being configured. Required.
02	OptionId	Visibility: public Type: int? Purpose: Stores the selected option for string/option-type attributes.
03	MinValue	Visibility: public Type: int? Purpose: Stores the minimum value for numeric/range attributes.
04	MaxValue	Visibility: public Type: int? Purpose: Stores the maximum value for numeric/range attributes.

4.2.16.4 UserPreferenceBatchUpsertRequest

No	Name	Description
Attributes		
01	Preferences	Visibility: public Type: List Purpose: Stores a list of attribute preference settings to be inserted or updated in batch. Required.

4.2.17 AppointmentDTO

4.2.17.1 CreateAppointmentRequest

No	Name	Description
Attributes		
01	MatchId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the match identifier related to this appointment request.</p>
02	InviterPetId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the pet ID of the inviter who initiates the appointment.</p>
03	InviteePetId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the pet ID of the invitee who receives the appointment invitation.</p>
04	AppointmentDateTime	<p>Visibility: public</p> <p>Type: DateTime</p> <p>Purpose: Stores the requested appointment date and time.</p>
05	LocationId	<p>Visibility: public</p> <p>Type: int?</p>

		Purpose: Stores the selected location ID from an existing list (optional).
06	CustomLocation	<p>Visibility: public</p> <p>Type: CreateLocationRequest?</p> <p>Purpose: Stores custom location details when the user does not select a predefined location.</p>
07	ActivityType	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the activity type of the appointment (walk, cafe, playdate, park, other).</p>

4.2.17.2 RespondAppointmentRequest

No	Name	Description
Attributes		
01	AppointmentId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the appointment identifier to respond to.</p>
02	Accept	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the appointment is accepted (true) or declined (false).</p>
03	DeclineReason	Visibility: public

		<p>Type: string?</p> <p>Purpose: Stores the decline reason (required when Accept is false).</p>
--	--	---

4.2.17.3 CounterOfferRequest

No	Name	Description
Attributes		
01	AppointmentId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the appointment identifier for the counter-offer action.</p>
02	NewDateTime	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the newly proposed appointment date and time (optional).</p>
03	NewLocationId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the newly proposed predefined location ID (optional).</p>
04	NewCustomLocation	<p>Visibility: public</p> <p>Type: CreateLocationRequest?</p> <p>Purpose: Stores the newly proposed custom location details (optional).</p>

4.2.17.4 CancelAppointmentRequest

No	Name	Description
Attributes		
01	AppointmentId	Visibility: public Type: int Purpose: Stores the appointment identifier to cancel.
02	Reason	Visibility: public Type: string Purpose: Stores the cancellation reason provided by the user.

4.2.17.5 CheckInRequest

No	Name	Description
Attributes		
01	AppointmentId	Visibility: public Type: int Purpose: Stores the appointment identifier for check-in.
02	Latitude	Visibility: public Type: decimal Purpose: Stores the GPS latitude coordinate used for check-in validation.
03	Longitude	Visibility: public Type: decimal

		Purpose: Stores the GPS longitude coordinate used for check-in validation.
--	--	--

4.2.17.6 CreateLocationRequest

No	Name	Description
Attributes		
01	Name	Visibility: public Type: string Purpose: Stores the location name.
02	Address	Visibility: public Type: string Purpose: Stores the full address string of the location.
03	Latitude	Visibility: public Type: decimal Purpose: Stores the latitude coordinate of the location.
04	Longitude	Visibility: public Type: decimal Purpose: Stores the longitude coordinate of the location.
05	City	Visibility: public Type: string? Purpose: Stores the city name (optional).

06	District	Visibility: public Type: string? Purpose: Stores the district name (optional).
07	PlaceType	Visibility: public Type: string? Purpose: Stores the location category/type (optional).
08	GooglePlaceId	Visibility: public Type: string? Purpose: Stores the Google Place ID for mapping integration (optional).

4.2.17.7 AppointmentResponse

No	Name	Description
Attributes		
01	AppointmentId	Visibility: public Type: int Purpose: Stores the unique identifier of the appointment.
02	MatchId	Visibility: public Type: int Purpose: Stores the match identifier associated with the appointment.

03	InviterPetId	Visibility: public Type: int Purpose: Stores the inviter pet identifier.
04	InviterPetName	Visibility: public Type: string? Purpose: Stores the inviter pet name (optional).
05	InviterUserId	Visibility: public Type: int Purpose: Stores the inviter user identifier.
06	InviterUserName	Visibility: public Type: string? Purpose: Stores the inviter user name (optional).
07	InviteePetId	Visibility: public Type: int Purpose: Stores the invitee pet identifier.
08	InviteePetName	Visibility: public Type: string? Purpose: Stores the invitee pet name (optional).
09	InviteeUserId	Visibility: public Type: int

		Purpose: Stores the invitee user identifier.
10	InviteeUserName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the invitee user name (optional).</p>
11	AppointmentDateTime	<p>Visibility: public</p> <p>Type: DateTime</p> <p>Purpose: Stores the scheduled appointment date and time.</p>
12	Location	<p>Visibility: public</p> <p>Type: LocationResponse?</p> <p>Purpose: Stores the appointment location details (optional).</p>
13	ActivityType	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the activity type of the appointment.</p>
14	Status	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the current status of the appointment.</p>
15	CurrentDecisionUserId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the user ID who currently needs to make the next decision (optional).</p>

16	CounterOfferCount	Visibility: public Type: int Purpose: Stores the number of counter-offers made for this appointment.
17	InviterCheckedIn	Visibility: public Type: bool Purpose: Indicates whether the inviter has checked in.
18	InviteeCheckedIn	Visibility: public Type: bool Purpose: Indicates whether the invitee has checked in.
19	InviterCheckInTime	Visibility: public Type: DateTime? Purpose: Stores the inviter check-in timestamp (optional).
20	InviteeCheckInTim e	Visibility: public Type: DateTime? Purpose: Stores the invitee check-in timestamp (optional).
21	CancelledBy	Visibility: public Type: int? Purpose: Stores the user ID who canceled the appointment (optional).

22	CancelReason	Visibility: public Type: string? Purpose: Stores the cancellation reason (optional).
23	CreatedAt	Visibility: public Type: DateTime Purpose: Stores the timestamp when the appointment was created.
24	UpdatedAt	Visibility: public Type: DateTime Purpose: Stores the timestamp when the appointment was last updated.

4.2.17.8 LocationResponse

No	Name	Description
Attributes		
01	LocationId	Visibility: public Type: int Purpose: Stores the unique identifier of the location.
02	Name	Visibility: public Type: string Purpose: Stores the location name.

03	Address	Visibility: public Type: string Purpose: Stores the full address string of the location.
04	Latitude	Visibility: public Type: decimal Purpose: Stores the latitude coordinate of the location.
05	Longitude	Visibility: public Type: decimal Purpose: Stores the longitude coordinate of the location.
06	City	Visibility: public Type: string? Purpose: Stores the city name (optional).
07	District	Visibility: public Type: string? Purpose: Stores the district name (optional).
08	IsPetFriendly	Visibility: public Type: bool Purpose: Indicates whether the location is marked as pet-friendly.
09	PlaceType	Visibility: public

		Type: string? Purpose: Stores the location type/category (optional).
10	GooglePlaceId	Visibility: public Type: string? Purpose: Stores the Google Place ID for mapping integration (optional).

4.2.17.9 AppointmentCardDto

No	Name	Description
Attributes		
01	AppointmentId	Visibility: public Type: int Purpose: Stores the appointment identifier displayed in the chat invitation card.
02	InviterPetName	Visibility: public Type: string Purpose: Stores the inviter pet name displayed on the card.
03	InviteePetName	Visibility: public Type: string Purpose: Stores the invitee pet name displayed on the card.
04	AppointmentDateTime	Visibility: public Type: DateTime

		Purpose: Stores the appointment date and time displayed on the card.
05	LocationName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the display location name for the card (optional).</p>
06	ActivityType	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the activity type displayed on the card.</p>
07	Status	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the current appointment status displayed on the card.</p>
08	CanRespond	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the current user can respond (accept/decline) to the appointment.</p>
09	CanCounterOffer	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the current user can make a counter-offer for this appointment.</p>

10	CanCheckIn	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the current user can perform check-in for this appointment.</p>
----	------------	---

4.2.18 EventDTO

4.2.18.1 CreateEventRequest

No	Name	Description
Attributes		
01	Title	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the title of the event created by admin.</p>
02	Description	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the detailed description of the event (optional).</p>
03	CoverImageUrl	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the URL of the event cover image (optional).</p>
04	StartTime	<p>Visibility: public</p> <p>Type: DateTime</p> <p>Purpose: Stores the start time of the event.</p>

05	SubmissionDeadline	Visibility: public Type: DateTime Purpose: Stores the deadline for submitting entries to the event.
06	EndTime	Visibility: public Type: DateTime Purpose: Stores the end time of the event.
07	PrizeDescription	Visibility: public Type: string? Purpose: Stores the description of prizes for the event (optional).
08	PrizePoints	Visibility: public Type: int? Purpose: Stores the reward points assigned to the event winners (optional).

4.2.18.2 UpdateEventRequest

No	Name	Description
Attributes		
01	Title	Visibility: public Type: string? Purpose: Stores the updated event title (optional).

02	Description	Visibility: public Type: string? Purpose: Stores the updated event description (optional).
03	CoverImageUrl	Visibility: public Type: string? Purpose: Stores the updated event cover image URL (optional).
04	StartTime	Visibility: public Type: DateTime? Purpose: Stores the updated start time of the event (optional).
05	SubmissionDeadline	Visibility: public Type: DateTime? Purpose: Stores the updated submission deadline (optional).
06	EndTime	Visibility: public Type: DateTime? Purpose: Stores the updated end time of the event (optional).
07	PrizeDescription	Visibility: public Type: string? Purpose: Stores the updated prize description (optional).
08	PrizePoints	Visibility: public

		<p>Type: int?</p> <p>Purpose: Stores the updated prize points for the event (optional).</p>
--	--	---

4.2.18.3 SubmitEntryRequest

No	Name	Description
Attributes		
01	EventId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the event to submit an entry to.</p>
02	PetId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the pet identifier associated with the submission.</p>
03	MediaUrl	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the URL of the submitted media (image or video).</p>
04	MediaType	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the MIME type of the submitted media (e.g., image/jpeg, video/mp4).</p>

05	MediaSize	<p>Visibility: public</p> <p>Type: long?</p> <p>Purpose: Stores the media file size in bytes (maximum 50MB, optional).</p>
06	ThumbnailUrl	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the thumbnail URL for video submissions (optional).</p>
07	Caption	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the caption or description for the submission (optional).</p>

4.2.18.4 EventResponse

No	Name	Description
Attributes		
01	EventId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the event.</p>
02	Title	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the event title.</p>

03	Description	Visibility: public Type: string? Purpose: Stores the event description (optional).
04	CoverImageUrl	Visibility: public Type: string? Purpose: Stores the event cover image URL (optional).
05	StartTime	Visibility: public Type: DateTime Purpose: Stores the start time of the event.
06	SubmissionDeadline	Visibility: public Type: DateTime Purpose: Stores the submission deadline of the event.
07	EndTime	Visibility: public Type: DateTime Purpose: Stores the end time of the event.
08	Status	Visibility: public Type: string Purpose: Stores the current lifecycle status of the event (upcoming, active, submission_closed, voting_ended).
09	PrizeDescription	Visibility: public

		Type: string? Purpose: Stores the prize description of the event (optional).
10	PrizePoints	Visibility: public Type: int Purpose: Stores the reward points assigned to winners.
11	SubmissionCount	Visibility: public Type: int Purpose: Stores the total number of submissions for the event.
12	TotalVotes	Visibility: public Type: int Purpose: Stores the total number of votes across all submissions.
13	CreatedAt	Visibility: public Type: DateTime Purpose: Stores the timestamp when the event was created.

4.2.18.5 EventDetailResponse

No	Name	Description
Attributes		
01	CreatedBy Name	Visibility: public Type: string?

		Purpose: Stores the name of the user/admin who created the event (optional).
02	Submissions	<p>Visibility: public</p> <p>Type: IEnumerable<SubmissionResponse>?</p> <p>Purpose: Stores the list of all submissions for the event (optional).</p>
03	Winners	<p>Visibility: public</p> <p>Type: IEnumerable<SubmissionResponse>?</p> <p>Purpose: Stores the list of winning submissions for the event (optional).</p>

4.2.18.6 SubmissionResponse

No	Name	Description
Attributes		
01	SubmissionId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the unique identifier of the submission.</p>
02	EventId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the identifier of the related event.</p>
03	UserId	<p>Visibility: public</p> <p>Type: int</p>

		Purpose: Stores the user identifier who submitted the entry.
04	UserName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the user name of the submitter (optional).</p>
05	UserAvatar	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the avatar URL of the submitter (optional).</p>
06	PetId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the pet identifier associated with the submission.</p>
07	PetName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the pet name (optional).</p>
08	PetPhotoUrl	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the pet photo URL (optional).</p>
09	MediaUrl	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the media URL of the submission.</p>

10	MediaType	Visibility: public Type: string Purpose: Stores the media MIME type of the submission.
11	ThumbnailUrl	Visibility: public Type: string? Purpose: Stores the thumbnail URL for video submissions (optional).
12	Caption	Visibility: public Type: string? Purpose: Stores the caption text for the submission (optional).
13	VoteCount	Visibility: public Type: int Purpose: Stores the total number of votes received by the submission.
14	Rank	Visibility: public Type: int? Purpose: Stores the ranking position of the submission (optional).
15	IsWinner	Visibility: public Type: bool Purpose: Indicates whether the submission is a winning entry.

16	HasVoted	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the current user has voted for this submission.</p>
17	IsOwner	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the current user is the owner of this submission.</p>
18	CreatedAt	<p>Visibility: public</p> <p>Type: DateTime</p> <p>Purpose: Stores the timestamp when the submission was created.</p>

4.2.18.7 LeaderboardResponse

No	Name	Description
Attributes		
01	Rank	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the ranking position in the leaderboard.</p>
02	Submission	<p>Visibility: public</p> <p>Type: SubmissionResponse</p> <p>Purpose: Stores the submission data associated with this leaderboard rank.</p>

4.2.19 PolicyDTO

4.2.19.1 CreatePolicyRequest

No	Name	Description
Attributes		
01	PolicyCode	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the unique policy code used to identify a policy (uppercase letters and underscore).</p>
02	PolicyName	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the display name of the policy.</p>
03	Description	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the policy description (optional).</p>
04	DisplayOrder	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the display ordering index used for sorting policies in UI; default value is 0.</p>
05	RequireConsent	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether user consent is required for this policy; default value is true.</p>

4.2.19.2 UpdatePolicyRequest

No	Name	Description
Attributes		
01	PolicyName	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the updated policy name (optional).</p>
02	Description	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the updated policy description (optional).</p>
03	DisplayOrder	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the updated display order value (optional).</p>
04	RequireConsent	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Stores the updated consent requirement flag (optional).</p>
05	IsActive	<p>Visibility: public</p> <p>Type: bool?</p> <p>Purpose: Stores the updated activation status of the policy (optional).</p>

4.2.19.3 CreatePolicyVersionRequest

No	Name	Description
Attributes		
01	Title	Visibility: public Type: string Purpose: Stores the title of the new policy version.
02	Content	Visibility: public Type: string Purpose: Stores the content/body of the policy version.
03	ChangeLog	Visibility: public Type: string? Purpose: Stores the change log describing differences from the previous version (optional).

4.2.19.4 UpdatePolicyVersionRequest

No	Name	Description
Attributes		
01	Title	Visibility: public Type: string? Purpose: Stores the updated title of the policy version (optional, draft only).
02	Content	Visibility: public Type: string?s

		Purpose: Stores the updated content of the policy version (optional, draft only).
03	ChangeLog	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the updated change log of the policy version (optional, draft only).</p>

4.2.19.5 AcceptPolicyRequest

No	Name	Description
Attributes		
01	PolicyCode	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the policy code the user is accepting.</p>
02	VersionNumber	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the policy version number that the user accepts.</p>

4.2.19.6 AcceptMultiplePoliciesRequest

No	Name	Description
Attributes		
01	Policies	<p>Visibility: public</p> <p>Type: List<AcceptPolicyRequest></p>

		Purpose: Stores the list of policy acceptance requests to be confirmed in a single operation.
--	--	---

4.2.19.7 PolicyResponse

No	Name	Description
Attributes		
01	PolicyId	Visibility: public Type: int Purpose: Stores the unique identifier of the policy.
02	PolicyCode	Visibility: public Type: string Purpose: Stores the policy code used for identification.
03	PolicyName	Visibility: public Type: string Purpose: Stores the policy display name.
04	Description	Visibility: public Type: string? Purpose: Stores the policy description (optional).
05	DisplayOrder	Visibility: public Type: int Purpose: Stores the display order index for UI sorting.

06	RequireConsent	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether user consent is required for this policy.</p>
07	IsActive	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the policy is currently active.</p>
08	CreatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the policy was created (optional).</p>
09	UpdatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the timestamp when the policy was last updated (optional).</p>
10	ActiveVersion	<p>Visibility: public</p> <p>Type: PolicyVersionResponse?</p> <p>Purpose: Stores the currently active policy version (optional, if available).</p>
11	TotalVersions	<p>Visibility: public</p> <p>Type: int</p>

		Purpose: Stores the total number of versions belonging to this policy.
--	--	--

4.2.19.8 PolicyVersionResponse

No	Name	Description
Attributes		
01	PolicyVersionId	Visibility: public Type: int Purpose: Stores the unique identifier of the policy version.
02	PolicyId	Visibility: public Type: int Purpose: Stores the parent policy identifier.
03	PolicyCode	Visibility: public Type: string Purpose: Stores the policy code related to this version.
04	VersionNumber	Visibility: public Type: int Purpose: Stores the policy version number.
05	Title	Visibility: public Type: string Purpose: Stores the title of the policy version.

06	Content	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the content/body of the policy version.</p>
07	ChangeLog	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the change log for the version (optional).</p>
08	Status	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the status of the policy version (e.g., DRAFT, PUBLISHED, DEACTIVATED).</p>
09	PublishedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the published timestamp of the version (optional).</p>
10	DeactivatedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the deactivated timestamp of the version (optional).</p>
11	CreatedByUserId	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the user ID who created the version (optional).</p>

12	CreatedByUserName	Visibility: public Type: string? Purpose: Stores the name of the user who created the version (optional).
13	CreatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the version was created (optional).
14	UpdatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when the version was last updated (optional).

4.2.19.9 PendingPolicyResponse

No	Name	Description
Attributes		
01	PolicyCode	Visibility: public Type: string Purpose: Stores the policy code that requires user acceptance.
02	PolicyName	Visibility: public Type: string Purpose: Stores the policy name requiring acceptance.

03	Description	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the policy description (optional).</p>
04	DisplayOrder	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the display order for showing pending policies in UI.</p>
05	VersionNumber	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the version number that the user must accept.</p>
06	Title	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the title of the version requiring acceptance.</p>
07	Content	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the content of the version requiring acceptance.</p>
08	ChangeLog	<p>Visibility: public</p> <p>Type: string?</p> <p>Purpose: Stores the change log of the pending version (optional).</p>

09	PublishedAt	<p>Visibility: public</p> <p>Type: DateTime?</p> <p>Purpose: Stores the published timestamp of the pending version (optional).</p>
10	HasPreviousAccept	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the user has previously accepted an older version of this policy.</p>
11	PreviousAcceptVersion	<p>Visibility: public</p> <p>Type: int?</p> <p>Purpose: Stores the previously accepted version number by the user (optional).</p>

4.2.19.10 PolicyStatusResponse

No	Name	Description
Attributes		
01	IsCompliant	<p>Visibility: public</p> <p>Type: bool</p> <p>Purpose: Indicates whether the user is compliant and eligible to use the application.</p>
02	Status	<p>Visibility: public</p> <p>Type: string</p>

		Purpose: Stores the compliance status (e.g., ACTIVE, PENDING_POLICY).
03	Message	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores a descriptive message for the compliance status.</p>
04	PendingPolicies	<p>Visibility: public</p> <p>Type: List<PendingPolicyResponse></p> <p>Purpose: Stores the list of policies the user must accept (empty if compliant).</p>

4.2.19.11 UserAcceptHistoryResponse

No	Name	Description
Attributes		
01	AcceptId	<p>Visibility: public</p> <p>Type: long</p> <p>Purpose: Stores the unique identifier of a user acceptance record.</p>
02	PolicyCode	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the policy code that the user accepted.</p>
03	PolicyName	Visibility: public

		Type: string Purpose: Stores the policy name that the user accepted.
04	VersionNumber	Visibility: public Type: int Purpose: Stores the version number that the user accepted.
05	VersionTitle	Visibility: public Type: string Purpose: Stores the title of the accepted version.
06	AcceptedAt	Visibility: public Type: DateTime Purpose: Stores the timestamp when the user accepted the policy version.
07	IsValid	Visibility: public Type: bool Purpose: Indicates whether this acceptance record is still valid for current compliance.
08	InvalidatedAt	Visibility: public Type: DateTime? Purpose: Stores the timestamp when this acceptance became invalid (optional).

4.2.19.12 PolicyAcceptStatsResponse

No	Name	Description
Attributes		
01	PolicyId	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the policy identifier for statistics reporting.</p>
02	PolicyCode	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the policy code for statistics reporting.</p>
03	PolicyName	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the policy name for statistics reporting.</p>
04	ActiveVersionNumber	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the current active version number of the policy.</p>
05	TotalActiveUsers	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the total number of active users in the system.</p>
06	AcceptedUsers	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the number of users who accepted the current active version.</p>

07	PendingUsers	<p>Visibility: public</p> <p>Type: int</p> <p>Purpose: Stores the number of users who have not accepted the current active version.</p>
08	AcceptRate	<p>Visibility: public</p> <p>Type: double</p> <p>Purpose: Stores the acceptance rate percentage for the current active version.</p>

4.2.19.13 PolicyRequiredErrorResponse

No	Name	Description
Attributes		
01	ErrorCode	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the application error code for missing policy acceptance; default is "POLICY_REQUIRED".</p>
02	Message	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the default error message indicating that policy acceptance is required.</p>
03	PendingPolicies	<p>Visibility: public</p> <p>Type: List<PendingPolicyResponse></p>

		Purpose: Stores the list of pending policies required for the user to continue using the application.
--	--	---

4.2.20 CloudinaryPhotoStorage

4.2.20.1 CloudinaryPhotoStorage class

No	Name	Description
Attributes		
01	_cloudinary	<p>Visibility: private (readonly)</p> <p>Type: Cloudinary</p> <p>Purpose: Stores the Cloudinary client instance used to upload and delete images on the Cloudinary platform.</p>
02	_settings	<p>Visibility: private (readonly)</p> <p>Type: CloudinarySettings</p> <p>Purpose: Stores Cloudinary configuration settings such as root folder path for uploaded images.</p>
03	_allowed	<p>Visibility: private (static, readonly)</p> <p>Type: HashSet<string></p> <p>Purpose: Stores the list of allowed MIME content types for image uploads (image/jpeg, image/png, image/webp).</p>
Methods/Operations		
01	CloudinaryPhotoStorage(Cloudinary cloudinary, IOptions<CloudinarySettings> settings)	<p>Visibility: public</p> <p>Type: Constructor</p> <p>Purpose: Initializes the Cloudinary photo storage service with injected Cloudinary client and configuration settings.</p>

02	UploadAsync(int petId, IFormFile file, CancellationToken ct = default)	Visibility: public Type: Task<(string Url, string PublicId)> Purpose: Uploads an image file to Cloudinary under a pet-specific folder, validates file type and size, applies automatic image optimization (quality and format), and returns the secure image URL and public ID.
03	DeleteAsync(string publicId, CancellationToken ct = default)	Visibility: public Type: Task Purpose: Deletes an uploaded image from Cloudinary using its public ID; safely ignores empty or invalid public IDs.

4.2.21 BanRequests

4.2.21.1 BanUserRequest

No	Name	Description
Attributes		
01	Reason	Visibility: public Type: string? Purpose: Stores the reason for banning the user (optional, used for moderation records and audit logs).
02	DurationDays	Visibility: public Type: int Purpose: Stores the ban duration in days when the ban is temporary; default value is 1 day.
03	IsPermanent	Visibility: public Type: bool Purpose: Indicates whether the ban is permanent; when true, the ban has no end date (BanEnd = null).

4.3 Controller

4.3.1 AddressController

No	Name	Description
Attributes		
01	_addressService	<p>Visibility: private readonly</p> <p>Type: IAddressService</p> <p>Purpose: Provides business logic for handling user address operations.</p>
Methods/Operations		
01	CreateAddressForUser	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Handles POST requests to create an address for a specific user.</p> <p>Parameters:- userId: int – The ID of the user for whom the address is created.- locationDto: LocationDto – The address data to create. ct: CancellationToken – Optional cancellation token.</p>
02	UpdateAddress	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Handles PUT requests to update an existing address.</p> <p>Parameters:- addressId: int – The ID of the address to update.- locationDto: LocationDto – Updated address information. ct: CancellationToken – Optional cancellation token.</p>
03	UpdateAddressManual	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Handles PATCH requests to manually update address details.</p> <p>Parameters:- addressId: int – The ID of the address to update.- dto: ManualAddressDto – Manual address data to apply.- ct: CancellationToken – Optional cancellation token.</p>

04	GetAddressById	<p>Visibility: public</p> <p>Return: TaskPurpose: Handles GET requests to retrieve address details by ID.</p> <p>Parameters:- addressId: int – The ID of the address to retrieve.- ct: CancellationToken – Optional cancellation token.</p>
----	----------------	---

4.3.2 AdminsController

No	Name	Description
Attributes		
01	_adminService	<p>Visibility: private readonly</p> <p>Type: IAdminService</p> <p>Purpose: Provides business logic for admin-level user management operations.</p>
Methods/Operations		
01	ReassignExpertConfirmation	<p>Visibility: public</p> <p>Return:</p> <p>Task<ActionResult<ReassignExpertConfirmationResponse>></p> <p>Purpose: Reassigns expert confirmation to another expert user.</p> <p>Parameters:- req: ReassignExpertConfirmationRequest - Contains reassignment details.- ct: CancellationToken - Optional cancellation token.</p>
02	BanUser	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Bans a user with a given ID using the provided ban request.</p> <p>Parameters:- id: int - The ID of the user to ban.- req: BanUserRequest - Contains ban reason and configuration.- ct: CancellationToken</p>

03	UnbanUser	<p>Visibility: public</p> <p>Return: TaskPurpose: Removes ban status from a user.Parameters:- id: int - The ID of the user to unban.- req: UnbanUserRequest? - Optional unban request containing a reason.- ct: CancellationToken</p>
04	GetUserBans	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all ban records associated with a user.Parameters:- id: int - User ID to retrieve ban logs for.- ct: CancellationToken</p>
05	UpdateUserByAdmin	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates user details as an admin.Parameters:- id: int - The ID of the user to update.- request: AdUserUpdateRequest - Updated user information.- ct: CancellationToken</p>
06	Register	<p>Visibility: public</p> <p>Return: Task<ActionResult<UserResponse>>Purpose: Registers a new user through the admin interface.Parameters:- req: AdUserCreateRequest - Contains new user registration details.- ct: CancellationToken</p>

4.3.3 AttributeOptionController

No	Name	Description
Attributes		

01	_optionService	<p>Visibility: private readonlyType: IAttributeOptionService</p> <p>Purpose: Provides business logic for managing attribute options.</p>
Methods/Operations		
01	GetAllOptions	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all attribute options from the system.Parameters:- ct: CancellationToken - Optional cancellation token.</p>
02	GetOptionsByAttribute	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves attribute options for a specific attribute ID.Parameters:- attributId: int - The ID of the attribute.- ct: CancellationToken - Optional cancellation token.</p>
03	CreateOption	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new option under the specified attribute.Parameters:- attributId: int - The attribute ID to attach the option to.- optionName: string - The name of the new option.- ct: CancellationToken - Optional cancellation token.</p>
04	UpdateOptions	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates the name of an existing option.Parameters:- optionId: int - The ID of the option to update.- optionNames: string - The updated option name.- ct: CancellationToken - Optional cancellation token.</p>

05	DeleteOption	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes an attribute option based on its ID.Parameters:- optionId: int - The ID of the option to delete.- ct: CancellationToken - Optional cancellation token.</p>
----	--------------	--

4.3.4 AttributeController

No	Name	Description
Attributes		
01	_attributeService	<p>Visibility: private readonly</p> <p>Type: IAttributeService</p> <p>Purpose: Provides business logic for managing attribute data and operations.</p>
Methods/Operations		
01	GetList	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves a paginated list of attributes with optional filtering and searching.Parameters:- search: string? – Text used to filter attribute names.- page: int – Page number for pagination.- pageSize: int – Number of items per page.- includeDeleted: bool – Whether to include soft-deleted attributes.- ct: CancellationToken – Optional cancellation token.</p>
02	GetById	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves attribute details by its ID.Parameters:- id: int – ID of the attribute to retrieve.- ct: CancellationToken – Optional cancellation token.</p>

03	Create	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new attribute and returns the created resource.Parameters:- request: AttributeCreateRequest – Data for creating the attribute.- ct: CancellationToken – Optional cancellation token.</p>
04	Update	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates an existing attribute by ID.Parameters:- id: int – ID of the attribute to update.- request: AttributeUpdateRequest – Updated attribute data.- ct: CancellationToken – Optional cancellation token.</p>
05	Delete	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes an attribute (soft or hard depending on parameter).Parameters:- id: int – ID of the attribute to delete.- hard: bool – Whether to permanently delete (true) or soft delete (false).- ct: CancellationToken – Optional cancellation token.</p>
06	GetAttributesForFilter	<p>Visibility:</p> <p>public</p> <p>Return: TaskPurpose: Retrieves a lightweight attribute list for filtering operations in the client.Parameters:- ct: CancellationToken – Optional cancellation token.</p>

4.3.5 AuthController

No	Name	Description
Attributes		

01	_authService	<p>Visibility: private readonly</p> <p>Type: IAuthService</p> <p>Purpose: Provides authentication-related business logic such as login, token refresh, password change, and logout.</p>
Methods/Operations		
01	Login	<p>Visibility: public</p> <p>Return: TaskPurpose: Authenticates a user and returns login result containing tokens.</p> <p>Parameters:- request: LoginRequest – Contains login credentials.- ct: CancellationToken – Optional cancellation token.</p>
02	Refresh	<p>Visibility: public</p> <p>Return: TaskPurpose: Refreshes an authentication token using a valid refresh token.</p> <p>Parameters:- request: RefreshTokenRequest – Contains the refresh token.- ct: CancellationToken – Optional cancellation token.</p>
03	Logout	<p>Visibility: public</p> <p>Return: TaskPurpose: Logs out the current authenticated user and invalidates their session/token.</p> <p>Parameters:- ct: CancellationToken – Optional cancellation token.</p>

04	ChangePassword	<p>Visibility: public</p> <p>Return: TaskPurpose: Changes the password for the authenticated user.</p> <p>Parameters:- request: ChangePasswordRequest – Contains current and new passwords.- ct: CancellationToken – Optional cancellation token.</p>
----	----------------	---

4.3.6 BlockController

No	Name	Description
Attributes		
01	_blockService	<p>Visibility: private readonly</p> <p>Type: IBlockService</p> <p>Purpose: Provides business logic for blocking, retrieving, and unblocking users.</p>
Methods/Operations		
01	GetBlockedUsers	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves the list of users blocked by a specific user.</p> <p>Parameters:- fromUserId: int – The ID of the user who is blocking others.- ct: CancellationToken – Optional cancellation token.</p>
02	CreateBlock	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a block relationship between two users.</p> <p>Parameters:- fromUserId: int – ID of the user initiating the block.- toUserId: int – ID of the user being blocked.- ct: CancellationToken – Optional cancellation token.</p>

03	DeleteBlock	<p>Visibility: public</p> <p>Return: TaskPurpose: Removes a block relationship between two users.</p> <p>Parameters:- fromUserId: int – ID of the user who blocked another user.- toUserId: int – ID of the user being unblocked.- ct: CancellationToken – Optional cancellation token.</p>
----	-------------	---

4.3.7 ChatAIController

No	Name	Description
Attributes		
01	_chatAIService	<p>Visibility: private readonly</p> <p>Type: IChatAIService</p> <p>Purpose: Provides business logic for chat interactions, message processing, token usage, and conversation management.</p>
Methods/Operations		
01	GetTokenUsage	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves the token usage statistics of the currently authenticated user.</p> <p>Parameters:- ct: CancellationToken - Optional cancellation token.</p>
02	GetAllChats	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all chat sessions belonging to the specified user. Only the owner can access their own chats unless Expert/Admin.</p> <p>Parameters:- userId: int - ID of the user whose chats are being requested.- ct: CancellationToken - Optional cancellation token.</p>

03	CreateChat	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new chat session for the specified user.Parameters:- userId: int - ID of the user creating the chat.- request: CreateChatRequest - Contains the chat title.- ct: CancellationToken - Optional cancellation token.</p>
04	UpdateChatTitle	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates the title of an existing chat session.Parameters:- chatAild: int - ID of the chat to update.- request: UpdateChatTitleRequest - Contains the new title.- ct: CancellationToken - Optional cancellation token.</p>
05	DeleteChat	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes a chat session for the authenticated user.Parameters:- chatAild: int - ID of the chat to delete.- ct: CancellationToken - Optional cancellation token.</p>
06	GetChatHistory	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves message history of a chat. Admin/Expert can view any chat; normal users only their own.Parameters:- chatAild: int - ID of the chat.- ct: CancellationToken - Optional cancellation token.</p>

07	SendMessage	<p>Visibility: public</p> <p>Return: TaskPurpose: Sends a message to the AI within an existing chat session and receives a response.</p> <p>Parameters:-</p> <ul style="list-style-type: none"> chatAId: int - ID of the chat session. - request: SendMessageRequest - Contains the message/question to send. - ct: CancellationToken - Optional cancellation token.
----	-------------	---

4.3.8 ChatExpertContentController

No	Name	Description
Attributes		
01	_contentService	<p>Visibility: private readonly</p> <p>Type: IChatExpertContentService</p> <p>Purpose: Provides business logic for managing expert chat messages, including retrieving chat history and sending new messages.</p>
Methods/Operations		
01	GetChatMessages	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all chat messages associated with a specific Expert Chat session.</p> <p>Parameters:-</p> <ul style="list-style-type: none"> chatExpertId: int - The ID of the expert chat session. - ct: CancellationToken - Optional cancellation token.
02	SendMessage	<p>Visibility: public</p> <p>Return: TaskPurpose: Sends a message to an expert chat session on behalf of a user or expert.</p> <p>Parameters:-</p> <ul style="list-style-type: none"> chatExpertId: int - The ID of the expert chat session. - fromId: int - ID of the sender (user or expert). - request: SendMessageRequestChatExpert - Contains message content, sender role, and related IDs. - ct: CancellationToken - Optional cancellation token.

4.3.9 ChatExpertController

No	Name	Description
Attributes		
01	_chatExpertService	<p>Visibility: private readonly</p> <p>Type: IChatExpertService</p> <p>Purpose: Provides business logic for expert-user chat creation and retrieval operations.</p>
Methods/Operations		
01	GetChatsByUserId	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Retrieves all expert chat sessions belonging to a specific user.</p> <p>Parameters:- userId: int - The ID of the user whose chats are being retrieved.- ct: CancellationToken - Optional cancellation token.</p>
02	GetChatsByExpertId	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Retrieves chat sessions assigned to an expert. Only chats already created are returned; no automatic creation occurs.</p> <p>Parameters:- expertId: int - The ID of the expert whose chat sessions are being retrieved.- ct: CancellationToken - Optional cancellation token.</p>
03	CreateChat	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Creates a new chat session between an expert and a user if it doesn't already exist. If it exists, returns the existing session.</p> <p>Parameters:- expertId: int - The ID of the expert to chat with.- userId: int - The ID of the user starting the chat.- ct: CancellationToken - Optional cancellation token.</p>

4.3.10 ChatUserContentController

No	Name	Description
Attributes		
01	_contentService	<p>Visibility: private readonly</p> <p>Type: IChatUserContentService</p> <p>Purpose: Provides business logic for retrieving and sending chat messages between matched users.</p>
Methods/Operations		
01	GetChatMessages	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Retrieves all chat messages for a specific match between users.</p> <p>Parameters:- matchId: int – The match ID representing the user pair. - ct: CancellationToken – Optional cancellation token.</p>
02	SendMessage	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Sends a chat message from one matched user to another.</p> <p>Parameters:- matchId: int – The match ID where the message is sent.- fromUserId: int – ID of the user sending the message.- message: string – The message text sent by the user.- ct: CancellationToken – Optional cancellation token.</p>

4.3.11 ChatUserController

No	Name	Description
Attributes		

01	_chatUserService	<p>Visibility: private readonly</p> <p>Type: IChatUserService</p> <p>Purpose: Provides business logic for managing chat sessions, invites, and friend requests between pet owners.</p>
Methods/Operations		
01	GetInvites	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all friend requests (invites) sent to a specific user.Parameters:- userId: int – The ID of the user receiving invites.- ct: CancellationToken – Optional cancellation token.</p>
02	GetChats	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves chat conversations for a user, optionally filtered by pet ID.Parameters:- userId: int – ID of the user whose chats are requested.- petId: int? – Optional filter to get chats related to a specific pet.- ct: CancellationToken – Optional cancellation token.</p>
03	CreateFriendRequest	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a friend request between two pets (pet-to-pet interaction).Parameters:- fromPetId: int – The ID of the pet sending the request.- toPetId: int – The ID of the pet receiving the request.- ct: CancellationToken – Optional cancellation token.</p>

04	UpdateFriendRequest	<p>Visibility: public</p> <p>Return: Task<ActionResultPurpose: Updates an existing friend request (accept or modify status).Parameters:- matchId: int – The ID representing the matched pets.- ct: CancellationToken – Optional cancellation token.</p>
05	DeleteFriendRequest	<p>Visibility: public</p> <p>Return: Task<ActionResultPurpose: Deletes a friend request between two users' pets.Parameters:- matchId: int – The ID of the friend request to delete.- ct: CancellationToken – Optional cancellation token.</p>
06	DeleteChat	<p>Visibility: public</p> <p>Return: Task<ActionResultPurpose: Deletes (hides) a specific chat between two matched pets.Parameters:- matchId: int – The ID of the chat to delete.- ct: CancellationToken – Optional cancellation token.</p>

4.3.12 DailyLimitsController

No	Name	Description
Attributes		
01	_limitService	<p>Visibility: private readonly</p> <p>Type: IDailyLimitService</p> <p>Purpose: Provides business logic for checking and managing user daily action limits.</p>
Methods/Operations		

01	GetRemainingCount	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves the remaining number of allowed actions a user can perform for a given action type on the current day.</p> <p>Parameters:- userId: int – ID of the user whose remaining quota is requested.- actionType: string – The type of action being checked (e.g., "ai_chat_question", "match_request").- ct: CancellationToken – Optional cancellation token.</p>
----	-------------------	---

4.3.13 ExpertController

No	Name	Description
Attributes		
01	_expertConfirmationService	<p>Visibility: private readonly</p> <p>Type: IExpertConfirmationService</p> <p>Purpose: Provides business logic for managing expert confirmation requests, updating confirmations, and retrieving expert-user chat data.</p>
Methods/Operations		
01	GetAllExpertConfirmations	<p>Visibility: public</p> <p>Return: Task<ActionResult<List>></p> <p>Purpose: Retrieves all expert confirmation requests in the system.</p> <p>Parameters:- ct: CancellationToken – Optional cancellation token.</p>

02	GetExpertConfirmation	<p>Visibility: public</p> <p>Return: Task<ActionResult<ExpertConfirmationDTO>>>Purpose: Retrieves a single expert confirmation record using expertId, userId, and chatId.Parameters:- expertId: int – The ID of the expert.- userId: int – The ID of the user requesting confirmation.- chatId: int – The ID of the related chat session.- ct: CancellationToken – Optional cancellation token.</p>
03	GetUserExpertConfirmations	<p>Visibility: public</p> <p>Return: Task<ActionResult<List>>Purpose: Retrieves all expert confirmation requests created by a specific user.Parameters:- userId: int – ID of the user whose confirmations are being retrieved.- ct: CancellationToken – Optional cancellation token.</p>
04	CreateExpertConfirmation	<p>Visibility: public</p> <p>Return:</p> <p>Task<ActionResult<ExpertConfirmationResponseDTO>>>Purpose: Creates a new expert confirmation request for a user related to a specific chat session.Parameters:- userId: int – ID of the user creating the confirmation.- chatId: int – ID of the related chat.- dto: ExpertConfirmationCreateDTO – Data required for creating expert confirmation.- ct: CancellationToken – Optional cancellation token.</p>
05	UpdateExpertConfirmation	<p>Visibility: public</p> <p>Return:</p> <p>Task<ActionResult<ExpertConfirmationResponseDTO>>>Purpose: Updates an existing expert confirmation based on expert, user, and chat IDs.Parameters:- expertId: int – ID of the expert updating the confirmation.- userId: int – ID of the requesting user.- chatId: int – ID of the chat session.- dto: ExpertConfirmationUpdateDto – Updated confirmation information.- ct: CancellationToken – Optional cancellation token.</p>

06	GetUserExpertChats	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all chat sessions between a user and experts.</p> <p>Parameters:- userId: int – The ID of the user whose expert chats are being retrieved.- ct: CancellationToken – Optional cancellation token.</p>
----	--------------------	--

4.3.14 MatchController

No	Name	Description
Attributes		
01	_matchService	<p>Visibility: private readonly</p> <p>Type: IMatchService</p> <p>Purpose: Provides business logic for sending likes, responding to likes, retrieving match stats, received likes, and badge counts.</p>
Methods/Operations		
01	GetLikesReceived	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves a list of likes received by a user, optionally filtered by pet ID.</p> <p>Parameters:- userId: int – The ID of the user who received likes.- petId: int? – Optional pet filter.- ct: CancellationToken – Optional cancellation token.</p>
02	GetStats	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves match-related statistics for a user (likes sent, likes received, matches, etc.).</p> <p>Parameters:- userId: int – The ID of the user whose stats are requested.- ct: CancellationToken – Optional cancellation token.</p>

03	SendLike	<p>Visibility: public</p> <p>Return: TaskPurpose: Sends a like request from one pet to another.Parameters:- request: LikeRequest – Contains sender and receiver pet IDs.- ct: CancellationToken – Optional cancellation token.</p>
04	RespondToLike	<p>Visibility: public</p> <p>Return: TaskPurpose: Responds to a like request (accept or reject).Parameters:- request: RespondRequest – Contains the match ID and response type.- ct: CancellationToken – Optional cancellation token.</p>
05	GetBadgeCounts	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves notification badge counters for user matches (pending invites, unread chats, etc.).Parameters:- userId: int – ID of the user requesting badge counts.- petId: int? – Optional pet filter.- ct: CancellationToken – Optional cancellation token.</p>

4.3.15 NotificationController

No	Name	Description
Attributes		
01	_notificationService	<p>Visibility: private readonly</p> <p>Type: INotificationService</p> <p>Purpose: Provides business logic for retrieving, creating, updating, and deleting user notifications.</p>
Methods/Operations		

01	GetAllNotifications	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all notifications in the system (Admin only).Parameters:- ct: CancellationToken – Optional cancellation token.</p>
02	GetNotificationById	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves a specific notification by its ID.Parameters:- notificationId: int – ID of the notification.- ct: CancellationToken – Optional cancellation token.</p>
03	GetNotificationsByUserId	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all notifications belonging to a specific user.Parameters:- userId: int – ID of the user.- ct: CancellationToken – Optional cancellation token.</p>
04	CreateNotification	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new notification and saves it to the system.Parameters:- notificationDto: NotificationDto_1 – Notification data to create.- ct: CancellationToken – Optional cancellation token.</p>

05	MarkAsRead	<p>Visibility: public</p> <p>Return: TaskPurpose: Marks a notification as read using its ID.Parameters:- notificationId: int – ID of the notification.- ct: CancellationToken – Optional cancellation token.</p>
06	MarkAllAsRead	<p>Visibility: public</p> <p>Return: TaskPurpose: Marks all notifications of a user as read and returns the count of updated items.Parameters:- userId: int – ID of the user.- ct: CancellationToken – Optional cancellation token.</p>
07	GetUnreadCount	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves the number of unread notifications for a user.Parameters:- userId: int – ID of the user.- ct: CancellationToken – Optional cancellation token.</p>
08	DeleteNotification	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes a notification by its ID.Parameters:- notificationId: int – ID of the notification to delete.- ct: CancellationToken – Optional cancellation token.</p>

4.3.16 PaymentHistoryController

No	Name	Description
Attributes		

01	<code>_paymentHistoryService</code>	<p>Visibility: private readonlyType: IPaymentHistoryService</p> <p>Purpose: Provides business logic for handling payment history, QR generation, VIP upgrades, and SePay transaction verification.</p>
Methods/Operations		
01	<code>GetAllPaymentHistories</code>	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all payment histories (Admin only).Parameters:- ct: CancellationToken — optional.</p>
02	<code>GenerateQr</code>	<p>Visibility: public</p> <p>Return: TaskPurpose: Validates payment request and generates a QR code for VIP purchase via SePay.Parameters:- request: GenerateQrRequest – contains Amount and Months . ct: CancellationToken — optional.</p>
03	<code>PaymentCallback</code>	<p>Visibility: public</p> <p>Return: TaskPurpose: Validates SePay transaction callback and verifies payment amount/content.Parameters:- request: PaymentCallbackRequest – contains TransferAmount and Content.- ct: CancellationToken — optional.</p>
04	<code>CreatePaymentHistory</code>	<p>Visibility: public</p> <p>Return: TaskPurpose: Inserts a new payment history record into the system.Parameters:- request: CreatePaymentHistoryRequest – payment creation data.- ct: CancellationToken — optional.</p>

05	GetPaymentHistoryByUserId	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all payment histories belonging to a specific user.Parameters:- userId: int – ID of the user.- ct: CancellationToken — optional.</p>
06	GetVipStatus	<p>Visibility: public</p> <p>Return: TaskPurpose: Checks whether a user currently has VIP status and its expiration.Parameters:- userId: int – ID of the user.- ct: CancellationToken — optional.</p>
07	GetUserIdFromToken	<p>Visibility: private</p> <p>Return: int?Purpose: Extracts the authenticated user's ID from JWT token claims.Parameters: None. (Utility method)</p>

4.3.17 PetCharacteristicController

No	Name	Description
Attributes		
01	_petCharacteristicService	<p>Visibility: private readonly</p> <p>Type: IPetCharacteristicService</p> <p>Purpose: Provides business logic for retrieving, creating, and updating pet characteristics.</p>
Methods/Operations		

01	GetPetCharacteristics	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all characteristics of a pet by its ID.Parameters:- petId: int — ID of the pet.- ct: CancellationToken — optional.</p>
02	CreatePetCharacteristic	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new characteristic for a pet with a given attribute.Parameters:- petId: int — ID of the pet.- attributId: int — ID of the attribute.- dto: PetCharacteristicDTO — characteristic data.- ct: CancellationToken — optional.</p>
03	UpdatePetCharacteristic	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates an existing pet characteristic (based on petId & attributId).Parameters:- petId: int — ID of the pet.- attributId: int — ID of the attribute.- dto: PetCharacteristicDTO — updated characteristic values.- ct: CancellationToken — optional.</p>

4.3.18 PetController

No	Name	Description
Attributes		
01	_petService	<p>Visibility: private readonly</p> <p>Type: IPetService</p> <p>Purpose: Provides business logic for retrieving, creating, updating, deleting pets and managing active pet selection.</p>
Methods/Operations		

01	GetPetsByUser	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves the list of pets that belong to a specific user.Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>
02	GetPetsForMatching	<p>Visibility: public</p> <p>Return: TaskPurpose: Gets all pets of a user, formatted for matching purposes.Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>
03	GetPetById	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves details of a pet by its petId.Parameters:- petId: int — ID of the pet.- ct: CancellationToken — optional.</p>
04	CreatePet	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new pet for a user.Parameters:- petDto: PetDto_2 — pet creation data.- ct: CancellationToken — optional.</p>

05	UpdatePet	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates information of an existing pet.Parameters:- petId: int — ID of the pet.- updatedPet: PetDto_2 — updated pet data.- ct: CancellationToken — optional.</p>
06	DeletePet	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes a pet by its ID.Parameters:- petId: int — ID of the pet.- ct: CancellationToken — optional.</p>
07	SetActivePet	<p>Visibility: public</p> <p>Return: TaskPurpose: Marks a pet as the active/default pet for the user.Parameters:- petId: int — ID of the pet.- ct: CancellationToken — optional.</p>

4.3.18 PetImageAnalysisController

No	Name	Description
Attributes		
01	_analysisService	<p>Visibility: private readonly</p> <p>Type: IPetImageAnalysisService</p> <p>Purpose: Provides logic for analyzing pet images, extracting attributes, and returning structured analysis results.</p>
Methods/Operations		

01	AnalyzeImage	<p>Visibility: public</p> <p>Return: TaskPurpose: Analyzes a single pet image and returns detected attributes.Parameters:- image: IFormFile — uploaded pet image.</p>
02	AnalyzeMultipleImages	<p>Visibility: public</p> <p>Return: TaskPurpose: Analyzes multiple uploaded images and returns a list of attribute analysis results for each image.Parameters:- images: List<IFormFile> — list of uploaded images.</p>

4.3.19 PetPhotoController

No	Name	Description
Attributes		
01	_photoService	<p>Visibility: private readonly</p> <p>Type: IPetPhotoService</p> <p>Purpose: Provides business logic for uploading, retrieving, reordering, and deleting pet photos.</p>
Methods/Operations		
01	GetAllByPet	<p>Visibility: public</p> <p>Return: TaskPurpose: Retrieves all photos associated with a specific pet.Parameters:- petId: int — ID of the pet.- ct: CancellationToken — optional.</p>

02	Upload	<p>Visibility: public</p> <p>Return: TaskPurpose: Uploads one or more photos for a pet.Parameters:- petId: int — ID of the pet.- files: List<IFormFile> — list of uploaded image files.- ct: CancellationToken — optional.</p>
03	Reorder	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates the display order of photos belonging to a pet.Parameters:- items: List<ReorderPhotoRequest> — photo reorder instructions.- ct: CancellationToken — optional.</p>
04	Delete	<p>Visibility: public</p> <p>Return: TaskPurpose: Deletes a pet photo, optionally performing a hard delete.Parameters:- photoId: int — ID of the photo.- hard: bool — if true, permanently deletes the photo.- ct: CancellationToken — optional.</p>

4.3.20 PetRecommendationController

No	Name	Description
Attributes		
01	_petRecommendationService	<p>Visibility: private readonly</p> <p>Type: IPetRecommendationService</p>

		Purpose: Provides business logic for generating pet recommendations tailored to user preferences and pet compatibility.
Methods/Operations		
01	RecommendPets	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Retrieves recommended pets for a specific user based on system-defined matching algorithms.</p> <p>Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>

4.3.21 ReportController

No	Name	Description
Attributes		
01	_reportService	<p>Visibility: private readonly</p> <p>Type: IReportService</p> <p>Purpose: Provides business logic for creating, retrieving, and updating user reports.</p>
Methods/Operations		
01	GetAllReports	<p>Visibility: public</p> <p>Return: Task<ActionResult<IEnumerable>></p> <p>Purpose: Retrieves all reports in the system (Admin only).</p> <p>Parameters:- ct: CancellationToken — optional.</p>

02	GetReportById	<p>Visibility: public</p> <p>Return: Task<ActionResult> Purpose: Retrieves a single report by its ID.Parameters:- reportId: int — ID of the report.- ct: CancellationToken — optional.</p>
03	GetReportsByUserId	<p>Visibility: public</p> <p>Return: Task<ActionResult<IEnumerable>> Purpose: Retrieves all reports created by a specific user.Parameters:- userReportId: int — ID of the reporting user.- ct: CancellationToken — optional.</p>
04	CreateReport	<p>Visibility: public</p> <p>Return: TaskPurpose: Creates a new report for a given user and content item.Parameters:- userReportId: int — ID of the user submitting the report.- contentId: int — ID of the reported content.- dto: ReportCreateDTO — report details.- ct: CancellationToken — optional.</p>
05	UpdateReport	<p>Visibility: public</p> <p>Return: TaskPurpose: Updates a report's status or resolution.Parameters:- reportId: int — ID of the report to update.- dto: ReportUpdateDTO — updated report information.- ct: CancellationToken — optional.</p>

4.3.22 SendMailOtpController

No	Name	Description
Attributes		

01	_otpService	<p>Visibility: private readonly</p> <p>Type: IOtpService</p> <p>Purpose: Provides logic for generating, sending, and validating OTP codes for user email verification and password recovery.</p>
Methods/Operations		
01	SendOtp	<p>Visibility: public</p> <p>Return: TaskPurpose: Sends an OTP code to the user's email for registration or password reset.</p> <p>Parameters:- email: string — target email to receive OTP.- purpose: string — purpose of OTP, default "register".- ct: CancellationToken — optional.</p>
02	CheckOtp	<p>Visibility: public</p> <p>Return: TaskPurpose: Validates an OTP submitted by the user.</p> <p>Parameters:- request: OtpRequest — contains Email and Otp.- ct: CancellationToken — optional.</p>

4.3.23 UserPreferenceController

No	Name	Description
Attributes		
01	_userPreferenceService	<p>Visibility: private readonly</p> <p>Type: IUserPreferenceService</p> <p>Purpose: Provides business logic for retrieving, creating, updating, deleting, and batch-upserting user preference data.</p>
Methods/Operations		
01	GetAllByUser	<p>Visibility: public</p> <p>Return: Task<ActionResult<IEnumerable>></p> <p>Purpose: Retrieves all preference settings of a specified user.</p> <p>Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>
02	Create	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Creates a user preference entry for a specific attribute.</p> <p>Parameters:- userId: int — ID of the user.- attributeId: int — ID of the attribute.- req: UserPreferenceUpsertRequest — preference data to create.- ct: CancellationToken — optional.</p>
03	Update	<p>Visibility: public</p> <p>Return: Task<ActionResult></p> <p>Purpose: Updates a user's existing preference for a given attribute.</p> <p>Parameters:- userId: int — ID of the user.- attributeId: int — ID of the attribute.- req: UserPreferenceUpsertRequest — new preference values.- ct: CancellationToken — optional.</p>

No	Name	Description
Attributes		
01	_userPreferenceService	<p>Visibility: private readonly</p> <p>Type: IUserPreferenceService</p> <p>Purpose: Provides business logic for retrieving, creating, updating, deleting, and batch-upserting user preference data.</p>
Methods/Operations		
04	DeleteUserPreferences	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Deletes all preferences belonging to a specific user.</p> <p>Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>
05	UpsertBatch	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Updates or inserts multiple user preference entries in a single batch operation.</p> <p>Parameters:- userId: int — ID of the user.- request: UserPreferenceBatchUpsertRequest — list of preferences to upsert.- ct: CancellationToken — optional.</p>

4.3.24 UserController

No	Name	Description
Attributes		
01	_userService	<p>Visibility: private readonly</p> <p>Type: IUserService</p>

		Purpose: Provides business logic for retrieving, creating, updating, deleting users, and managing profile & password operations.
Methods/Operations		
01	GetUsers	<p>Visibility: public</p> <p>Return: Task<ActionResult<PagedResult>> Purpose: Retrieves paginated users with optional search, role, status, and deleted filters.</p> <p>Parameters:- search: string? — search keyword.- roleId: int? — filter by role.- statusId: int? — filter by status.- page: int — page index.- pageSize: int — number of items per page.- includeDeleted: bool — include soft-deleted users.- ct: CancellationToken — optional.</p>
02	GetUser	<p>Visibility: public</p> <p>Return: Task<ActionResult> Purpose: Retrieves a single user by ID.</p> <p>Parameters:- userId: int — ID of the user.- ct: CancellationToken — optional.</p>
03	Register	<p>Visibility: public</p> <p>Return: Task<ActionResult> Purpose: Registers a new user account.</p> <p>Parameters:- req: UserCreateRequest — user registration data.- ct: CancellationToken — optional.</p>
04	UpdateUser	<p>Visibility: public</p> <p>Return: Task<ActionResult> Purpose: Updates user profile information.</p> <p>Parameters:- userId: int — user ID.- req: UserUpdateRequest — updated user data.- ct: CancellationToken — optional.</p>

05	SoftDelete	<p>Visibility: public</p> <p>Return: TaskPurpose: Soft-deletes a user account (Admin only).Parameters:- userId: int - ct: CancellationToken — optional.</p>
06	CompleteProfile	<p>Visibility: public</p> <p>Return: TaskPurpose: Marks a user's profile as completed.Parameters:- id: int — user ID.- ct: CancellationToken — optional.</p>
07	ResetPassword	<p>Visibility: public</p> <p>Return: TaskPurpose: Resets a user's password using a valid request payload.Parameters:- request: ResetPasswordRequest — contains email and new password.- ct: CancellationToken — optional.</p>

4.3.25 AppointmentController

No	Name	Description
Attributes		
01	_appointmentService	<p>Visibility: private readonly</p> <p>Type: IAppointmentService</p> <p>Purpose: Provides business logic for creating, retrieving, updating, responding to pet appointments, handling</p>

		counter-offers, cancellations, check-in, completion, and managing appointment locations.
Methods/Operations		
01	CreateAppointment	<p>Visibility: public Return: Task<IActionResult> Purpose: Creates a new pet appointment between two matched pets. Parameters: request: CreateAppointmentRequest, ct: CancellationToken</p>
02	GetAppointmentById	<p>Visibility: public Return: Task<IActionResult> Purpose: Retrieves appointment details by appointment ID. Parameters: appointmentId: int, ct: CancellationToken</p>
03	GetAppointmentsByMatch	<p>Visibility: public Return: Task<IActionResult> Purpose: Retrieves all appointments related to a specific match. Parameters: matchId: int, ct: CancellationToken</p>
04	GetMyAppointments	<p>Visibility: public Return: Task<IActionResult> Purpose: Retrieves all appointments that the current user participates in. Parameters: ct: CancellationToken</p>
05	RespondToAppointment	<p>Visibility: public Return: Task<IActionResult> Purpose: Accepts or declines an appointment invitation. Parameters: appointmentId: int, request: RespondAppointmentRequest, ct: CancellationToken</p>
06	CounterOffer	<p>Visibility: public Return: Task<IActionResult> Purpose: Submits a counter-offer with new appointment details. Parameters: appointmentId: int, request: CounterOfferRequest, ct: CancellationToken</p>

07	CancelAppointment	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Cancels an existing appointment and records cancellation information.</p> <p>Parameters: appointmentId: int, request: CancelAppointmentRequest, ct: CancellationToken</p>
08	CheckIn	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Performs GPS-based check-in at the appointment location.</p> <p>Parameters: appointmentId: int, request: CheckInRequest, ct: CancellationToken</p>
09	CompleteAppointment	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Marks an appointment as completed.</p> <p>Parameters: appointmentId: int, ct: CancellationToken</p>
10	GetMyRecentLocations	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Retrieves recent appointment locations of the current user.</p> <p>Parameters: limit: int, ct: CancellationToken</p>
11	CreateLocation	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Creates a new pet-friendly location for appointments.</p> <p>Parameters: request: CreateLocationRequest, ct: CancellationToken</p>
12	ValidatePreconditions	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Validates required conditions before allowing appointment creation.</p> <p>Parameters: matchId: int, inviterPetId: int, inviteePetId: int, ct: CancellationToken</p>

4.3.26 EventController

No	Name	Description
Attributes		
01	_eventService	<p>Visibility: private readonly Type: IEventService</p> <p>Purpose: Provides business logic for managing online events, including event creation, updating, cancellation, submission handling, voting, and leaderboard generation.</p>
02	_photoStorage	<p>Visibility: private readonly Type: IPhotoStorage</p> <p>Purpose: Handles media upload and storage for event submissions (images and videos).</p>
Methods/Operations		
01	GetAllEvents	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Retrieves all events regardless of status (Admin only). Parameters: ct: CancellationToken</p>
02	CreateEvent	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Creates a new online event (Admin only). Parameters: request: CreateEventRequest, ct: CancellationToken</p>
03	UpdateEvent	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Updates an existing event's information (Admin only). Parameters: eventId: int, request: UpdateEventRequest, ct: CancellationToken</p>

04	CancelEvent	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Cancels an event and records cancellation reason if provided (Admin only).</p> <p>Parameters: eventId: int, request: CancelEventRequest?, ct: CancellationToken</p>
05	GetActiveEvents	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Retrieves all currently active events for public users.</p> <p>Parameters: ct: CancellationToken</p>
06	GetEventById	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Retrieves detailed information of a specific event, including user-related context if logged in.</p> <p>Parameters: eventId: int, ct: CancellationToken</p>
07	SubmitEntry	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Submits a photo or video entry to an event (User only).</p> <p>Parameters: eventId: int, request: SubmitEntryRequest, ct: CancellationToken</p>
08	Vote	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Casts a vote for a specific event submission (User only).</p> <p>Parameters: submissionId: int, ct: CancellationToken</p>
09	Unvote	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Removes a previously cast vote from a submission (User only).</p> <p>Parameters: submissionId: int, ct: CancellationToken</p>

10	GetLeaderboard	<p>Visibility: public Return: Task<IActionResult> Purpose: Retrieves the leaderboard of an event based on vote counts and rankings. Parameters: eventId: int, ct: CancellationToken</p>
11	UploadMedia	<p>Visibility: public Return: Task<IActionResult> Purpose: Uploads image or video files for event submissions with file type and size validation (User only). Parameters: file: IFormFile, ct: CancellationToken</p>

4.3.27 PolicyController

No	Name	Description
Attributes		
01	_policyService	<p>Visibility: private readonly Type: IPolicyService Purpose: Provides business logic for managing policies and versions (CRUD, publish workflow, statistics) and handling user policy viewing and acceptance operations.</p>
Methods/Operations		
01	GetAllPolicies	<p>Visibility: public Return: Task<ActionResult<List<PolicyResponse>>> Purpose: Retrieves all policies (Admin only), including all statuses. Parameters: ct: CancellationToken</p>
02	GetPolicyById	<p>Visibility: public Return: Task<ActionResult<PolicyResponse>> Purpose: Retrieves policy details by policy ID (Admin only). Parameters: policyId: int, ct: CancellationToken</p>

03	CreatePolicy	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyResponse>></p> <p>Purpose: Creates a new policy (Admin only).</p> <p>Parameters: request: CreatePolicyRequest, ct: CancellationToken</p>
04	UpdatePolicy	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyResponse>></p> <p>Purpose: Updates an existing policy by ID (Admin only).</p> <p>Parameters: policyId: int, request: UpdatePolicyRequest, ct: CancellationToken</p>
05	DeletePolicy	<p>Visibility: public</p> <p>Return: Task<ActionResult></p> <p>Purpose: Soft-deletes a policy by ID (Admin only).</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
06	GetVersionsByPolicyId	<p>Visibility: public</p> <p>Return: Task<ActionResult<List<PolicyVersionResponse>>></p> <p>Purpose: Retrieves all versions of a policy by policy ID (Admin only).</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
07	GetVersionById	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyVersionResponse>></p> <p>Purpose: Retrieves policy version details by version ID (Admin only).</p> <p>Parameters: versionId: int, ct: CancellationToken</p>
08	CreateVersion	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyVersionResponse>></p> <p>Purpose: Creates a new policy version in DRAFT status (Admin only).</p> <p>Parameters: policyId: int, request: CreatePolicyVersionRequest, ct: CancellationToken</p>

09	UpdateVersion	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyVersionResponse>></p> <p>Purpose: Updates a policy version (Admin only, only when status is DRAFT).</p> <p>Parameters: versionId: int, request: UpdatePolicyVersionRequest, ct: CancellationToken</p>
10	PublishVersion	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyVersionResponse>></p> <p>Purpose: Publishes a policy version (Admin only), transitioning status from DRAFT to ACTIVE and deactivating previous active version if applicable.</p> <p>Parameters: versionId: int, ct: CancellationToken</p>
11	DeleteVersion	<p>Visibility: public</p> <p>Return: Task<ActionResult></p> <p>Purpose: Deletes a policy version (Admin only, only when status is DRAFT).</p> <p>Parameters: versionId: int, ct: CancellationToken</p>
12	GetAcceptStats	<p>Visibility: public</p> <p>Return: Task<ActionResult<List<PolicyAcceptStatsResponse>>></p> <p>Purpose: Retrieves acceptance statistics for policies (Admin only).</p> <p>Parameters: ct: CancellationToken</p>
13	GetAllActivePolicies	<p>Visibility: public</p> <p>Return: Task<ActionResult<List<PendingPolicyResponse>>></p> <p>Purpose: Retrieves all active policies for public viewing (AllowAnonymous).</p> <p>Parameters: ct: CancellationToken</p>
14	GetActivePolicyContent	<p>Visibility: public</p> <p>Return: Task<ActionResult<PendingPolicyResponse>></p> <p>Purpose: Retrieves the active policy content by policy code (AllowAnonymous).</p> <p>Parameters: policyCode: string, ct: CancellationToken</p>

15	CheckPolicyStatus	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyStatusResponse>></p> <p>Purpose: Checks the policy acceptance status of the current logged-in user (User/Admin).</p> <p>Parameters: ct: CancellationToken</p>
16	GetPendingPolicies	<p>Visibility: public</p> <p>Return: Task<ActionResult<List<PendingPolicyResponse>>></p> <p>Purpose: Retrieves policies that the current user needs to accept (User/Admin).</p> <p>Parameters: ct: CancellationToken</p>
17	AcceptPolicy	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyStatusResponse>></p> <p>Purpose: Accepts a single policy version for the current user (User/Admin).</p> <p>Parameters: request: AcceptPolicyRequest, ct: CancellationToken</p>
18	AcceptMultiplePolicies	<p>Visibility: public</p> <p>Return: Task<ActionResult<PolicyStatusResponse>></p> <p>Purpose: Accepts multiple policy versions at once for the current user (User/Admin).</p> <p>Parameters: request: AcceptMultiplePoliciesRequest, ct: CancellationToken</p>
19	GetAcceptHistory	<p>Visibility: public</p> <p>Return: Task<ActionResult<List<UserAcceptHistoryResponse>>></p> <p>Purpose: Retrieves the acceptance history of the current user (User/Admin).</p> <p>Parameters: ct: CancellationToken</p>
20	GetCurrentUserId	<p>Visibility: private</p> <p>Return: int</p> <p>Purpose: Extracts current user ID from JWT claims and throws UnauthorizedAccessException if missing.</p> <p>Parameters: None</p>

4.3.28 BadWordController

No	Name	Description
Attributes		
01	_badWordManagementService	<p>Visibility: private readonly Type: IBadWordManagementService</p> <p>Purpose: Provides business logic for managing prohibited words, including retrieval, creation, updating, deletion, and cache reloading for the bad word filtering system.</p>
Methods/Operations		
01	GetAllBadWords	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Retrieves the list of all prohibited words in the system (Admin only).</p> <p>Parameters: ct: CancellationToken</p>
02	GetBadWordById	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Retrieves detailed information of a prohibited word by its ID (Admin only).</p> <p>Parameters: id: int, ct: CancellationToken</p>
03	CreateBadWord	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Creates a new prohibited word entry for content moderation (Admin only).</p> <p>Parameters: request: CreateBadWordRequest, ct: CancellationToken</p>
04	UpdateBadWord	<p>Visibility: public Return: Task<IActionResult></p> <p>Purpose: Updates an existing prohibited word by ID (Admin only).</p> <p>Parameters: id: int, request: UpdateBadWordRequest, ct: CancellationToken</p>

05	DeleteBadWord	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Deletes a prohibited word from the system (Admin only).</p> <p>Parameters: id: int, ct: CancellationToken</p>
06	ReloadCache	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Reloads the in-memory cache of prohibited words to immediately apply updates to the filtering system (Admin only).</p> <p>Parameters: ct: CancellationToken</p>

4.3.29 PetImageAnalysisController

No	Name	Description
Attributes		
01	_analysisService	<p>Visibility: private readonly</p> <p>Type: IPetImageAnalysisService</p> <p>Purpose: Provides business logic for analyzing pet images and returning extracted pet attributes from AI/image analysis processing.</p>
Methods/Operations		
01	AnalyzeImage	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Analyzes a single pet image and returns extracted attributes; validates file type and size before processing.</p> <p>Parameters: image: IFormFile</p>
02	AnalyzeMultipleImages	<p>Visibility: public</p> <p>Return: Task<IActionResult></p> <p>Purpose: Analyzes multiple pet images (batch) and returns aggregated analysis results; validates file types and size for each image before processing.</p> <p>Parameters: images: List<IFormFile></p>

4.3.30 ExpertController

No	Name	Description
Attributes		
01	_expertConfirmationService	<p>Visibility: private readonly Type: IExpertConfirmationService</p> <p>Purpose: Provides business logic for expert confirmation workflows, including retrieving requests, creating confirmation requests, updating confirmation status, and retrieving expert chat sessions.</p>
Methods/Operations		
01	GetAllExpertConfirmations	<p>Visibility: public Return: Task<ActionResult<List<ExpertConfirmationDTO>>> Purpose: Retrieves all expert confirmation requests (Admin/Expert). Parameters: ct: CancellationToken</p>
02	GetExpertConfirmation	<p>Visibility: public Return: Task<ActionResult<ExpertConfirmationDTO>> Purpose: Retrieves a specific expert confirmation request by composite identifiers (expertId, userId, chatId) (Admin/Expert). Parameters: expertId: int, userId: int, chatId: int, ct: CancellationToken</p>
03	GetUserExpertConfirmations	<p>Visibility: public Return: Task<ActionResult<List<ExpertConfirmationDTO>>> Purpose: Retrieves expert confirmation requests related to a specific user (User/Expert/Admin). Parameters: userId: int, ct: CancellationToken</p>

04	CreateExpertConfirmation	<p>Visibility: public Return: Task<ActionResult<ExpertConfirmationResponseDTO>> Purpose: Creates a new expert confirmation request for a given user and chat; enforces business rules such as daily request limit (returns 429 when exceeded). Parameters: userId: int, chatId: int, dto: ExpertConfirmationCreateDTO, ct: CancellationToken</p>
05	UpdateExpertConfirmation	<p>Visibility: public Return: Task<ActionResult<ExpertConfirmationResponseDTO>> Purpose: Updates an expert confirmation request (e.g., approve/decline) by composite identifiers (Expert/Admin). Parameters: expertId: int, userId: int, chatId: int, dto: ExpertConfirmationUpdateDto, ct: CancellationToken</p>
06	GetUserExpertChats	<p>Visibility: public Return: Task<ActionResult> Purpose: Retrieves expert chat sessions for a given user (User/Expert/Admin). Parameters: userId: int, ct: CancellationToken</p>

4.4 Service

4.4.1 AddressService

No	Name	Description
Attributes		
01	_addressRepository	<p>Visibility: private Type: IAddressRepository Purpose: Repository responsible for managing address data operations.</p>
02	_context	Visibility: private

		Type: PawnderDatabaseContext Purpose: Entity Framework database context used to access user and address entities.
03	_httpClient	Visibility: private Type: HttpClient Purpose: Used to send HTTP requests to the LocationIQ reverse geocoding API.
04	_configuration	Visibility: private Type: IConfiguration Purpose: Provides access to application configuration values such as API keys.
05	_cache	Visibility: private Type: IMemoryCache Purpose: Stores temporary data for rate limiting user requests.
06	MAX_REQUESTS_PER_SECOND	Visibility: private const Type: int Purpose: Defines the maximum number of requests allowed per user per second.
07	RATE_LIMIT_WINDOW	Visibility: private static readonly Type: TimeSpan Purpose: Defines the time window used for rate limiting (1 second).
Methods/Operations		

01	CreateAddressForUserAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Creates a new address for a user based on GPS coordinates, performs reverse geocoding, and links the address to the user.</p> <p>Parameters: userId, locationDto, cancellationToken</p>
02	UpdateAddressAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Updates an existing address using new latitude and longitude values and refreshes geocoded address information.</p> <p>Parameters: addressId, locationDto, cancellationToken</p>
03	UpdateAddressManualAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Allows manual update of address fields such as city, district, and ward without calling the geocoding API.</p> <p>Parameters: addressId, manualAddressDto, cancellationToken</p>
04	GetAddressByIdAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Retrieves address information by address identifier and formats timestamps for display.</p> <p>Parameters: addressId, cancellationToken</p>
05	GeocodeAsync	<p>Visibility: private</p> <p>Return: Task<(string?, string?, string?, string?)></p>

		<p>Purpose: Calls the LocationIQ reverse geocoding API to convert coordinates into human-readable address components.</p> <p>Parameters: latitude, longitude, cancellationToken</p>
06	CleanVietnameseAddress	<p>Visibility: private</p> <p>Return: string?</p> <p>Purpose: Removes Vietnamese administrative prefixes from address strings to normalize data.</p> <p>Parameters: rawAddress, prefixes</p>
07	ValidateCoordinates	<p>Visibility: private</p> <p>Return: void</p> <p>Purpose: Validates that latitude and longitude values fall within valid geographic ranges.</p> <p>Parameters: latitude, longitude</p>
08	CheckRateLimit	<p>Visibility: private</p> <p>Return: bool</p> <p>Purpose: Enforces per-user request rate limits using in-memory cache tracking.</p> <p>Parameters: userId</p>

4.4.2 AdminService

No	Name	Description
Attributes		

01	<code>_expertConfirmationRepository</code>	<p>Visibility: private</p> <p>Type: IExpertConfirmationRepository</p> <p>Purpose: Repository used to access and manage expert confirmation records.</p>
02	<code>_userRepository</code>	<p>Visibility: private</p> <p>Type: IUserRepository</p> <p>Purpose: Repository used to access and manage user data.</p>
03	<code>_context</code>	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to query and update entities such as ChatAi, UserBanHistory and UserStatus.</p>
04	<code>_passwordService</code>	<p>Visibility: private</p> <p>Type: PasswordService</p> <p>Purpose: Service used to hash user passwords before storing them in the database.</p>
Methods/Operations		
01	<code>ReassignExpertConfirmationAsync</code>	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Reassigns an existing expert confirmation from one expert to another after validating user, chat session and experts, and only when the current status is “pending”.</p> <p>Parameters: ReassignExpertConfirmationRequest req, CancellationToken ct = default</p>
02	<code>BanUserAsync</code>	Visibility: public

		<p>Return: Task<object></p> <p>Purpose: Creates a new ban record for a user (temporary or permanent), deactivates expired bans, and sets the user status to “Bị khóa”.</p> <p>Parameters: int userId, BanUserRequest req, CancellationToken ct = default</p>
03	UnbanUserAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Lifts all active bans for a user, updates ban history with unban reason and restores the user status based on payment history (VIP or normal).</p> <p>Parameters: int userId, UnbanUserRequest? req, CancellationToken ct = default</p>
04	GetUserBansAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves the list of all ban histories for a given user ordered by ban start time in descending order.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
05	UpdateUserByAdminAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Allows admin to update user flags such as soft delete state and user status id, then saves the changes.</p> <p>Parameters: int userId, AdUserUpdateRequest request, CancellationToken ct = default</p>

06	RegisterUserByAdminAsync	<p>Visibility: public</p> <p>Return: Task<UserResponse></p> <p>Purpose: Registers a new user account by admin, including email uniqueness check, password hashing and setting default user status when not provided.</p> <p>Parameters: AdUserCreateRequest req, CancellationToken ct = default</p>
----	--------------------------	---

4.4.3 AttributeOptionService

No	Name	Description
Attributes		
01	_optionRepository	<p>Visibility: private</p> <p>Type: IAttributeOptionRepository</p> <p>Purpose: Repository used to access and manage attribute option data.</p>
02	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to validate and query attribute entities.</p>
Methods/Operations		
01	GetAllOptionsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<OptionResponse>></p> <p>Purpose: Retrieves all attribute options that are not deleted from the system.</p>

		Parameters: CancellationToken ct = default
02	GetOptionsByAttributeIdAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all options belonging to a specific attribute after validating that the attribute exists and is active.</p> <p>Parameters: int attributeId, CancellationToken ct = default</p>
03	CreateOptionAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Creates a new option for a given attribute after validating the attribute and option name.</p> <p>Parameters: int attributeId, string optionName, CancellationToken ct = default</p>
04	UpdateOptionAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Updates the name of an existing attribute option after validating that it exists and is not deleted.</p> <p>Parameters: int optionId, string optionName, CancellationToken ct = default</p>
05	DeleteOptionAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Performs a soft delete on an attribute option by marking it as deleted and updating the timestamp.</p> <p>Parameters: int optionId, CancellationToken ct = default</p>

4.4.4 AttributeService

No	Name	Description
Attributes		
01	_attributeRepository	<p>Visibility: private</p> <p>Type: IAttributeRepository</p> <p>Purpose: Repository used to access and manage attribute entities in the database.</p>
Methods/Operations		
01	GetAttributesAsync	<p>Visibility: public</p> <p>Return: Task<PagedResult<AttributeResponse>></p> <p>Purpose: Retrieves a paginated list of attributes, optionally filtered by search text and including deleted records if requested.</p> <p>Parameters: string? search, int page, int pageSize, bool includeDeleted, CancellationToken ct = default</p>
02	GetAttributeByIdA sync	<p>Visibility: public</p> <p>Return: Task<AttributeResponse?></p> <p>Purpose: Retrieves a single attribute by its identifier and maps it to an AttributeResponse object; returns null if not found.</p> <p>Parameters: int id, CancellationToken ct = default</p>
03	CreateAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<AttributeResponse></p>

		<p>Purpose: Creates a new attribute after validating that the attribute name does not already exist, then returns the created attribute as AttributeResponse.</p> <p>Parameters: AttributeCreateRequest request, CancellationToken ct = default</p>
04	UpdateAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Updates an existing attribute's name, type, unit and soft delete flag after checking for duplicate names.</p> <p>Parameters: int id, AttributeUpdateRequest request, CancellationToken ct = default</p>
05	DeleteAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Deletes an attribute either by hard delete from the database or by soft delete (marking IsDeleted = true) based on the hard flag.</p> <p>Parameters: int id, bool hard = false, CancellationToken ct = default</p>

4.4.5 AttributeService

No	Name	Description
Attributes		
01	_attributeRepository	<p>Visibility: private</p> <p>Type: IAttributeRepository</p>

		Purpose: Repository used to access and manage attribute entities in the database.
Methods/Operations		
01	GetAttributesAsync	<p>Visibility: public</p> <p>Return: Task<PagedResult<AttributeResponse>></p> <p>Purpose: Retrieves a paginated list of attributes, optionally filtered by search text and including deleted records if requested.</p> <p>Parameters: string? search, int page, int pageSize, bool includeDeleted, CancellationToken ct = default</p>
02	GetAttributeByIdA sync	<p>Visibility: public</p> <p>Return: Task<AttributeResponse?></p> <p>Purpose: Retrieves a single attribute by its identifier and maps it to an AttributeResponse object; returns null if not found.</p> <p>Parameters: int id, CancellationToken ct = default</p>
03	CreateAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<AttributeResponse></p> <p>Purpose: Creates a new attribute after validating that the attribute name does not already exist, then returns the created attribute as AttributeResponse.</p> <p>Parameters: AttributeCreateRequest request, CancellationToken ct = default</p>
04	UpdateAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<bool></p>

		<p>Purpose: Updates an existing attribute's name, type, unit and soft delete flag after checking for duplicate names.</p> <p>Parameters: int id, AttributeUpdateRequest request, CancellationToken ct = default</p>
05	DeleteAttributeAs ync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Deletes an attribute either by hard delete from the database or by soft delete (marking IsDeleted = true) based on the hard flag.</p> <p>Parameters: int id, bool hard = false, CancellationToken ct = default</p>

4.4.6 AuthService

No	Name	Description
Attributes		
01	_userRepository	<p>Visibility: private</p> <p>Type: IUserRepository</p> <p>Purpose: Repository used to access and manage user data (find by email, id, update user, etc.).</p>
02	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to query ban histories, payment histories and user statuses.</p>
03	_passwordService	<p>Visibility: private</p> <p>Type: PasswordService</p>

		Purpose: Service used to verify passwords, hash new passwords and detect legacy hashes.
04	_tokenService	<p>Visibility: private</p> <p>Type: TokenService</p> <p>Purpose: Service used to generate and validate JWT access tokens and refresh tokens.</p>
Methods/Operations		
01	LoginAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Authenticates a user by email and password, validates role based on platform (user/admin), checks ban status, upgrades legacy password hashes and generates access and refresh tokens for successful login.</p> <p>Parameters: LoginRequest request, CancellationToken ct = default</p>
02	RefreshTokenAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Validates an existing refresh token, verifies that it belongs to a valid user with an allowed role, checks ban status and returns a new pair of access and refresh tokens.</p> <p>Parameters: RefreshTokenRequest request, CancellationToken ct = default</p>
03	LogoutAsync	<p>Visibility: public</p> <p>Return: Task<bool></p>

		<p>Purpose: Logs out a user logically by updating the user record (for example, to track last update or future invalidation logic).</p> <p>Parameters: int userId, CancellationToken ct = default</p>
--	--	---

4.4.7 BlockService

No	Name	Description
Attributes		
01	_blockRepository	<p>Visibility: private</p> <p>Type: IBlockRepository</p> <p>Purpose: Repository used to access and manage block records between users.</p>
02	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to validate users and update related chat data.</p>
Methods/Operations		
01	GetBlockedUsersA sync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves the list of users that have been blocked by a specific user.</p> <p>Parameters: int fromUserId, CancellationToken ct = default</p>
02	CreateBlockAsync	<p>Visibility: public</p> <p>Return: Task<object></p>

		<p>Purpose: Creates a new block relationship between two users after validating that both users exist, are not the same user and are not already blocked; also soft-deletes any existing chat between them.</p> <p>Parameters: int fromUserId, int toUserId, CancellationToken ct = default</p>
03	DeleteBlockAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Removes an existing block relationship between two users if it exists.</p> <p>Parameters: int fromUserId, int toUserId, CancellationToken ct = default</p>

4.4.8 ChatAIService

No	Name	Description
Attributes		
01	_geminiService	<p>Visibility: private</p> <p>Type: IGeminiAIService</p> <p>Purpose: Service dùng để tạo phiên chat AI, gửi câu hỏi và lấy lịch sử hội thoại từ mô hình Gemini.</p>
02	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context dùng để truy vấn và cập nhật dữ liệu liên quan đến ChatAi và Users.</p>
03	_dailyLimitService	Visibility: private

		<p>Type: DailyLimitService</p> <p>Purpose: Service dùng để theo dõi và giới hạn số lượng token AI mà user sử dụng mỗi ngày (freemium quota).</p>
Methods/Operations		
01	GetAllChatsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Lấy danh sách tất cả cuộc trò chuyện AI của một người dùng, sắp xếp theo thời gian cập nhật gần nhất, kèm số lượng tin nhắn và câu hỏi gần nhất.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
02	CreateChatAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Tạo một phiên chat AI mới cho user thông qua IGeminiAIService, với tiêu đề mặc định “New Chat” nếu không truyền title.</p> <p>Parameters: int userId, string? title, CancellationToken ct = default</p>
03	UpdateChatTitleAs ync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Cập nhật tiêu đề của một cuộc trò chuyện AI thuộc về user, sau khi kiểm tra tồn tại và không bị xóa mềm.</p> <p>Parameters: int chatAild, int userId, string title, CancellationToken ct = default</p>
04	DeleteChatAsync	<p>Visibility: public</p>

		<p>Return: Task<bool></p> <p>Purpose: Đánh dấu xóa mềm (IsDeleted = true) một cuộc trò chuyện AI; cho phép admin/expert truyền userId = 0 để xóa bất kỳ chat nào.</p> <p>Parameters: int chatAild, int userId, CancellationToken ct = default</p>
05	GetChatHistoryAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Lấy chi tiết lịch sử hội thoại AI (danh sách question/answer) cho một chat cụ thể; cho phép user chỉ xem chat của mình, còn admin/expert có thể xem mọi chat (userId = 0).</p> <p>Parameters: int chatAild, int userId, CancellationToken ct = default</p>
06	SendMessageAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Gửi câu hỏi đến phiên chat AI sau khi kiểm tra user, xác định trạng thái VIP, ước lượng token sẽ dùng, kiểm tra quota ngày, gọi IGeminiAIService, trừ token thực tế và trả về câu trả lời kèm thông tin usage/quota.</p> <p>Parameters: int chatAild, int userId, string question, CancellationToken ct = default</p>
07	GetTokenUsageAsync	<p>Visibility: public</p> <p>Return: Task<object></p>

		<p>Purpose: Lấy thông tin sử dụng token trong ngày của user, bao gồm tổng quota, số token đã dùng và số token còn lại, phân biệt giữa user thường và VIP.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
08	EstimateTokens	<p>Visibility: private</p> <p>Return: int</p> <p>Purpose: Ước lượng số token sẽ tiêu thụ dựa trên độ dài text câu hỏi, giả định output dài hơn input khoảng 3 lần, dùng để kiểm tra quota trước khi gọi AI.</p> <p>Parameters: string text</p>

4.4.9 ChatExpertContentService

No	Name	Description
Attributes		
01	_contentRepository	<p>Visibility: private</p> <p>Type: IChatExpertContentRepository</p> <p>Purpose: Repository used to access and manage chat expert content (messages) in the database.</p>
02	_chatExpertRepository	<p>Visibility: private</p> <p>Type: IChatExpertRepository</p> <p>Purpose: Repository used to validate and access chat expert sessions.</p>
03	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p>

		Purpose: Entity Framework database context used to validate chat, expert confirmation and related entities.
04	_hubContext	<p>Visibility: private</p> <p>Type: IHubContext<ChatHub></p> <p>Purpose: SignalR hub context used to send real-time chat messages and notifications.</p>
05	_dailyLimitService	<p>Visibility: private</p> <p>Type: IDailyLimitService</p> <p>Purpose: Service used to check and record daily limits for user actions such as expert chat messaging.</p>
Methods/Operations		
01	GetChatMessages Async	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all messages belonging to a specific expert chat after validating that the chat exists.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default</p>
02	SendMessageAsyn c	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Sends a message in an expert chat after validating chat participants, checking daily chat limits for users, optionally validating expert confirmation, saving the message, recording usage limits and sending real-time notifications via SignalR.</p> <p>Parameters: int chatExpertId, int fromId, string message, int? expertId, int? userId, int? chatAiid, CancellationToken ct = default</p>

4.4.10 ChatExpertService

No	Name	Description
Attributes		
01	_chatExpertRepository	<p>Visibility: private Type: IChatExpertRepository</p> <p>Purpose: Repository used to access and manage expert chat sessions between users and experts.</p>
02	_context	<p>Visibility: private Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to validate users/experts and interact with related entities.</p>
Methods/Operations		
01	GetChatsByUserId Async	<p>Visibility: public Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions associated with a specific user after validating that the user exists.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
02	GetChatsByExpertId dAsync	<p>Visibility: public Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions associated with a specific expert after validating that the expert exists, and logs diagnostic information to the console.</p> <p>Parameters: int expertId, CancellationToken ct = default</p>

03	CreateChatAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Creates a new expert chat session between an expert and a user after validating both exist and ensuring a chat does not already exist; if a chat exists, returns the existing chat information instead of creating a new one.</p> <p>Parameters: int expertId, int userId, CancellationToken ct = default</p>
----	-----------------	---

4.4.11 ChatHubService

No	Name	Description
Attributes		
01	UserConnections	<p>Visibility: public static readonly</p> <p>Type: ConcurrentDictionary<int, HashSet<string>></p> <p>Purpose: Stores the mapping between user ids and their active SignalR connection ids to support multi-device connections and targeted notifications.</p>
02	OnlineUsers	<p>Visibility: private static readonly</p> <p>Type: ConcurrentDictionary<int, DateTime></p> <p>Purpose: Tracks which users are currently online by user id and the timestamp of their last registration or activity.</p>
Methods/Operations		
01	OnConnectedAsyn c	<p>Visibility: public override</p> <p>Return: Task</p>

		<p>Purpose: Called when a client connects to the hub; logs the new connection id for debugging purposes.</p> <p>Parameters: none</p>
02	OnDisconnectedAs ync	<p>Visibility: public override</p> <p>Return: Task</p> <p>Purpose: Called when a client disconnects; removes the connection id from all tracked users, updates online status and notifies others when a user goes offline.</p> <p>Parameters: Exception? exception</p>
03	RegisterUser	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Registers a user with the current connection by adding the connection id to UserConnections, marking the user as online and notifying other clients that the user is online.</p> <p>Parameters: int userId</p>
04	JoinChat	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Adds the current connection to a match chat group and notifies other users in the group that this user has joined the chat.</p> <p>Parameters: int matchId, int userId</p>
05	LeaveChat	<p>Visibility: public</p> <p>Return: Task</p>

		<p>Purpose: Removes the current connection from a match chat group and notifies other users in the group that this user has left the chat.</p> <p>Parameters: int matchId, int userId</p>
06	JoinExpertChat	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Adds the current connection to an expert chat group for real-time expert–user communication.</p> <p>Parameters: int chatExpertId, int userId</p>
07	LeaveExpertChat	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Removes the current connection from an expert chat group when the user leaves the expert chat screen.</p> <p>Parameters: int chatExpertId, int userId</p>
08	SendExpertMessage	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Sends a message to all connections in an expert chat group and broadcasts it via the “ReceiveExpertMessage” event for both user and expert clients.</p> <p>Parameters: int chatExpertId, int fromId, string message</p>
09	SendMessage	<p>Visibility: public</p> <p>Return: Task</p>

		<p>Purpose: Sends a message in a match chat to all connections in the corresponding match group and broadcasts it via the “ReceiveMessage” event, including optional pet information.</p> <p>Parameters: int matchId, int fromUserId, string message, int? fromPetId = null</p>
10	Typing	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Notifies other users in a match chat group that a specific user is typing or has stopped typing by broadcasting a “UserTyping” event.</p> <p>Parameters: int matchId, int userId, bool isTyping</p>
11	MarkAsRead	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Notifies other users in a match chat group that messages have been read by a user through the “MessagesRead” event.</p> <p>Parameters: int matchId, int userId</p>
12	IsUserOnline	<p>Visibility: public</p> <p>Return: bool</p> <p>Purpose: Checks whether a user is currently online based on the OnlineUsers dictionary.</p> <p>Parameters: int userId</p>

13	GetOnlineUsers	<p>Visibility: public</p> <p>Return: Task<List<int>></p> <p>Purpose: Returns the list of all user ids that are currently tracked as online.</p> <p>Parameters: none</p>
Static methods		
14	SendNewMessageBadge	<p>Visibility: public static</p> <p>Return: Task</p> <p>Purpose: Sends a “NewMessageBadge” notification to all active connections of the target user when a new chat message is received in a match, including match and pet metadata.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int matchId, int? fromPetId = null, int? toPetId = null</p>
15	SendNewLikeBadge	<p>Visibility: public static</p> <p>Return: Task</p> <p>Purpose: Sends a “NewLikeBadge” notification to all active connections of the target user when they receive a new like from another user.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int fromUserId</p>
16	SendNewExpertMessageBadge	<p>Visibility: public static</p> <p>Return: Task</p> <p>Purpose: Sends a “NewExpertMessageBadge” notification to all active connections of a user when there is a new message from an expert in an expert chat.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int chatExpertId</p>

17	SendMatchNotification	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a “MatchSuccess” notification to all active connections of a user when a new match is created, including information about the other user and pet details.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, string otherUserName, int otherUserId, int matchId, string? petName, string? petPhotoUrl</p>
18	SendNotification	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a general “NewNotification” event with title, message and type to all active connections of a specific user, logging diagnostic information about online status and connections.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, string title, string message, string type = "system"</p>
19	SendMatchDeleted Notification	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a “MatchDeleted” notification to all active connections of a user when a match is deleted or unmatched.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int matchId</p>
20	SendNotificationWithMetadata	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a “NewNotification” event with additional metadata such as expert id and chat id, typically for expert confirmation related notifications.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, string title, string message, string type, int? expertId = null, int? chatId = null</p>

21	SendExpertMessage	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends an expert chat message payload to the expert chat group and optionally directly to a specific user's active connections if they are not currently in the group.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int chatExpertId, int fromId, string message, int? toUserId = null</p>
16	SendNewExpertMessageBadge	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a "NewExpertMessageBadge" notification to all active connections of a user when there is a new message from an expert in an expert chat.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int chatExpertId</p>
17	SendMatchNotification	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a "MatchSuccess" notification to all active connections of a user when a new match is created, including information about the other user and pet details.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, string otherUserName, int otherUserId, int matchId, string? petName, string? petPhotoUrl</p>
18	SendNotification	<p>Visibility: public static Return: Task</p> <p>Purpose: Sends a general "NewNotification" event with title, message and type to all active connections of a specific user, logging diagnostic information about online status and connections.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, string title, string message, string type = "system"</p>

19	SendMatchDelete dNotification	<p>Visibility: public static</p> <p>Return: Task</p> <p>Purpose: Sends a “MatchDeleted” notification to all active connections of a user when a match is deleted or unmatched.</p> <p>Parameters: IHubContext<ChatHub> hubContext, int toUserId, int matchId</p>
----	----------------------------------	--

4.4.12 AuthService

No	Name	Description
Attributes		
01	_contentRepository	<p>Visibility: private</p> <p>Type: IChatUserContentRepository</p> <p>Purpose: Repository used to access and manage chat message records between matched users.</p>
02	_chatUserRepository	<p>Visibility: private</p> <p>Type: IChatUserRepository</p> <p>Purpose: Repository used to access and validate chat match records between users.</p>
03	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to query ChatUsers, pets and related entities for validation and data retrieval.</p>
04	_hubContext	<p>Visibility: private</p> <p>Type: IHubContext<ChatHub></p>

		Purpose: SignalR hub context used to send real-time chat messages and notifications to connected clients.
Methods/Operations		
01	GetChatMessages Async	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all messages of a specific match chat after validating that the chat exists; returns an empty collection if there are no messages yet.</p> <p>Parameters: int matchId, CancellationToken ct = default</p>
02	SendMessageAsyn c	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Sends a new message in a match chat after validating that the match exists and is accepted, determining which pet and user are sending and receiving, saving the message, broadcasting it via SignalR to the chat group and directly to the recipient, and triggering a new-message badge notification.</p> <p>Parameters: int matchId, int fromUserId, string message, CancellationToken ct = default</p>

4.4.13 ChatUserService

No	Name	Description
Attributes		
01	_chatUserReposito ry	<p>Visibility: private</p> <p>Type: IChatUserRepository</p>

		Purpose: Repository used to access and manage chat/match data between users based on pets.
02	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Entity Framework database context used to query pets, users and chat user entities.</p>
03	_limitService	<p>Visibility: private</p> <p>Type: DailyLimitService</p> <p>Purpose: Service used to check and record daily limits for user actions such as sending friend requests.</p>
Methods/Operations		
01	GetInvitesAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all pending friend invitations received by a user.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
02	GetChatsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat/match sessions for a user, optionally filtered by pet identifier.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default</p>
03	CreateFriendRequestAsync	<p>Visibility: public</p> <p>Return: Task<object></p>

		<p>Purpose: Creates a new friend request (match) between two pets after validating pets, checking daily request limits, handling mutual requests, and recording usage limits.</p> <p>Parameters: int fromPetId, int toPetId, CancellationToken ct = default</p>
04	UpdateFriendRequestAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Accepts an existing friend request by updating its status to “Accepted” and setting the update timestamp.</p> <p>Parameters: int matchId, CancellationToken ct = default</p>
05	DeleteFriendRequestAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Deletes an existing friend request if it exists.</p> <p>Parameters: int matchId, CancellationToken ct = default</p>

4.4.14 DailyLimitService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p> <p>Purpose: Entity Framework database context used to store and retrieve daily limit records, payment history and user action counts.</p>

02	FREE_TOKENS_PER_DAY	<p>Visibility: private const</p> <p>Type: int</p> <p>Purpose: Defines the maximum number of AI chat tokens a free user can use per day.</p>
03	VIP_TOKENS_PER_DAY	<p>Visibility: private const</p> <p>Type: int</p> <p>Purpose: Defines the maximum number of AI chat tokens a VIP user can use per day.</p>
04	_actionLimits	<p>Visibility: private readonly</p> <p>Type: Dictionary<string, (int FreeQuota, int VipLimit)></p> <p>Purpose: Stores daily action limits for different action types, differentiated between free users and VIP users.</p>
Methods/Operations		
01	IsVipUserAsync	<p>Visibility: private</p> <p>Return: Task<bool></p> <p>Purpose: Determines whether a user currently has an active VIP subscription based on payment history and service status.</p> <p>Parameters: int userId</p>
02	GetLimitForActionAsync	<p>Visibility: private</p> <p>Return: Task<int></p> <p>Purpose: Retrieves the daily limit for a specific action type based on whether the user is VIP or free; returns -1 for unlimited actions.</p> <p>Parameters: int userId, string actionType</p>

03	GetFreeQuotaForAction	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Retrieves the free user quota for a given action type; for AI chat, this represents the daily token limit instead of action count.</p> <p>Parameters: string actionType</p>
04	GetFreeTokensUsedToday	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Retrieves the total number of AI chat tokens a user has used on the current day.</p> <p>Parameters: int userId</p>
05	RecordTokenUsage	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Records the number of AI chat tokens consumed by a user for the current day, creating or updating the daily limit record as necessary.</p> <p>Parameters: int userId, int tokensUsed</p>
06	CanPerformAction	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a user is allowed to perform a specific action type based on daily limits and VIP status.</p> <p>Parameters: int userId, string actionType</p>
07	RecordAction	Visibility: public

		<p>Return: Task<bool></p> <p>Purpose: Records the execution of an action by a user and increments the daily count if the user has not exceeded the limit.</p> <p>Parameters: int userId, string actionType</p>
08	GetActionCountTo day	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Retrieves the number of times a user has performed a specific action type on the current day.</p> <p>Parameters: int userId, string actionType</p>
09	GetRemainingCou nt	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Retrieves the remaining number of allowed executions for a given action type on the current day.</p> <p>Parameters: int userId, string actionType</p>
10	GetRemainingCou ntAsync	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Calculates and returns the remaining action count for a given action type by subtracting today's usage from the allowed daily limit.</p> <p>Parameters: int userId, string actionType, CancellationToken ct = default</p>

4.4.15 DistanceService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Entity Framework database context used to load users and their addresses for distance calculation.</p>
Methods/Operations		
01	GetDistanceBetweenUsersAsync	<p>Visibility: public</p> <p>Return: Task<double?></p> <p>Purpose: Calculates the distance in kilometers between two users based on their stored addresses; returns null if either user does not have an address.</p> <p>Parameters: int userId1, int? userId2</p>
02	CalculateDistanceKm	<p>Visibility: public</p> <p>Return: double</p> <p>Purpose: Computes the geographic distance between two coordinate pairs (latitude/longitude) in kilometers using the Haversine formula.</p> <p>Parameters: double lat1, double lon1, double lat2, double lon2</p>
03	ToRadians	<p>Visibility: private</p> <p>Return: double</p> <p>Purpose: Converts a degree value to radians for use in trigonometric calculations.</p> <p>Parameters: double deg</p>

4.4.16 EmailService

No	Name	Description
Attributes		
01	_settings	<p>Visibility: private Type: EmailSettings Purpose: Stores SMTP configuration values such as server, port, sender email and credentials loaded from application settings.</p>
02	SmtpServer	<p>Visibility: public Type: string Purpose: Defines the SMTP server address used for sending emails.</p>
03	SmtpPort	<p>Visibility: public Type: int Purpose: Defines the SMTP server port number.</p>
04	SenderName	<p>Visibility: public Type: string Purpose: Represents the display name of the email sender.</p>
05	SenderEmail	<p>Visibility: public Type: string Purpose: Represents the email address used as the sender account.</p>
06	SenderPassword	Visibility: public

		<p>Type: string</p> <p>Purpose: Stores the password or app-specific password for authenticating with the SMTP server.</p>
Methods/Operations		
01	SendEmailAsync	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Sends an email using SMTP with the configured sender credentials, supporting HTML email content.</p> <p>Parameters: string toEmail, string subject, string body</p>

4.4.17 ExpertConfirmationService

No	Name	Description
Attributes		
01	_expertConfirmationRepository	<p>Visibility: private</p> <p>Type: IExpertConfirmationRepository</p> <p>Purpose: Repository used to access and manage expert confirmation records in the database.</p>
02	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Entity Framework database context used to query users, chats, expert confirmations and related entities.</p>
03	_dailyLimitService	<p>Visibility: private</p> <p>Type: DailyLimitService</p>

		Purpose: Service used to check and record daily limits for actions related to expert confirmations.
Methods/Operations		
01	GetAllExpertConfirmationsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation records from the system for administrative or monitoring purposes.</p> <p>Parameters: CancellationToken ct = default</p>
02	GetExpertConfirmationAsync	<p>Visibility: public</p> <p>Return: Task<ExpertConfirmationDTO?></p> <p>Purpose: Retrieves a specific expert confirmation by expert id, user id and chat id, and maps it to a DTO; returns null if not found.</p> <p>Parameters: int expertId, int userId, int chatId, CancellationToken ct = default</p>
03	GetUserExpertConfirmationsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmations associated with a specific user after validating that the user exists.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
04	CreateExpertConfirmationAsync	<p>Visibility: public</p> <p>Return: Task<ExpertConfirmationResponseDTO></p> <p>Purpose: Creates a new expert confirmation request for a given user and chat after checking daily limits, validating user</p>

		<p>and chat, optionally validating or auto-assigning an expert, and ensuring no duplicate request exists.</p> <p>Parameters: int userId, int chatId, ExpertConfirmationCreateDTO dto, CancellationToken ct = default</p>
05	UpdateExpertConfirmationAsync	<p>Visibility: public</p> <p>Return: Task<ExpertConfirmationResponseDTO></p> <p>Purpose: Updates the status and message of an existing expert confirmation; when the status changes to “confirmed” with a message, creates an in-app notification for the user and sends a real-time SignalR notification with metadata.</p> <p>Parameters: int expertId, int userId, int chatId, ExpertConfirmationUpdateDto dto, CancellationToken ct = default</p>
06	GetUserExpertChatsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves a list of expert chat sessions for a user including expert information, last message, time and unread count for use in chat list screens.</p> <p>Parameters: int userId, CancellationToken ct = default</p>

4.4.18 GeminiAIService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDbContext</p>

		Purpose: Entity Framework database context used to store and retrieve chat sessions and AI chat contents.
02	_configuration	<p>Visibility: private</p> <p>Type: IConfiguration</p> <p>Purpose: Provides application configuration values, including the Gemini AI API key.</p>
03	_googleAI	<p>Visibility: private readonly</p> <p>Type: GoogleAI</p> <p>Purpose: Google AI client used to call the Gemini generative model for answering user questions.</p>
Methods/Operations		
01	GetCatCareSystemPrompt	<p>Visibility: private</p> <p>Return: string</p> <p>Purpose: Builds and returns the fixed system prompt in Vietnamese that defines the AI's role as a cat care assistant, including tone, style and safety guidelines.</p> <p>Parameters: none</p>
02	CreateChatSessionAsync	<p>Visibility: public</p> <p>Return: Task<ChatAi></p> <p>Purpose: Creates a new ChatAi session for a user with a given title, marks it as not deleted and saves it to the database with current timestamps.</p> <p>Parameters: int userId, string title</p>

03	SendMessageAsync	<p>Visibility: public</p> <p>Return: Task<GeminiResponse></p> <p>Purpose: Sends a user question to the Gemini model within a specific chat session after validating ownership, constructing a prompt with system instructions and recent history, handling timeouts and errors, saving the Q&A to the database and updating chat metadata and title if needed.</p> <p>Parameters: int userId, int chatAId, string question</p>
----	------------------	--

4.4.19 KickboxSettings

No	Name	Description
Attributes		
01	ApiKey	<p>Visibility: public</p> <p>Type: string</p> <p>Owner: KickboxSettings</p> <p>Purpose: Stores the API key used to authenticate requests to the Kickbox email verification service.</p>
02	Endpoint	<p>Visibility: public</p> <p>Type: string</p> <p>Owner: KickboxSettings</p> <p>Purpose: Base URL endpoint of the Kickbox email verification API.</p>
03	_settings	<p>Visibility: private readonly</p> <p>Type: KickboxSettings</p> <p>Owner: KickboxClient</p>

		Purpose: Holds configuration values (API key and endpoint) for communicating with Kickbox.
Methods/Operations		
01	VerifyEmailAsync	<p>Visibility: public</p> <p>Return: Task<KickboxResponse></p> <p>Purpose: Builds a verification request URL using the configured endpoint, API key and encoded email address, sends an HTTP GET request to Kickbox, validates the response status and deserializes the JSON response into a KickboxResponse object.</p> <p>Parameters: string email</p>

4.4.20 MatchService

No	Name	Description
Attributes		
01	_chatUserRepository	<p>Visibility: private</p> <p>Type: IChatUserRepository</p> <p>Purpose: Repository used to access and manage ChatUser records (likes, matches, chat relations between pets/users).</p>
02	_blockRepository	<p>Visibility: private</p> <p>Type: IBlockRepository</p> <p>Purpose: Repository used to check and manage block relationships between users to avoid unwanted interactions.</p>
03	_notificationRepository	Visibility: private

		<p>Type: INotificationRepository</p> <p>Purpose: Repository used to store in-app notifications related to matches and likes.</p>
04	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Entity Framework database context used to query users, pets, chat users, blocks, notifications and related entities.</p>
05	_hubContext	<p>Visibility: private</p> <p>Type: IHubContext<ChatHub></p> <p>Purpose: SignalR hub context used to send real-time events such as new likes, matches and match deletions to connected clients.</p>
06	_dailyLimitService	<p>Visibility: private</p> <p>Type: DailyLimitService</p> <p>Purpose: Service used to enforce and record daily limits for match/like actions per user.</p>
Methods/Operations		
01	GetLikesReceivedA sync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<object>></p> <p>Purpose: Retrieves a list of likes and matches related to a user (optionally filtered by pet), excluding users who are blocked in either direction, and mapping each entry with owner, pet and photo info.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default</p>

02	GetStatsAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Calculates the total number of matches and pending likes for all pets belonging to a user for use in statistics or dashboard screens.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
03	SendLikeAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Processes a like request between two users' pets after checking daily limits, block status, existing likes and mutual likes; either creates a pending like or upgrades to an accepted match, triggers notifications/badges and returns match status and remaining quota.</p> <p>Parameters: LikeRequest request, CancellationToken ct = default</p>
04	RespondToLikeAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Handles user response to a like request; on "match" it accepts and creates a match with notifications, on "pass" it soft-deletes the ChatUser (including unmatch case) and may send real-time MatchDeleted notifications.</p> <p>Parameters: RespondRequest request, CancellationToken ct = default</p>
05	GetBadgeCountsAsync	<p>Visibility: public</p> <p>Return: Task<object></p>

		<p>Purpose: Computes badge data for a user including list of matchIds with unread messages (based on last sender) and the count of pending likes received for their pets, optionally filtered by a specific pet.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default</p>
06	CreateMatchNotification	<p>Visibility: private</p> <p>Return: Task</p> <p>Purpose: Creates and stores in-app notifications for both users when a new match occurs, including basic match message content; errors are logged but do not interrupt main flow.</p> <p>Parameters: int userId1, int userId2, int matchId, CancellationToken ct = default</p>
07	SendLikeNotification	<p>Visibility: private</p> <p>Return: Task</p> <p>Purpose: Sends a real-time like badge notification to the target user via SignalR when they receive a new like; logs errors without throwing exceptions.</p> <p>Parameters: int toUserId, int fromUserId</p>

4.4.21 NotificationService

No	Name	Description
Attributes		
01	_notificationRepository	<p>Visibility: private</p> <p>Type: INotificationRepository</p>

		Purpose: Repository used to access, create, update and delete notification records in the database.
02	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Entity Framework database context used to validate users and access related data when creating notifications.</p>
03	_hubContext	<p>Visibility: private</p> <p>Type: IHubContext<ChatHub></p> <p>Purpose: SignalR hub context used to send real-time notification events to connected clients.</p>
Methods/Operations		
01	GetAllNotificationsA sync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<NotificationDto>></p> <p>Purpose: Retrieves all notifications in the system, typically for administrative or debugging purposes.</p> <p>Parameters: CancellationToken ct = default</p>
02	GetNotificationById Async	<p>Visibility: public</p> <p>Return: Task<NotificationDto?></p> <p>Purpose: Retrieves a specific notification by its identifier and maps it to a DTO; returns null if not found.</p> <p>Parameters: int notificationId, CancellationToken ct = default</p>
03	GetNotificationsByU serIdAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<Notification>></p>

		<p>Purpose: Retrieves all notifications belonging to a specific user from the repository.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
04	CreateNotificationA sync	<p>Visibility: public</p> <p>Return: Task<Notification></p> <p>Purpose: Validates the notification DTO and target user, creates a new notification record with default unread status and timestamps, saves it to the database and attempts to send a real-time SignalR notification; logging any SignalR failure without throwing.</p> <p>Parameters: NotificationDto_1 notificationDto, CancellationToken ct = default</p>
05	MarkAsReadAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Marks a specific notification as read if it exists and updates its timestamp; returns false if the notification is not found.</p> <p>Parameters: int notificationId, CancellationToken ct = default</p>
06	MarkAllAsReadAsyn c	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Marks all notifications of a given user as read and returns the number of updated records.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
07	GetUnreadCountAsy nc	Visibility: public

		<p>Return: Task<int></p> <p>Purpose: Retrieves the total number of unread notifications for a specific user from the repository.</p> <p>Parameters: int userId, CancellationToken ct = default</p>
08	DeleteNotificationA sync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Deletes a notification by id if it exists; returns false when the notification cannot be found.</p> <p>Parameters: int notificationId, CancellationToken</p>

4.4.22 OtpService

No	Name	Description
Attributes		
01	_emailService	<p>Visibility: private</p> <p>Type: EmailService</p> <p>Purpose: Service responsible for sending OTP emails to users through SMTP.</p>
02	_cache	<p>Visibility: private</p> <p>Type: IMemoryCache</p> <p>Purpose: In-memory cache used to temporarily store OTP codes with expiration time for validation.</p>
03	_kickboxClient	<p>Visibility: private</p> <p>Type: IKickboxClient</p>

		Purpose: External email verification client intended for validating email credibility (currently injected for extensibility).
Methods/Operations		
01	SendOtpAsync	<p>Visibility: public</p> <p>Return: Task<object></p> <p>Purpose: Validates email format and existence based on the requested purpose (register or forgot-password), generates a random 6-digit OTP, sends it via email, and stores the OTP in memory cache with a 5-minute expiration. Throws appropriate exceptions when email is invalid, already exists, not found, or email delivery fails.</p> <p>Parameters: string email, string purpose = "register", CancellationToken ct = default</p>
02	CheckOtpAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Validates the provided email and OTP by comparing it with the cached value. Removes the OTP from cache upon successful verification. Throws exceptions when the OTP is incorrect, expired, or missing.</p> <p>Parameters: string email, string otp, CancellationToken ct = default</p>

4.4.23 PasswordService

No	Name	Description
Methods/Operations		

01	HashPassword	<p>Visibility: public</p> <p>Return: string</p> <p>Purpose: Generates a secure password hash using the BCrypt algorithm. This method is intended for all new passwords and password updates to ensure strong protection against brute-force attacks.</p> <p>Parameters: string password</p>
02	VerifyPassword	<p>Visibility: public</p> <p>Return: bool</p> <p>Purpose: Verifies an input password against a stored password hash. Supports both BCrypt hashes (current standard) and legacy SHA256 hashes for backward compatibility. Automatically determines the hash type based on prefix and format.</p> <p>Parameters: string inputPassword, string hashedPassword</p>
03	IsLegacyHash	<p>Visibility: public</p> <p>Return: bool</p> <p>Purpose: Determines whether the provided password hash uses the legacy SHA256 format. Used to identify accounts that require password re-hashing upon successful login.</p> <p>Parameters: string hashedPassword</p>
04	HashPasswordSHA256	<p>Visibility: private</p> <p>Return: string</p> <p>Purpose: Generates a SHA256 hash of the input password for legacy account verification. This method exists only to support</p>

		<p>old password records and should not be used for new password creation.</p> <p>Parameters: string password</p>
--	--	--

4.4.24 PaymentExpirationBackgroundService

No	Name	Description
Attributes		
01	_logger	ILogger<PaymentExpirationBackgroundService>. Used to log service lifecycle events, execution status, and errors during background processing.
02	_serviceProvider	IServiceProvider. Used to create scoped services for accessing the database context inside the background task.
03	_checkInterval	TimeSpan. Defines how often the service checks for expired payment records. Configured to run once every hour.
Methods/Operations		
01	ExecuteAsync	<p>Visibility: protected override</p> <p>Return: Task</p> <p>Purpose: Main execution loop of the background service. Runs continuously while the application is active, periodically triggering payment expiration checks and handling retry logic in case of errors.</p> <p>Parameters: CancellationToken stoppingToken</p>
02	CheckAndUpdateExpiredPaymentsAsynch	<p>Visibility: private</p> <p>Return: Task</p>

		<p>Purpose: Checks all payment history records where the subscription EndDate has passed and the StatusService is still marked as "active". Updates such records to the "pending" status and persists changes to the database.</p> <p>Parameters: CancellationToken ct</p>
--	--	--

4.4.25 PetCharacteristicService

No	Name	Description
Attributes		
01	_petCharacteristicRepository	<p>Visibility: private readonly</p> <p>Type: IPetCharacteristicRepository</p> <p>Purpose: Repository used to manipulate PetCharacteristic data (create, check existence, update, and query by pet).</p>
02	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDbContext</p> <p>Purpose: DbContext used to query and validate related data (Pets, Attributes, AttributeOptions).</p>
Methods/Operations		
01	GetPetCharacteristicsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves the list of characteristics of a pet by petId via the repository.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>

02	CreatePetCharacteristicAsync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Creates a new characteristic for a pet; validates pet existence, attribute validity, checks for duplicates; saves the value or corresponding option and returns the newly created characteristic data.</p> <p>Parameters: int petId, int attributeId, PetCharacteristicDTO dto, CancellationToken ct = default.</p>
03	UpdatePetCharacteristicAsync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Updates an existing pet characteristic; allows updating value or option, validates the option, updates the modification timestamp, and returns the updated characteristic data.</p> <p>Parameters: int petId, int attributeId, PetCharacteristicDTO dto, CancellationToken ct = default.</p>

4.4.26 PetImageAnalysisService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: DbContext used to query the Attributes, AttributeOptions, Pets, and PetCharacteristics tables.</p>
02	_configuration	Visibility: private readonly

		<p>Type: IConfiguration</p> <p>Purpose: Reads configuration related to Gemini AI (ApiKey) and other parameters if needed..</p>
03	_httpClient	<p>Visibility: private readonly</p> <p>Type: HttpClient</p> <p>Purpose: Used to call the Google Gemini Vision API for image analysis</p>
Methods/Operations		
01	AnalyzeImageAsync	<p>Visibility: public async</p> <p>Return type: Task<PetImageAnalysisResponse></p> <p>Purpose: Main workflow for pet image analysis: validates the image file, converts it to base64, retrieves the attribute list from the database, builds the prompt, calls the Gemini API, maps the results to database attributes, generates a sample SQL script, and returns the response to the client.</p> <p>Parameters: IFormFile image.</p>
02	BuildAnalysisPrompt	<p>Visibility: private</p> <p>Return type: string</p> <p>Purpose: Builds a detailed prompt (in Vietnamese) for the AI, listing attributes to be analyzed and enforcing a required JSON output format.</p> <p>Parameters: List<Attribute> attributes</p>
03	CallGeminiVisionAPI	<p>Visibility: private async</p> <p>Return type: Task<List<AttributeAnalysisResult>?></p>

		<p>Purpose: Calls the Google Gemini Vision API with a base64 image and prompt; handles common errors (API key, quota, location), parses the response text, extracts JSON, and converts it into a list of AttributeAnalysisResult.</p> <p>Parameters: string base64Image, string prompt, string contentType.</p>
04	ParseAttributeResults	<p>Visibility: private</p> <p>Return type: List<AttributeAnalysisResult></p> <p>Purpose: Parses the JSON returned by the AI into a list of AttributeAnalysisResult; flexibly handles numeric or string values and rounds to int when necessary.</p> <p>Parameters: string jsonText.</p>
05	EnrichWithDatabaseResults	<p>Visibility: private async</p> <p>Return type: Task</p> <p>Purpose: Enriches analysis results with AttributeId and OptionId by matching attributeName / optionName against database records.</p> <p>Parameters: List<AttributeAnalysisResult> results.</p>
06	GenerateSqlInsertScript	<p>Visibility: public async</p> <p>Return type: Task<string></p> <p>Purpose: Generates an SQL INSERT script for the PetCharacteristic table based on a list of AttributeAnalysisResult; supports both cases where petId is provided or resolved via SELECT by pet name.</p> <p>Parameters: int petId, List<AttributeAnalysisResult> attributes.</p>

07	InsertPetCharacteristicsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Persists analyzed characteristics into the PetCharacteristic table: checks pet existence, updates existing characteristics if present, inserts new ones if not, and saves changes.</p> <p>Parameters: int petId, List<AttributeAnalysisResult> attributes.</p>
----	-------------------------------	---

4.4.27 PetPhotoService

No	Name	Description
Attributes		
01	_photoRepository	<p>Visibility: private readonly</p> <p>Type: IPetPhotoRepository</p> <p>Purpose: Repository used to manage pet photo data (retrieve list, count photos, create, update, get by id, etc.).</p>
02	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDbContext</p> <p>Purpose: DbContext used to verify pet existence and query the PetPhotos table when needed..</p>
03	_storage	<p>Visibility: private readonly</p> <p>Type: IPhotoStorage</p> <p>Purpose: Service responsible for physical image storage (cloud/local), handling image upload and deletion.</p>
04	MaxPhotosPerPet	Visibility: private const

		<p>Type: int</p> <p>Purpose: Maximum number of photos allowed per pet (fixed value = 6).</p>
Methods/Operations		
01	GetPhotosByPetId Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<PetPhotoResponse>></p> <p>Purpose: Retrieves the list of photos for a pet; validates that the pet exists and is not deleted before querying the repository.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	UploadPhotosAsyn c	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<PetPhotoResponse>></p> <p>Purpose: Uploads multiple new photos for a pet: validates the file list, verifies pet existence, ensures the MaxPhotosPerPet limit is not exceeded, uploads files via IPhotoStorage, saves PetPhoto records with increasing sort order, and returns the saved photo information.</p> <p>Parameters: int petId, List<IFormFile> files, CancellationToken ct = default.</p>
03	ReorderPhotosAsy nc	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Reorders photo display sequence: receives a list of photoids with new sortOrder values, validates that all photos exist and are not deleted, updates SortOrder and UpdatedAt, and saves the changes.</p>

		Parameters: List<ReorderPhotoRequest> items, CancellationToken ct = default.
04	DeletePhotoAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Deletes a pet photo: performs a soft delete by setting IsDeleted = true; if the hard parameter is true and a PublicId exists, calls IPhotoStorage.DeleteAsync to remove the physical file. Returns false if the photo does not exist or has already been deleted.</p> <p>Parameters: int photoid, bool hard = false, CancellationToken ct = default.</p>

4.4.28 PetRecommendationService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDbContext</p> <p>Purpose: DbContext used to query Users, Pets, UserPreferences, ChatUsers, Blocks and PetPhotos for pet recommendation logic.</p>
02	_distanceService	<p>Visibility: private readonly</p> <p>Type: DistanceService</p> <p>Purpose: Service responsible for calculating distance between users based on address information, used for distance-based filtering in recommendations.</p>

Methods/Operations		
01	RecommendPetsA sync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Recommends a list of suitable pets for a user by loading user preferences and address, reading distance preference if available, excluding already matched or blocked users, loading active pets, calculating matching scores based on preference weight percentages, filtering by distance when applied, sorting by match percentage and distance, selecting the top 20 pets, and returning the result with summary information.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

4.4.29 PetService

No	Name	Description
Attributes		
01	_petRepository	<p>Visibility: private readonly</p> <p>Type: IPetRepository</p> <p>Purpose: Repository responsible for pet data access such as retrieving, creating, updating, deleting pets and matching-related queries.</p>
02	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDbContext</p> <p>Purpose: DbContext used for advanced queries involving pets, chat relationships, blocked users, and pet characteristics.</p>

Methods/Operations		
01	GetPetsByUserIdA sync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<PetDto>></p> <p>Purpose: Retrieves all pets belonging to a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetPetsForMatchi ngAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves a list of pets available for matching by excluding pets from users already matched or blocked by the current user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
03	GetPetByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<object?></p> <p>Purpose: Retrieves detailed information of a pet including owner, address, images, and calculates age from pet characteristics if available.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
04	CreatePetAsync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Creates a new pet after validating input data and returns basic information of the created pet.</p> <p>Parameters: PetDto_2 petDto, CancellationToken ct = default.</p>
05	UpdatePetAsync	Visibility: public async

		<p>Return type: Task<object></p> <p>Purpose: Updates an existing pet's information such as name, breed, gender, age, status, and description.</p> <p>Parameters: int petId, PetDto_2 updatedPet, CancellationToken ct = default.</p>
06	DeletePetAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Performs a soft delete on a pet by marking it as deleted instead of removing it from the database.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
07	SetActivePetAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Sets a pet as active and deactivates all other pets belonging to the same user.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>

4.4.30 ReportService

No	Name	Description
Attributes		
01	_reportRepository	<p>Visibility: private readonly</p> <p>Type: IReportRepository</p> <p>Purpose: Handles data access for reports such as querying, creating, and updating report records.</p>

02	<code>_context</code>	<p>Visibility: private readonly</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: DbContext used to validate users, chat contents, blocks and to manage related entities when creating or processing reports.</p>
03	<code>_notificationService</code>	<p>Visibility: private readonly</p> <p>Type: INotificationService</p> <p>Purpose: Responsible for creating notifications to inform users about the status and resolution of their reports.</p>
Methods/Operations		
01	<code>GetAllReportsAsync</code>	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<ReportDto>></p> <p>Purpose: Retrieves all reports in the system for admin or management purposes.</p> <p>Parameters: CancellationToken ct = default.</p>
02	<code>GetReportByIdAsync</code>	<p>Visibility: public async</p> <p>Return type: Task<ReportDto?></p> <p>Purpose: Retrieves a specific report by its ID, including basic report information.</p> <p>Parameters: int reportId, CancellationToken ct = default.</p>
03	<code>GetReportsByUserIdAsync</code>	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all reports submitted by a specific user (reporting user).</p>

		Parameters: int userReportId, CancellationToken ct = default.
04	CreateReportAsyn c	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Creates a new report for a chat message: validates reporter user, validates content, resolves the reported user through the pet, prevents self-reporting, creates the report record, automatically blocks the reported user if not already blocked, soft-deletes the related chat, and returns the created report data.</p> <p>Parameters: int userReportId, int contentId, ReportCreateDTO dto, CancellationToken ct = default.</p>
05	UpdateReportAsyn c	<p>Visibility: public async</p> <p>Return type: Task<ReportDto></p> <p>Purpose: Updates the status and resolution of an existing report, refreshes the updated time, reloads the full report with user info, and sends a notification to the reporting user about the outcome (resolved, rejected, or updated).</p> <p>Parameters: int reportId, ReportUpdateDTO dto, CancellationToken ct = default.</p>

4.4.31 sePayService

No	Name	Description
Attributes		
01	_client	Visibility: private readonly

		<p>Type: HttpClient</p> <p>Purpose: HTTP client used to call SePay APIs, configured with a 15-second timeout and JSON headers.</p>
02	_config	<p>Visibility: private readonly</p> <p>Type: IConfiguration</p> <p>Purpose: Provides access to SePay configuration values such as ApiUrl, ApiKey, AccountNumber and Limit from app settings.</p>
Methods/Operations		
01	GetTransactionsAs ync	<p>Visibility: public async</p> <p>Return type: Task</p> <p>Purpose: Retrieves the recent transaction list from SePay API using the configured account number and limit, logs the request URL and raw JSON response to the console.</p> <p>Parameters: None.</p>
02	GetTransactionDet ailsAsync	<p>Visibility: public async</p> <p>Return type: Task</p> <p>Purpose: Retrieves detailed information of a specific transaction from SePay API based on a transactionId, validates the input, calls the details endpoint, and logs either the formatted JSON response or error message to the console.</p> <p>Parameters: string transactionId.</p>
03	CallWithRetryAsyn c	<p>Visibility: private async</p> <p>Return type: Task<HttpResponseMessage></p> <p>Purpose: Generic retry logic for calling SePay APIs: attempts a GET request up to maxRetries times, logs non-success status</p>

		codes and connection errors, applies an incremental delay between retries and throws an exception if all attempts fail. Parameters: string url, int maxRetries.
--	--	--

4.4.32 TokenService

No	Name	Description
Attributes		
01	_secret	Visibility: private readonly Type: string Purpose: Secret key used to sign JWT tokens, loaded from Jwt:Secret configuration.
02	_issuer	Visibility: private readonly Type: string Purpose: JWT issuer identifier, loaded from Jwt:Issuer configuration.
03	_audience	Visibility: private readonly Type: string Purpose: JWT audience identifier, loaded from Jwt:Audience configuration.
04	_accessTokenExpirationMinutes	Visibility: private readonly Type: int Purpose: Access token expiration time in minutes (short-lived, default 1 minute).
05	_refreshTokenExpirationDays	Visibility: private readonly

		<p>Type: int</p> <p>Purpose: Refresh token expiration time in days (long-lived, default 30 days).</p>
Methods/Operations		
01	GenerateAccessToken	<p>Visibility: public</p> <p>Return type: string</p> <p>Purpose: Generates a short-lived JWT access token containing userId and role claims, signed with HmacSha256 for authentication and authorization.</p> <p>Parameters: int userId, string role.</p>
02	GenerateRefreshToken	<p>Visibility: public</p> <p>Return type: string</p> <p>Purpose: Generates a long-lived JWT refresh token containing userId, role, and a token_type claim, used to obtain new access tokens without re-authentication.</p> <p>Parameters: int userId, string role.</p>

4.4.33 UserPreferenceService

No	Name	Description
Attributes		
01	_userPreferenceRepository	<p>Visibility: private readonly</p> <p>Type: IUserPreferenceRepository</p> <p>Purpose: Handles data access for user preferences, including querying, creating, updating and deleting UserPreference entities.</p>

02	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: DbContext used to validate users and attributes and to support business rules related to user preferences.</p>
Methods/Operations		
01	GetUserPreferencesAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<UserPreferenceResponse>></p> <p>Purpose: Validates that the user exists and is not deleted, then retrieves all preferences of the specified user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	CreateUserPreferenceAsync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Creates a new user preference for a specific attribute after validating the user, validating the attribute and checking that a preference for this attribute does not already exist.</p> <p>Parameters: int userId, int attributeId, UserPreferenceUpsertRequest req, CancellationToken ct = default.</p>
03	UpdateUserPreferenceAsync	<p>Visibility: public async</p> <p>Return type: Task<UserPreferenceResponse></p> <p>Purpose: Updates an existing user preference (option and numeric range), refreshes the UpdatedAt timestamp and returns the updated preference with attribute and option details.</p>

		Parameters: int userId, int attributeId, UserPreferenceUpsertRequest req, CancellationToken ct = default.
04	DeleteUserPreferencesAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Deletes all preferences of a given user; returns false if the user has no preferences, otherwise deletes them in bulk and returns true.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
05	UpsertBatchAsync	<p>Visibility: public async</p> <p>Return type: Task<object></p> <p>Purpose: Performs batch upsert of user preferences: validates user and attribute IDs, loads existing preferences, deletes preferences not present in the request, updates existing preferences and creates new ones as needed, then returns a summary (created, updated, deleted) with a message.</p> <p>Parameters: int userId, UserPreferenceBatchUpsertRequest request, CancellationToken ct = default.</p>

4.4.34 UserService

No	Name	Description
Attributes		
01	_userRepository	<p>Visibility: private readonly</p> <p>Type: IUserRepository</p> <p>Purpose: Handles data access operations for users such as querying, creating, updating and soft-deleting user records.</p>

02	_passwordService	<p>Visibility: private readonly</p> <p>Type: PasswordService</p> <p>Purpose: Provides password hashing functionality used when registering users and resetting passwords.</p>
Methods/Operations		
01	GetUsersAsync	<p>Visibility: public async</p> <p>Return type: Task<PagedResult<UserResponse>></p> <p>Purpose: Retrieves a paginated list of users filtered by search text, role, status and deletion state; validates and normalizes page and pageSize values before querying the repository.</p> <p>Parameters: string? search, int? roleId, int? statusId, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.</p>
02	GetUserByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<UserResponse?></p> <p>Purpose: Retrieves a single user by ID (excluding deleted users by default) and maps the result to a UserResponse DTO; returns null if the user is not found.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
03	RegisterAsync	<p>Visibility: public async</p> <p>Return type: Task<UserResponse></p> <p>Purpose: Registers a new user: trims email and password, checks if the email already exists, hashes the password, sets default role (RoleId = 3) and default status if not provided, saves the user and returns a UserResponse DTO.</p>

		Parameters: UserCreateRequest req, CancellationToken ct = default.
04	UpdateUserAsync	<p>Visibility: public async</p> <p>Return type: Task<UserResponse></p> <p>Purpose: Updates basic user information such as addressId, full name, gender and optionally password, then saves changes and returns the updated user as a UserResponse DTO.</p> <p>Parameters: int userId, UserUpdateRequest req, CancellationToken ct = default.</p>
05	SoftDeleteUserAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Performs a soft delete on a user by setting IsDeleted to true and updating the timestamp; returns false if the user does not exist, or true if deletion was applied or already done.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
06	CompleteProfileAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Marks a user's profile as complete by setting IsProfileComplete to true and updating the timestamp; returns false if the user does not exist.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
07	ResetPasswordAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p>

		<p>Purpose: Resets the user's password: validates input email and new password, finds the user by email, hashes the new password, clears the stored JWT token for security and saves the changes.</p> <p>Parameters: ResetPasswordRequest request, CancellationToken ct = default.</p>
--	--	--

4.4.35 VietQRService

No	Name	Description
Attributes		
01	ApiUrl	<p>Visibility: private const</p> <p>Type: string</p> <p>Purpose: VietQR API endpoint used to generate payment QR codes.</p>
02	ClientId	<p>Visibility: private const</p> <p>Type: string</p> <p>Purpose: Client identifier provided by VietQR, used for API authentication.</p>
03	ApiKey	<p>Visibility: private const</p> <p>Type: string</p> <p>Purpose: API key used to authorize requests to the VietQR service.</p>
Methods/Operations		
01	GenerateQRAsync	<p>Visibility: public async</p> <p>Return type: Task<VietQRResponse></p> <p>Purpose: Calls the VietQR API to generate a payment QR code by sending account information, amount and transaction</p>

		<p>description; returns the response containing the base64-encoded QR image.</p> <p>Parameters: VietQRRequest request.</p>
--	--	--

4.4.36 AppointmentService

No	Name	Description
Attributes		
01	_appointmentRepository	<p>Visibility: private</p> <p>Type: IAppointmentRepository</p> <p>Purpose: Provides data access methods for PetAppointment, including match-based queries and message/profile checks.</p>
02	_locationRepository	<p>Visibility: private</p> <p>Type: IAppointmentLocationRepository</p> <p>Purpose: Provides data access methods for PetAppointmentLocation and location validation/lookup.</p>
03	_chatUserRepository	<p>Visibility: private</p> <p>Type: IChatUserRepository</p> <p>Purpose: Validates match existence/acceptance and retrieves match-related chat user information.</p>
04	_notificationService	<p>Visibility: private</p> <p>Type: INotificationService</p> <p>Purpose: Sends notifications for appointment lifecycle events (invite, accept, reject, counter-offer, cancel, ongoing, completed, expired).</p>
05	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p>

		Purpose: Direct database access via Entity Framework for complex queries and background processing (expired appointments, appointment checks).
06	MIN_MESSAGES_REQUIRED	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Minimum number of chat messages required before allowing appointment creation (value: 10).</p>
07	MIN_HOURS_ADVANCE	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Minimum hours required in advance for appointment scheduling (value: 2).</p>
08	MAX_COUNTER_OFFERS	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Maximum allowed counter-offer attempts per appointment (value: 3).</p>
09	CHECK_IN_RADIUS_METERS	<p>Visibility: private (const)</p> <p>Type: double</p> <p>Purpose: Maximum distance from appointment location to allow GPS check-in (value: 100m).</p>
10	CHECK_IN_BEFORE_MINUTES	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Earliest allowed check-in window before appointment time (value: 30 minutes).</p>
11	CHECK_IN_AFTER_MINUTES	<p>Visibility: private (const)</p> <p>Type: int</p>

		Purpose: Latest allowed check-in window after appointment time (value: 90 minutes).
12	AUTO_NO_SHOW_MINUTES	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Automatically marks confirmed appointments as NO_SHOW when check-in is missing after threshold (value: 90 minutes).</p>
13	AUTO_COMPLETE_MINUTES	<p>Visibility: private (const)</p> <p>Type: int</p> <p>Purpose: Automatically completes on-going appointments after threshold (value: 90 minutes).</p>
Methods/Operations		
01	AppointmentService()	<p>Return type: Constructor Purpose: Initializes AppointmentService with required repositories, services, and database context via dependency injection. Parameters: IAppointmentRepository appointmentRepository, IAppointmentLocationRepository locationRepository, IChatUserRepository chatUserRepository, INotificationService notificationService, PawnderDbContext context.</p>
02	ValidatePreConditionsAsync()	<p>Return type: Task< (bool IsValid, string? ErrorMessage) ></p> <p>Purpose: Validates prerequisites before creating an appointment: not self-appointment, match accepted, no existing pending/confirmed appointment, minimum message count, and pet profile completeness. Parameters: int matchId, int inviterPetId, int inviteePetId, CancellationToken ct = default.</p>

03	CreateAppointmentAsync()	Return type: Task<AppointmentResponse> Purpose: Creates a new appointment (status = pending), validates time constraints, resolves location (existing or custom), persists data, and sends invitation notification. Parameters: int userId, CreateAppointmentRequest request, CancellationToken ct = default.
04	GetAppointmentByIdAsync()	Return type: Task<AppointmentResponse?> Purpose: Retrieves appointment by ID with full details (navigation properties) and maps to AppointmentResponse. Parameters: int appointmentId, CancellationToken ct = default.
05	GetAppointmentsByMatchIdAsync()	Return type: Task<IEnumerable<AppointmentResponse>> Purpose: Retrieves appointments for a given match and maps results to response DTOs. Parameters: int matchId, CancellationToken ct = default.
06	GetAppointmentsByUserIdAsync()	Return type: Task<IEnumerable<AppointmentResponse>> Purpose: Retrieves appointments associated with a user (inviter/invitee) and maps results to response DTOs. Parameters: int userId, CancellationToken ct = default.
07	RespondToAppointmentAsync()	Return type: Task<AppointmentResponse> Purpose: Allows the current decision user to accept or reject a pending appointment, updates status, stores decline reason if rejected, and sends notifications to both parties.

		Parameters: int userId, RespondAppointmentRequest request, CancellationToken ct = default.
08	CounterOfferAsync()	<p>Return type: Task<AppointmentResponse></p> <p>Purpose: Proposes new time and/or location, enforces counter-offer limit, switches decision ownership to the other user, updates count, and sends counter-offer notification.</p> <p>Parameters: int userId, CounterOfferRequest request, CancellationToken ct = default.</p>
09	CancelAppointmentAsync()	<p>Return type: Task<AppointmentResponse></p> <p>Purpose: Cancels an appointment by authorized user, appends “last-minute” note when applicable, updates status, and notifies the other participant.</p> <p>Parameters: int userId, CancelAppointmentRequest request, CancellationToken ct = default.</p>
10	CheckInAsync()	<p>Return type: Task<AppointmentResponse></p> <p>Purpose: Performs GPS-based check-in within allowed time window and radius; when both users check in, transitions status to on_going and notifies both users.</p> <p>Parameters: int userId, CheckInRequest request, CancellationToken ct = default.</p>
11	CompleteAppointmentAsync()	<p>Return type: Task<AppointmentResponse></p> <p>Purpose: Manually completes an on-going appointment (must be a participant; must be after appointment time) and notifies both users.</p> <p>Parameters: int userId, int appointmentId, CancellationToken ct = default.</p>

12	ProcessExpiredAppointmentsAsync()	<p>Return type: Task</p> <p>Purpose: Background processing for expired appointments using Vietnam time:</p> <ul style="list-style-type: none"> • pending → expired (past appointment time) • confirmed → no_show (missing check-in after threshold) • on-going → completed (auto-complete after threshold) <p>Sends notifications for each transition and saves changes.</p> <p>Parameters: CancellationToken ct = default.</p>
13	CreateLocationAsync()	<p>Return type: Task<LocationResponse></p> <p>Purpose: Creates a new appointment location; prevents duplicates by GooglePlaceId and maps result to LocationResponse.</p> <p>Parameters: CreateLocationRequest request, CancellationToken ct = default.</p>
14	GetRecentLocationsAsync()	<p>Return type: Task<IEnumerable<LocationResponse>></p> <p>Purpose: Retrieves distinct recently used locations from user's appointment history, sorted by newest, limited by given count.</p> <p>Parameters: int userId, int limit = 10, CancellationToken ct = default.</p>
15	MapToResponse()	<p>Return type: AppointmentResponse</p> <p>Purpose: Maps PetAppointment domain entity (including navigation properties) into AppointmentResponse DTO.</p> <p>Parameters: PetAppointment a.</p>

16	MapLocationToResponse()	<p>Return type: LocationResponse</p> <p>Purpose: Maps PetAppointmentLocation entity into LocationResponse DTO.</p> <p>Parameters: PetAppointmentLocation l.</p>
17	CalculateDistance()	<p>Return type: double Purpose: Calculates distance between two coordinates (meters) using Haversine formula for check-in validation. Parameters: decimal lat1, decimal lon1, decimal lat2, decimal lon2.</p>
18	GetVietnamTime()	<p>Return type: DateTime Purpose: Retrieves current Vietnam time (UTC+7) with fallback support for Windows and Linux timezone IDs. Parameters: None.</p>
19	ToRadians()	<p>Return type: double Purpose: Converts degrees to radians for distance calculations. Parameters: double degrees.</p>

4.4.37 EventService

No	Name	Description
Attributes		
01	_eventRepository	<p>Visibility: private Type: IEventRepository Purpose: Provides data access methods for PetEvent entities, including admin listing, active events retrieval, and transition querying.</p>

02	_submissionRepository	Visibility: private Type: ISubmissionRepository Purpose: Provides data access methods for EventSubmission entities, including voting checks, leaderboard queries, and submission details retrieval.
03	_notificationService	Visibility: private Type: INotificationService Purpose: Sends notifications related to event lifecycle (created, updated, cancelled), voting milestones, and winner announcements.
04	_context	Visibility: private Type: PawnderDatabaseContext Purpose: Direct database access via Entity Framework for user/pet validation and global user notification broadcasting.
Methods/Operations		
01	EventService()	<p>Return type: Constructor</p> <p>Purpose: Initializes EventService with repositories, notification service, and database context via dependency injection.</p> <p>Parameters: IEventRepository eventRepository, ISubmissionRepository submissionRepository, INotificationService notificationService, PawnderDatabaseContext context.</p>
02	GetAllEventsAsync()	<p>Return type: Task<IEnumerable<EventResponse>></p> <p>Purpose: Retrieves all events (all statuses) for administrative management and maps to EventResponse.</p>

		Parameters: CancellationToken ct = default.
03	CreateEventAsync()	<p>Return type: Task<EventResponse></p> <p>Purpose: Creates a new event after validating time constraints (StartTime < SubmissionDeadline < EndTime; StartTime in the future), persists the event, and broadcasts “event created” notifications to all active users.</p> <p>Parameters: int adminId, CreateEventRequest request, CancellationToken ct = default.</p>
04	SafeSendNotificationAsync()	<p>Return type: Task</p> <p>Purpose: Sends a notification safely without throwing exceptions, preventing service interruption during bulk notification broadcast.</p> <p>Parameters: int userId, string title, string message, string type.</p>
05	UpdateEventAsync()	<p>Return type: Task<EventResponse></p> <p>Purpose: Updates an existing event (not allowed for completed/cancelled), applies partial field updates, persists changes, and broadcasts “event updated” notifications to all active users.</p> <p>Parameters: int eventId, UpdateEventRequest request, CancellationToken ct = default.</p>
06	CancelEventAsync()	<p>Return type: Task</p> <p>Purpose: Cancels an event (not allowed if completed), updates status to cancelled, and broadcasts cancellation notifications to all active users with optional reason.</p> <p>Parameters: int eventId, string? reason, CancellationToken ct = default.</p>

07	GetActiveEventsAs ync()	<p>Return type: Task<IEnumerable<EventResponse>></p> <p>Purpose: Retrieves events that are active or upcoming for user visibility and maps to EventResponse.</p> <p>Parameters: CancellationToken ct = default.</p>
08	GetEventByIdAsyn c()	<p>Return type: Task<EventDetailResponse?></p> <p>Purpose: Retrieves event details including submissions and winners; calculates submission count and total votes; supports user context (HasVoted/IsOwner) when currentUserId is provided.</p> <p>Parameters: int eventId, int? currentUserId = null, CancellationToken ct = default.</p>
09	SubmitEntryAsync()	<p>Return type: Task<SubmissionResponse></p> <p>Purpose: Submits an entry to an event with runtime time validation (StartTime ≤ now ≤ SubmissionDeadline), prevents duplicate submissions per user, validates pet ownership, validates media type and size, persists submission, and returns detailed SubmissionResponse.</p> <p>Parameters: int userId, SubmitEntryRequest request, CancellationToken ct = default.</p>
10	VoteAsync()	<p>Return type: Task</p> <p>Purpose: Registers a user vote for a submission with runtime time validation (StartTime ≤ now ≤ EndTime), prevents self-voting and duplicate voting, updates vote records, and sends milestone notifications (1, 5, 10, 20, 50, 100, every +50 after 100).</p> <p>Parameters: int userId, int submissionId, CancellationToken ct = default.</p>

11	UnvoteAsync()	Return type: Task Purpose: Removes a user vote for a submission; disallowed after event end/completion; validates that the user has already voted before removal. Parameters: int userId, int submissionId, CancellationToken ct = default.
12	GetLeaderboardAsync()	Return type: Task<IEnumerable<LeaderboardResponse>> Purpose: Retrieves the top leaderboard entries for an event (Top 10) and assigns ranking order in the response. Parameters: int eventId, int? currentUserId = null, CancellationToken ct = default.
13	ProcessEventTransitionsAsync()	Return type: Task Purpose: Background job that transitions event status based on current time: upcoming → active → submission_closed → voting_ended, and triggers result processing when voting ends. Parameters: CancellationToken ct = default.
14	ProcessEventResultsAsync()	Return type: Task Purpose: Calculates winners (Top 3 by VoteCount then CreatedAt), marks submissions with rank and winner flags, sends winner notifications, and finalizes event status to completed. Parameters: int eventId, CancellationToken ct = default.
15	MapToResponse()	Return type: EventResponse Purpose: Maps PetEvent domain entity to EventResponse, including derived metrics (submission count and total votes). Parameters: PetEvent e.

16	MapSubmissionToResponse()	<p>Return type: SubmissionResponse Purpose: Maps EventSubmission entity to SubmissionResponse, resolves pet primary photo, and computes user-context fields (HasVoted, IsOwner) when currentUserId is provided. Parameters: EventSubmission s, int? currentUserId.</p>

4.4.38 PolicyService

No	Name	Description
Attributes		
01	_policyRepository	<p>Visibility: private Type: IPolicyRepository Purpose: Provides data access for Policy, PolicyVersion, and UserPolicyAccept entities, including versioning, publishing, acceptance tracking, and statistics.</p>
Methods/Operations		
01	PolicyService()	<p>Return type: Constructor Purpose: Initializes PolicyService with the required policy repository via dependency injection. Parameters: IPolicyRepository policyRepository.</p>
02	GetAllPoliciesAsync()	<p>Return type: Task<List<PolicyResponse>> Purpose: Retrieves all policies for admin management, including active version and total version count for each policy.</p>

		Parameters: CancellationToken ct = default.
03	GetPolicyByIdAsync()	<p>Return type: Task<PolicyResponse></p> <p>Purpose: Retrieves detailed information of a specific policy by ID, including its active version and total number of versions.</p> <p>Parameters: int policyId, CancellationToken ct = default.</p>
04	CreatePolicyAsync()	<p>Return type: Task<PolicyResponse></p> <p>Purpose: Creates a new policy after validating uniqueness of policy code; initializes policy as active and not deleted.</p> <p>Parameters: CreatePolicyRequest request, CancellationToken ct = default.</p>
05	UpdatePolicyAsync()	<p>Return type: Task<PolicyResponse></p> <p>Purpose: Updates policy metadata such as name, description, display order, consent requirement, and active status.</p> <p>Parameters: int policyId, UpdatePolicyRequest request, CancellationToken ct = default.</p>
06	DeletePolicyAsync()	<p>Return type: Task<bool></p> <p>Purpose: Performs soft deletion of a policy while preserving historical data and accept records.</p> <p>Parameters: int policyId, CancellationToken ct = default.</p>
07	GetVersionsByPolicyIdAsync()	<p>Return type: Task<List<PolicyVersionResponse>></p> <p>Purpose: Retrieves all versions associated with a policy and maps them to response DTOs.</p> <p>Parameters: int policyId, CancellationToken ct = default.</p>

08	GetVersionByIdAsyn()	<p>Return type: Task<PolicyVersionResponse></p> <p>Purpose: Retrieves detailed information of a specific policy version by ID.</p> <p>Parameters: int policyVersionId, CancellationToken ct = default.</p>
09	CreateVersionAsyn()	<p>Return type: Task<PolicyVersionResponse></p> <p>Purpose: Creates a new policy version in DRAFT state with auto-incremented version number.</p> <p>Parameters: int policyId, CreatePolicyVersionRequest request, int adminUserId, CancellationToken ct = default.</p>
10	UpdateVersionAsyn()	<p>Return type: Task<PolicyVersionResponse></p> <p>Purpose: Updates a policy version's content, title, or change log; only allowed when version status is DRAFT.</p> <p>Parameters: int policyVersionId, UpdatePolicyVersionRequest request, CancellationToken ct = default.</p>
11	PublishVersionAsyn()	<p>Return type: Task<PolicyVersionResponse></p> <p>Purpose: Publishes a DRAFT policy version by setting it to ACTIVE, deactivating the previous active version, and invalidating all related user accepts.</p> <p>Parameters: int policyVersionId, CancellationToken ct = default.</p>
12	DeleteVersionAsyn()	<p>Return type: Task<bool></p> <p>Purpose: Deletes a policy version; only permitted if the version is in DRAFT state.</p>

		Parameters: int policyVersionId, CancellationToken ct = default.
13	GetAcceptStatsAsync()	<p>Return type: Task<List<PolicyAcceptStatsResponse>></p> <p>Purpose: Retrieves acceptance statistics for required policies, including accepted users, pending users, and acceptance rate.</p> <p>Parameters: CancellationToken ct = default.</p>
14	CheckPolicyStatusAsync()	<p>Return type: Task<PolicyStatusResponse></p> <p>Purpose: Determines whether a user has accepted all required active policy versions and returns compliance status.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
15	GetPendingPoliciesAsync()	<p>Return type: Task<List<PendingPolicyResponse>></p> <p>Purpose: Retrieves the list of required policies that the user has not yet accepted, including previous acceptance information if applicable.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
16	AcceptPolicyAsync()	<p>Return type: Task<PolicyStatusResponse></p> <p>Purpose: Records user acceptance of a specific active policy version, invalidates older accepts for the same policy, and updates compliance status.</p> <p>Parameters: int userId, AcceptPolicyRequest request, CancellationToken ct = default.</p>

17	AcceptMultiplePoliciesAsync()	Return type: Task<PolicyStatusResponse> Purpose: Allows a user to accept multiple policies in one request and returns the updated policy compliance status. Parameters: int userId, AcceptMultiplePoliciesRequest request, CancellationToken ct = default.
18	GetUserAcceptHistoryAsync()	Return type: Task<List<UserAcceptHistoryResponse>> Purpose: Retrieves the complete policy acceptance history of a user for auditing and compliance tracking. Parameters: int userId, CancellationToken ct = default.
19	GetActivePolicyContentAsync()	Return type: Task<PendingPolicyResponse> Purpose: Retrieves the content of the currently active version of a policy for user review. Parameters: string policyCode, CancellationToken ct = default.
20	GetAllActivePoliciesAsync()	Return type: Task<List<PendingPolicyResponse>> Purpose: Retrieves all active required policy versions available for users to read. Parameters: CancellationToken ct = default.
21	MapToPolicyResponse()	Return type: PolicyResponse Purpose: Maps Policy entity to PolicyResponse, including active version and total version count. Parameters: Policy policy, PolicyVersion? activeVersion, int totalVersions.

22	MapToVersionResponse()	Return type: PolicyVersionResponse Purpose: Maps PolicyVersion entity to PolicyVersionResponse with creator and status metadata. Parameters: PolicyVersion version, string policyCode.
----	------------------------	--

4.4.39 BadWordService

No	Name	Description
Attributes		
01	_badWordRepository	Visibility: private Type: IBadWordRepository Purpose: Retrieves prohibited words from database (active rules), including filtering by activation and severity levels.
02	_cache	Visibility: private Type: IMemoryCache Purpose: Caches active prohibited word rules to improve runtime performance for message moderation.
03	CACHE_KEY	Visibility: private (const) Type: string Purpose: Cache key identifier for storing active bad words (value: "BadWords_Active").
04	CACHE_DURATION_MINUTES	Visibility: private (const) Type: int Purpose: Cache absolute expiration time in minutes (value: 30).

Methods/Operations		
01	BadWordService()	<p>Return type: Constructor</p> <p>Purpose: Initializes BadWordService with bad word repository and memory cache via dependency injection.</p> <p>Parameters: IBadWordRepository badWordRepository, IMemoryCache cache.</p>
02	CheckAndFilterMessageAsync()	<p>Return type: Task< (bool isBlocked, string filteredMessage, int violationLevel) ></p> <p>Purpose: Checks a message against active prohibited word rules and returns moderation result:</p> <ul style="list-style-type: none"> • Level 0: No violation • Level 1: Mask matched words with "****" (filter) • Level ≥2: Block the message completely (isBlocked = true) <p>Supports both regex-based rules and plain text rules with anti-evasion normalization.</p> <p>Parameters: string message, CancellationToken ct = default.</p>
03	NormalizeText()	<p>Return type: string</p> <p>Purpose: Normalizes text to mitigate evasion attempts by removing whitespace/special characters/zero-width chars, removing control characters, limiting repeated characters, and converting common leetspeak substitutions (e.g., 0→o, 1→i).</p> <p>Parameters: string text.</p>
04	CreateFlexiblePattern()	Return type: string

		<p>Purpose: Builds a flexible regex pattern for plain text bad words to match variants containing separators (whitespace, punctuation, zero-width characters) between letters.</p> <p>Parameters: string word.</p>
05	ReloadCacheAsync() 	<p>Return type: Task</p> <p>Purpose: Clears the bad word cache and reloads active prohibited word rules from the database to apply updates immediately.</p> <p>Parameters: CancellationToken ct = default.</p>
06	GetCachedBadWordsAsync() 	<p>Return type: Task<IEnumerable<BadWord>></p> <p>Purpose: Retrieves active prohibited word rules from cache if available; otherwise loads from database and caches results with both absolute and sliding expiration to optimize performance.</p> <p>Parameters: CancellationToken ct = default.</p>

4.4.40 BadWordManagementService

No	Name	Description
Attributes		
01	_badWordRepository	<p>Visibility: private</p> <p>Type: IBadWordRepository</p> <p>Purpose: Provides CRUD operations for BadWord entities, allowing administrators to manage prohibited word rules.</p>
02	_badWordService	<p>Visibility: private</p> <p>Type: IBadWordService</p>

		Purpose: Coordinates with runtime moderation service to reload cache after administrative changes.
Methods/Operations		
01	BadWordManagementService()	<p>Return type: Constructor</p> <p>Purpose: Initializes BadWordManagementService with bad word repository and bad word runtime service via dependency injection.</p> <p>Parameters: IBadWordRepository badWordRepository, IBadWordService badWordService.</p>
02	GetAllBadWordsAsync()	<p>Return type: Task<IEnumerable<BadWordDto>></p> <p>Purpose: Retrieves all bad words for administrative listing and management, regardless of activation state.</p> <p>Parameters: CancellationToken ct = default.</p>
03	GetBadWordByIdAsync()	<p>Return type: Task<BadWordDto?></p> <p>Purpose: Retrieves detailed information of a specific bad word by its ID.</p> <p>Parameters: int badWordId, CancellationToken ct = default.</p>
04	CreateBadWordAsync()	<p>Return type: Task<BadWordDto></p> <p>Purpose: Creates a new bad word rule after validating input (non-empty word, valid severity level), persists it, and reloads runtime cache to apply changes immediately.</p> <p>Parameters: CreateBadWordRequest request, CancellationToken ct = default.</p>

05	UpdateBadWordAs ync()	<p>Return type: Task<BadWordDto></p> <p>Purpose: Updates an existing bad word rule (word, regex flag, severity level, category, activation status), persists changes, and reloads runtime cache.</p> <p>Parameters: int badWordId, UpdateBadWordRequest request, CancellationToken ct = default.</p>
06	DeleteBadWordAsy nc()	<p>Return type: Task<bool></p> <p>Purpose: Deletes a bad word rule from the system and reloads runtime cache to ensure moderation rules are updated immediately.</p> <p>Parameters: int badWordId, CancellationToken ct = default.</p>
07	ReloadCacheAsync ()	<p>Return type: Task</p> <p>Purpose: Manually triggers reload of the bad word cache through the runtime moderation service.</p> <p>Parameters: CancellationToken ct = default.</p>

4.4.41 PetImageAnalysisService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Provides database access to Attributes/AttributeOptions, Pets, and PetCharacteristics for enrichment and persistence of analysis results.</p>
02	_configuration	<p>Visibility: private</p> <p>Type: IConfiguration</p>

		Purpose: Reads application settings such as Gemini API key and related AI configuration values.
03	_httpClient	<p>Visibility: private</p> <p>Type: HttpClient</p> <p>Purpose: Sends HTTP requests to Google Gemini Vision API for image analysis (single/multi-image).</p>
Methods/Operations		
01	PetImageAnalysisService()	<p>Return type: Constructor</p> <p>Purpose: Initializes PetImageAnalysisService with database context, configuration, and HttpClient via dependency dependency injection.</p> <p>Parameters: PawnderDbContext context, IConfiguration configuration, HttpClient httpClient.</p>
02	AnalyzeImageAsync()	<p>Return type: Task<PetImageAnalysisResponse></p> <p>Purpose: Validates single uploaded image, converts to Base64, loads attributes from DB, builds AI prompt, calls Gemini Vision API, parses JSON result, enriches with database IDs, and returns analysis response.</p> <p>Parameters: IFormFile image.</p>
03	AnalyzeImagesAsync()	<p>Return type: Task<PetImageAnalysisResponse></p> <p>Purpose: Validates multiple images (max 4), converts all to Base64, loads attributes from DB, builds prompt, calls Gemini Vision API (multi-image), parses JSON result, enriches with database IDs, and returns analysis response.</p> <p>Parameters: List<IFormFile> images.</p>

04	BuildAnalysisPrompt()	<p>Return type: string</p> <p>Purpose: Builds a strict instruction prompt for Gemini: only accept cat images, pick first cat image, return exactly one JSON object with isCat + attributes list; includes attribute definitions/options/units from DB.</p> <p>Parameters: List<BE.Models.Attribute> attributes.</p>
05	CallGeminiVisionAPI()	<p>Return type: Task<List<AttributeAnalysisResult>?></p> <p>Purpose: Calls Gemini Vision API with one image + prompt, validates API key, builds request body, handles common API errors (region/key/quota), extracts JSON from response, checks isCat flag, and returns parsed attributes.</p> <p>Parameters: string base64Image, string prompt, string contentType.</p>
06	CallGeminiVisionAPIMultiple()	<p>Return type: Task<List<AttributeAnalysisResult>?></p> <p>Purpose: Calls Gemini Vision API with multiple images + prompt, extracts JSON response, validates isCat, parses attributes; includes JSON error handling for invalid AI output.</p> <p>Parameters: List<(string base64, string mimeType)> images, string prompt.</p>
07	ParseAttributeResults()	<p>Return type: List<AttributeAnalysisResult></p> <p>Purpose: Parses the AI-returned JSON array into AttributeAnalysisResult list; supports optionName and numeric value (int,double/string-to-int conversion).</p> <p>Parameters: string jsonText.</p>
08	EnrichWithDatabaseIds()	Return type: Task

		<p>Purpose: Enriches analysis results by mapping attributeName → AttributId and optionName → OptionId using database lookup (Attributes + AttributeOptions).</p> <p>Parameters: List<AttributeAnalysisResult> results.</p>
09	GenerateSqlInsertScript()	<p>Return type: Task<string></p> <p>Purpose: Generates SQL INSERT statements into "PetCharacteristic" for each detected attribute (Option-based or Value-based). Supports petId provided or placeholder query by pet name if petId = 0.</p> <p>Parameters: int petId, List<AttributeAnalysisResult> attributes.</p>
10	InsertPetCharacteristicsAsync()	<p>Return type: Task<bool></p> <p>Purpose: Persists analysis results into PetCharacteristics: checks pet existence, upserts characteristics by (PetId, AttributId), updates OptionId/Value, saves changes. Returns success/failure.</p> <p>Parameters: int petId, List<AttributeAnalysisResult> attributes.</p>

4.4.42 GeminiAIService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database access for ChatAi and ChatAicontent entities (create session, store messages, load history).</p>
02	_configuration	<p>Visibility: private</p> <p>Type: IConfiguration</p>

		Purpose: Reads Gemini API key and related configuration values.
03	_googleAI	<p>Visibility: private</p> <p>Type: GoogleAI (Mscc.GenerativeAI)</p> <p>Purpose: Gemini SDK client used to create GenerativeModel and generate content responses.</p>
Methods/Operations		
01	GeminiAIService()	<p>Return type: Constructor</p> <p>Purpose: Initializes GeminiAIService with DbContext + Configuration, reads API key, and creates GoogleAI client.</p> <p>Parameters: PawnderDatabaseContext context, IConfiguration configuration.</p>
02	GetCatCareSystemPrompt()	<p>Return type: string</p> <p>Purpose: Returns fixed system prompt defining Pawnder AI role (cat care assistant), answer rules (Vietnamese, 80–150 words, bullet points), safety notes for serious symptoms, and forbidden markdown formatting.</p> <p>Parameters: None.</p>
03	CreateChatSessionAsync()	<p>Return type: Task<ChatAi></p> <p>Purpose: Creates a chat session record (ChatAi) for a user, sets default title when missing, and saves to DB.</p> <p>Parameters: int userId, string title.</p>
04	SendMessageAsyn c()	<p>Return type: Task<GeminiResponse></p> <p>Purpose: Validates chat session ownership, loads chat history, builds prompt (system prompt + recent Q&A + current</p>

		<p>question), calls Gemini model (gemini-2.5-flash) with 60s timeout, captures token usage, saves Q&A to DB, updates chat updated time, and auto-generates title for first message.</p> <p>Parameters: int userId, int chatAild, string question.</p>
05	GetChatHistoryAsync()	<p>Return type: Task<List<ChatAicontent>></p> <p>Purpose: Retrieves all messages in a chat session ordered by CreatedAt.</p> <p>Parameters: int chatAild.</p>
06	GenerateChatTitle()	<p>Return type: string</p> <p>Purpose: Generates an automatic chat title from the first question (max 50 characters, adds “...” if truncated).</p> <p>Parameters: string firstQuestion.</p>

4.4.43 DistanceService

No	Name	Description
Attributes		
01	_context	<p>Visibility: private</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Accesses User and Address data from the database to retrieve latitude and longitude for distance calculation.</p>
Methods/Operations		
01	DistanceService()	<p>Return type: Constructor</p> <p>Purpose: Initializes the DistanceService with the application database context.</p>

		<p>Parameters:</p> <ul style="list-style-type: none"> • PawnderDatabaseContext context
02	GetDistanceBetweenUsersAsync	<p>Return type: Task<double?></p> <p>Purpose: Calculates the distance between two users (in kilometers) by loading their Address information from the database. Returns <code>null</code> if either user does not have an address or missing latitude/longitude data.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • int userId1 – First user ID • int? userId2 – Second user ID
03	CalculateDistanceKm	<p>Return type: double</p> <p>Purpose: Computes the geographical distance between two GPS coordinates using the Haversine formula. The result is returned in kilometers.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • double lat1 • double lon1 • double lat2 • double lon2
04	ToRadians	<p>Return type: double</p> <p>Purpose: Converts degrees to radians to support trigonometric calculations required by the Haversine formula.</p> <p>Parameters:</p>

		• double deg
--	--	--------------

4.4.44 PasswordService

No	Name	Description
Attributes		
01	PasswordComplexityRegex	<p>Visibility: private static readonly Type: Regex</p> <p>Purpose: Regular expression used to validate password strength, requiring at least one uppercase letter, one lowercase letter, one digit, one special character, and a length between 8 and 100 characters.</p>
02	EmailFormatRegex	<p>Visibility: private static readonly Type: Regex</p> <p>Purpose: Regular expression used to validate email format (email@domain.tld). The validation is case-insensitive.</p>
Methods/Operations		
01	ValidatePasswordComplexity	<p>Return type: (bool IsValid, string? ErrorMessage) Parameters: string password.</p>
02	ValidateEmailFormat	<p>Return type: (bool IsValid, string? ErrorMessage) Parameters: string email.</p>
03	HashPassword	Return type: string

		<p>Purpose: Hashes a password using BCrypt, which is the recommended and secure approach for storing passwords.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string password
04	VerifyPassword	<p>Return type: bool</p> <p>Purpose: Verifies a password against its stored hash. Supports both BCrypt (new standard) and SHA256 (legacy) hashes for backward compatibility. If the hash starts with \$2a\$, \$2b\$, or \$2y\$, BCrypt verification is used; otherwise, SHA256 verification is applied.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string inputPassword • string hashedPassword
05	IsLegacyHash	<p>Return type: bool</p> <p>Purpose: Determines whether a given password hash is a legacy SHA256 hash (does not start with \$2 and has a length of 64 hexadecimal characters).</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string hashedPassword
06	HashPasswordSHA256	<p>Return type: string</p> <p>Purpose: Generates a SHA256 hash for backward compatibility with legacy user accounts. This method is not recommended for new passwords.</p> <p>Visibility: private</p>

		<p>Parameters:</p> <ul style="list-style-type: none"> • string password <p>Parameters: string password.</p>
--	--	--

4.4.45 TokenService

No	Name	Description
Attributes		
01	_secret	<p>Visibility: private readonly</p> <p>Type: string</p> <p>Purpose: Secret key used to sign and validate JWTs using the HMAC SHA256 algorithm.</p>
02	_issuer	<p>Visibility: private readonly</p> <p>Type: string</p> <p>Purpose: The issuer of the JWT, used during token validation to ensure authenticity.</p>
03	_audience	<p>Visibility: private readonly</p> <p>Type: string</p> <p>Purpose: The intended audience of the JWT, used to validate that the token is issued for the correct recipient.</p>
04	_accessTokenExpirationMinutes	<p>Visibility: private readonly</p> <p>Type: int</p> <p>Purpose: Expiration time of the Access Token, measured in minutes.</p>
05	_refreshTokenExpirationDays	<p>Visibility: private readonly</p> <p>Type: int</p>

		Purpose: Expiration time of the Refresh Token , measured in days.
Methods/Operations		
01	GenerateAccessToken	<p>Return type: string</p> <p>Purpose: Generates a short-lived JWT Access Token used for authenticating users when calling protected APIs. The token payload includes UserId, Role, and JTI (unique token identifier).</p> <p>Expiration: <code>_accessTokenExpirationMinutes</code></p> <p>Parameters:</p> <ul style="list-style-type: none"> • int userId • string role
02	GenerateRefreshToken	<p>Return type: string</p> <p>Purpose: Generates a long-lived JWT Refresh Token, which is used to issue a new Access Token when the current one expires. Includes an additional claim <code>token_type = refresh</code> to distinguish it from Access Tokens.</p> <p>Expiration: <code>_refreshTokenExpirationDays</code></p> <p>Parameters:</p> <ul style="list-style-type: none"> • int userId • string role
03	GetPrincipalFromToken	<p>Return type: <code>ClaimsPrincipal?</code></p> <p>Purpose: Validates a given JWT (Access Token or Refresh Token) and extracts the associated ClaimsPrincipal. Supports</p>

		<p>optional lifetime validation and only accepts tokens signed with the HmacSha256 algorithm.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string token • bool validateLifetime = true
--	--	--

4.4.46 EmailService

No	Name	Description
Attributes		
01	_gmailService	<p>Visibility: private readonly</p> <p>Type: GmailOAuth2Service</p> <p>Purpose: Service responsible for sending emails via the Gmail API using OAuth2 authentication.</p>
02	_settings	<p>Visibility: private readonly</p> <p>Type: EmailSettings</p> <p>Purpose: Stores sender email configuration, including Sender Name and Sender Email Address.</p>
Methods/Operations		
01	SendEmailAsync	<p>Return type: Task</p> <p>Purpose: Sends an email to a specified recipient using Gmail OAuth2. The email content supports both HTML and plain text, including recipient address, subject, and body.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string toEmail • string subject

		• string body
--	--	---------------

4.4.47 GmailOAuth2Service

No	Name	Description
Attributes		
01	_settings	<p>Visibility: private readonly</p> <p>Type: GmailOAuth2Settings</p> <p>Purpose: Stores OAuth2 configuration for connecting to Gmail API, including ClientId, ClientSecret, AccessToken, and RefreshToken.</p>
02	_emailSettings	<p>Visibility: private readonly</p> <p>Type: EmailSettings</p> <p>Purpose: Stores sender email configuration (SenderName and SenderEmail) used to create Gmail OAuth credentials.</p>
03	_logger	<p>Visibility: private readonly</p> <p>Type: ILogger<GmailOAuth2Service>?</p> <p>Purpose: Logs service initialization, OAuth token refresh events, and email sending errors for monitoring and debugging.</p>
04	_gmailService	<p>Visibility: private</p> <p>Type: GmailService?</p> <p>Purpose: Cached Gmail API client instance, initialized once and reused to reduce overhead and improve performance.</p>
Methods/Operations		
01	GetGmailServiceA sync	Return type: Task<GmailService>

		<p>Purpose: Initializes and returns a GmailService instance. If an instance already exists, it is reused. Automatically refreshes the OAuth2 access token when necessary.</p>
02	SendEmailAsync	<p>Return type: Task</p> <p>Purpose: Sends an email via Gmail API by creating an RFC 2822-compliant message, encoding it in Base64URL format, and invoking the Gmail Users.Messages.Send endpoint.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string toEmail – Recipient email address • string subject – Email subject • string body – Email content (HTML) • string fromEmail – Sender email address • string fromName – Sender display name
03	CreateMessage	<p>Return type: Google.Apis.Gmail.v1.Data.Message</p> <p>Purpose: Builds a raw email message following the RFC 2822 standard, sets required headers (From, To, Subject, Date), configures MIME type (HTML, UTF-8), and encodes the message using Base64URL for Gmail API compatibility.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string toEmail • string subject • string body • string fromEmail • string fromName

04	EncodeHeaderValue	<p>Return type: string</p> <p>Purpose: Encodes email header values (e.g., Subject) using MIME encoded-word (UTF-8 Base64) when special or Unicode characters are present; otherwise returns the original value unchanged.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • string value
----	-------------------	---

4.4.48 sePayService

No	Name	Description
Attributes		
01	_client	<p>Visibility: private</p> <p>Type: HttpClient</p> <p>Purpose: Stores the HTTP client used to call SePay APIs with a configured timeout and request headers.</p>
02	_config	<p>Visibility: private</p> <p>Type: IConfiguration</p> <p>Purpose: Stores application configuration to retrieve SePay settings such as ApiUrl, ApiKey, AccountNumber, and Limit.</p>
Methods/Operations		
01	sePayService(IConfiguration config)	<p>Visibility: public</p> <p>Type: Constructor</p> <p>Purpose: Initializes the service with configuration dependency and creates an HttpClient instance with a 15-second timeout.</p>

02	GetTransactionsAs ync()	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Retrieves a list of transactions from SePay using the configured account number and limit, applies Bearer token authorization, and processes the API response.</p>
03	GetTransactionDet ailsAsync(string transactionId)	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Retrieves detailed information of a specific transaction by transaction ID, validates input, calls the SePay detail API, and outputs formatted JSON data.</p>
04	CallWithRetryAsyn c(string url, int maxRetries)	<p>Visibility: private</p> <p>Type: Task<HttpResponseMessage></p> <p>Purpose: Implements retry logic for SePay API calls, retries on connection failures or non-success responses with incremental delays, and throws an exception if all retries fail.</p>

4.4.49 VietQRService

No	Name	Description
Attributes		
01	ApiUrl	<p>Visibility: private (constant)</p> <p>Type: string</p> <p>Purpose: Stores the VietQR API endpoint used to generate payment QR codes.</p>
02	ClientId	<p>Visibility: private (constant)</p> <p>Type: string</p>

		Purpose: Stores the client identifier required for VietQR API authentication.
03	ApiKey	<p>Visibility: private (constant)</p> <p>Type: string</p> <p>Purpose: Stores the API key used to authenticate requests to the VietQR service.</p>
Methods/Operations		
01	GenerateQRAsync(VietQRRequest request)	<p>Visibility: public</p> <p>Type: Task<VietQRResponse></p> <p>Purpose: Sends a POST request to the VietQR API to generate a payment QR code using provided account information, amount, and description, then returns the deserialized QR response.</p>

4.4.50 KickboxClient

No	Name	Description
Attributes		
01	ApiKey	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the API key used to authenticate requests to the Kickbox email verification service.</p>
02	Endpoint	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the base API endpoint URL for Kickbox email verification.</p>
03	_settings	Visibility: private

		<p>Type: KickboxSettings</p> <p>Purpose: Stores configuration settings required to access the Kickbox API, including API key and endpoint.</p>
04	_httpClient	<p>Visibility: private</p> <p>Type: HttpClient</p> <p>Purpose: Stores the HTTP client used to send requests to the Kickbox email verification API.</p>
05	Result	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the overall email verification result returned by the Kickbox service.</p>
06	Reason	<p>Visibility: public</p> <p>Type: string</p> <p>Purpose: Stores the reason explaining the email verification result provided by Kickbox.</p>
Methods/Operations		
01	KickboxClient(IOptions<KickboxSettings> settings, HttpClient httpClient)	<p>Visibility: public</p> <p>Type: Constructor</p> <p>Purpose: Initializes the KickboxClient service with injected configuration settings and an HttpClient instance.</p>
02	VerifyEmailAsync(string email)	<p>Visibility: public</p> <p>Type: Task<KickboxResponse></p> <p>Purpose: Sends a request to the Kickbox API to verify the specified email address, ensures a successful response, and returns the email verification result.</p>

4.4.51 ChatHub

No	Name	Description
Attributes		
01	UserConnections	<p>Visibility: public (static, readonly)</p> <p>Type: ConcurrentDictionary<int, HashSet<string>></p> <p>Purpose: Tracks active SignalR connections per user (UserId → list of ConnectionIds) to support multi-device sessions and direct user notifications.</p>
02	OnlineUsers	<p>Visibility: private (static, readonly)</p> <p>Type: ConcurrentDictionary<int, DateTime></p> <p>Purpose: Tracks online users and stores the last active timestamp (UTC) to support presence features (online/offline status).</p>
Methods/Operations		
01	OnConnectedAsync()	<p>Visibility: public (override)</p> <p>Type: Task</p> <p>Purpose: Handles client connection events, logs the connected ConnectionId, and initializes base hub connection flow.</p>
02	OnDisconnectedAsync(Exception? exception)	<p>Visibility: public (override)</p> <p>Type: Task</p> <p>Purpose: Removes the disconnected ConnectionId from all tracked users; if a user has no remaining connections, marks them offline and broadcasts a "UserOffline" event to all clients.</p>
03	RegisterUser(int userId)	Visibility: public

		<p>Type: Task</p> <p>Purpose: Registers the current ConnectionId under the specified userId, marks the user as online, and notifies other clients that the user is online via "UserOnline".</p>
04	JoinChat(int matchId, int userId)	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Adds the current connection to a match chat group (Match_{matchId}) and notifies other members in the group that the user joined via "UserJoinedChat".</p>
05	LeaveChat(int matchId, int userId)	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Removes the current connection from a match chat group (Match_{matchId}) and notifies other members in the group that the user left via "UserLeftChat".</p>
06	JoinExpertChat(int chatExpertId, int userId)	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Adds the current connection to an expert chat group (ExpertChat_{chatExpertId}) to enable real-time expert messaging.</p>
07	LeaveExpertChat(int chatExpertId, int userId)	<p>Visibility: public</p> <p>Type: Task</p> <p>Purpose: Removes the current connection from an expert chat group (ExpertChat_{chatExpertId}) when the user exits the expert chat.</p>

08	<code>SendExpertMessage(int chatExpertId, int fromId, string message)</code>	<p>Visibility: public Type: Task</p> <p>Purpose: Sends a real-time expert message to all connections in an expert chat group using "ReceiveExpertMessage" event, including message metadata (chatExpertId, sender, UTC timestamp).</p>
09	<code>SendMessage(int matchId, int fromUserId, string message, int? fromPetId = null)</code>	<p>Visibility: public Type: Task</p> <p>Purpose: Sends a real-time message to all users in the match group (Match_{matchId}) using "ReceiveMessage", including sender info, optional petId, and UTC timestamp.</p>
10	<code>Typing(int matchId, int userId, bool isTyping)</code>	<p>Visibility: public Type: Task</p> <p>Purpose: Broadcasts typing status to other users in the match group via "UserTyping" to support typing indicators.</p>
11	<code>MarkAsRead(int matchId, int userId)</code>	<p>Visibility: public Type: Task</p> <p>Purpose: Notifies other users in the match group that messages were read via "MessagesRead", including who read and UTC timestamp.</p>
12	<code>IsUserOnline(int userId)</code>	<p>Visibility: public Type: bool</p> <p>Purpose: Checks whether a user is currently marked as online by verifying presence in OnlineUsers.</p>

13	GetOnlineUsers()	<p>Visibility: public</p> <p>Type: Task<List<int>></p> <p>Purpose: Returns a list of all currently online user IDs based on OnlineUsers keys.</p>
14	SendNewMessageBadge(IHubContext<ChatHub> hubContext, int toUserId, int matchId, int? fromPetId = null, int? toPetId = null)	<p>Visibility: public (static)</p> <p>Type: Task</p> <p>Purpose: Sends a "NewMessageBadge" notification to a specific user across all their active connections, including matchId, optional pet metadata, and UTC timestamp.</p>
15	SendNewLikeBadge(IHubContext<ChatHub> hubContext, int toUserId, int fromUserId)	<p>Visibility: public (static)</p> <p>Type: Task</p> <p>Purpose: Sends a "NewLikeBadge" notification to a specific user across all active connections, including liker userId and UTC timestamp.</p>
16	SendNewExpertMessageBadge(IHubContext<ChatHub> hubContext, int toUserId, int chatExpertId)	<p>Visibility: public (static)</p> <p>Type: Task</p> <p>Purpose: Sends a "NewExpertMessageBadge" notification to a user for a specific expert chat, or logs offline status if the user has no active connections.</p>
17	SendMatchNotification(IHubContext<ChatHub> hubContext, int toUserId, string otherUserName, int otherUserId,	<p>Visibility: public (static)</p> <p>Type: Task</p>

	<pre>int matchId, string? petName, string? petPhotoUrl)</pre>	Purpose: Sends a real-time match success notification ("MatchSuccess") to a specific user, including match metadata and an in-app message payload.
18	<pre>SendNotification(IHubContext<ChatHub> hubContext, int toUserId, string title, string message, string type = "system")</pre>	Visibility: public (static) Type: Task Purpose: Sends a general notification ("NewNotification") to a specific user across all active connections, including title, message, type, and UTC timestamp.
19	<pre>SendMatchDeleteNotification(IHubContext<ChatHub> hubContext, int toUserId, int matchId)</pre>	Visibility: public (static) Type: Task Purpose: Sends a real-time notification ("MatchDeleted") to inform a user that a match/unmatch action occurred, including matchId and UTC timestamp.
20	<pre>SendNotificationWithMetadata(IHubContext<ChatHub> hubContext, int toUserId, string title, string message, string type, int? expertId = null, int? chatId = null)</pre>	Visibility: public (static) Type: Task Purpose: Sends a notification ("NewNotification") with additional metadata (expertId, chatId) to support expert confirmations and contextual in-app alerts.
21	<pre>SendExpertMessage(IHubContext<ChatHub> hubContext, int chatExpertId, int fromId, string</pre>	Visibility: public (static) Type: Task Purpose: Sends an expert chat message to the expert chat group (ExpertChat_{chatExpertId}) via

	message, int? toUserId = null)	"ReceiveExpertMessage" event to avoid duplicates and support reconnection scenarios.
--	--------------------------------	--

4.4.52 AppointmentExpirationBackgroundService

No	Name	Description
Attributes		
01	_serviceProvider	<p>Visibility: private (readonly) Type: IServiceProvider</p> <p>Purpose: Provides access to dependency injection scope for resolving required services (e.g., IAppointmentService) during background execution.</p>
02	_logger	<p>Visibility: private (readonly) Type: ILogger<AppointmentExpirationBackgroundService></p> <p>Purpose: Stores the logger instance used to record service lifecycle events and error/debug information.</p>
03	_checkInterval	<p>Visibility: private (readonly) Type: TimeSpan</p> <p>Purpose: Defines the interval for periodic checks of expired appointments; default is 5 minutes.</p>
Methods/Operations		
01	AppointmentExpirationBackgroundService(IServiceProvider serviceProvider, ILogger<AppointmentExpirationBackgroundService> logger)	<p>Visibility: public Type: Constructor</p> <p>Purpose: Initializes the background service with dependency injection provider and logger for scheduled processing tasks.</p>

02	ExecuteAsync(CancellationToken stoppingToken)	<p>Visibility: protected (override) Type: Task Purpose: Runs the background loop until cancellation; periodically triggers expired appointment processing, handles exceptions, logs status, and delays based on the configured check interval.</p>
03	ProcessExpiredAppointmentsAsync(CancellationToken ct)	<p>Visibility: private Type: Task Purpose: Creates a scoped service provider, resolves IAppointmentService, invokes expiration processing logic, and logs execution time for monitoring.</p>

4.4.53 EventCompletionBackgroundService

No	Name	Description
Attributes		
01	_serviceProvider	<p>Visibility: private (readonly) Type: IServiceProvider Purpose: Provides access to dependency injection scope for resolving required services (e.g., IEventService) during background execution.</p>
02	_logger	<p>Visibility: private (readonly) Type: ILogger<EventCompletionBackgroundService> Purpose: Stores the logger instance used to record service lifecycle events, execution status, and error information.</p>
03	_interval	<p>Visibility: private (readonly) Type: TimeSpan</p>

		Purpose: Defines the interval for periodic event status checks; default is 1 minute.
Methods/Operations		
01	EventCompletionBackgroundService(IServiceProvider serviceProvider, ILogger<EventCompletionBackgroundService> logger)	<p>Visibility: public Type: Constructor</p> <p>Purpose: Initializes the background service with dependency injection provider and logger to support scheduled event state processing.</p>
02	ExecuteAsync(CancellationToken stoppingToken)	<p>Visibility: protected (override) Type: Task</p> <p>Purpose: Runs a continuous background loop until cancellation; periodically processes event state transitions, handles exceptions, logs service lifecycle events, and delays execution based on the configured interval.</p>
03	ProcessEventsAsync(CancellationToken ct)	<p>Visibility: private Type: Task</p> <p>Purpose: Creates a scoped service provider, resolves IEventService, and invokes business logic to process event state transitions and finalize results when required.</p>

4.4.54 PaymentExpirationBackgroundService

No	Name	Description
Attributes		
01	_logger	<p>Visibility: private (readonly) Type: ILogger<PaymentExpirationBackgroundService></p>

		Purpose: Stores the logger instance used to record lifecycle events, expiration check results, and error information during background processing.
02	_serviceProvider	<p>Visibility: private (readonly)</p> <p>Type: IServiceProvider</p> <p>Purpose: Provides access to dependency injection scope for resolving database context and required services during scheduled execution.</p>
03	_checkInterval	<p>Visibility: private (readonly)</p> <p>Type: TimeSpan</p> <p>Purpose: Defines the periodic interval to check expired payments; default is 1 hour.</p>
Methods/Operations		
01	PaymentExpirationBackgroundService (ILogger<PaymentExpirationBackgroundService> logger, IServiceProvider serviceProvider)	<p>Visibility: public</p> <p>Type: Constructor</p> <p>Purpose: Initializes the background service with logger and dependency injection provider to enable scheduled expiration processing.</p>
02	ExecuteAsync(CancellationToken stoppingToken)	<p>Visibility: protected (override)</p> <p>Type: Task</p> <p>Purpose: Runs a continuous background loop until cancellation; periodically checks and updates expired payment records, delays by the configured interval (1 hour), and retries after 5 minutes if an error occurs.</p>

03	CheckAndUpdateExpiredPaymentsAsync(CancellationToken ct)	<p>Visibility: private Type: Task</p> <p>Purpose: Creates a scoped service provider, resolves PawnderDbContext, queries expired payment histories (EndDate < today and StatusService = "active"), updates StatusService to "pending", sets UpdatedAt timestamp, and saves changes to the database.</p>
----	--	---

4.5 Repository

4.5.1 AttributeOptionRepository

No	Name	Description
Attributes		
01	_dbSet	<p>Visibility: protected (inherited) Type: DbSet<AttributeOption></p> <p>Purpose: Represents the AttributeOption table in the database and provides query access through Entity Framework.</p>
02	_context	<p>Visibility: protected (inherited) Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to access and manage AttributeOption entities.</p>
Methods/Operations		
01	GetAllOptionsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<OptionResponse>></p>

		<p>Purpose: Retrieves all non-deleted attribute options and maps them to OptionResponse DTOs, including option ID, attribute ID, name and deletion status.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetOptionsByAttributeIdAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all non-deleted options associated with a specific attribute ID and returns basic option information for client consumption.</p> <p>Parameters: int attributeId, CancellationToken ct = default.</p>

4.5.2 AttributeRepository

No	Name	Description
Attributes		
01	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<AttributeEntity></p> <p>Purpose: Represents the Attribute table in the database and provides query access through Entity Framework.</p>
02	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to access and manage Attribute entities and related data.</p>
Methods/Operations		
01	GetAttributesAsync	Visibility: public async

		<p>Return type: Task<PagedResult<AttributeResponse>></p> <p>Purpose: Retrieves a paginated list of attributes with optional search and soft-delete filtering; supports including or excluding deleted options and maps results to AttributeResponse DTOs with nested option responses.</p> <p>Parameters: string? search, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.</p>
02	GetAttributeByIdA sync	<p>Visibility: public async</p> <p>Return type: Task<AttributeEntity?></p> <p>Purpose: Retrieves a single attribute by its ID using a no-tracking query; returns null if the attribute does not exist.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
03	NameExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether an attribute name already exists among non-deleted attributes, with an option to exclude a specific attribute ID (useful for uniqueness validation during update).</p> <p>Parameters: string name, int? excludeId = null, CancellationToken ct = default.</p>
04	GetAttributesForFi IterAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all active attributes (non-deleted) with their non-deleted options, ordered by AttributId, and returns lightweight objects used for building filter UIs (including id, name, type, unit, percent and options).</p>

		Parameters: CancellationToken ct = default.
--	--	---

4.5.3 BaseRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected readonly</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context used to manage entity persistence and execute save operations.</p>
02	_dbSet	<p>Visibility: protected readonly</p> <p>Type: DbSet<T></p> <p>Purpose: Entity Framework DbSet representing the table corresponding to the generic entity type T.</p>
Methods/Operations		
01	GetByIdAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<T?></p> <p>Purpose: Retrieves an entity by its primary key using FindAsync; returns null if the entity does not exist.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
02	GetAllAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<IEnumerable<T>></p> <p>Purpose: Retrieves all entities of type T from the database.</p> <p>Parameters: CancellationToken ct = default.</p>

03	FindAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<IEnumerable<T>></p> <p>Purpose: Retrieves all entities that satisfy the specified predicate expression.</p> <p>Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.</p>
04	FirstOrDefaultAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<T?></p> <p>Purpose: Retrieves the first entity that matches the given predicate or returns null if none is found.</p> <p>Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.</p>
05	AddAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<T></p> <p>Purpose: Adds a new entity to the DbSet and immediately saves changes to the database.</p> <p>Parameters: T entity, CancellationToken ct = default.</p>
06	AddRangeAsync	<p>Visibility: public virtual async</p> <p>Return type: Task</p> <p>Purpose: Adds multiple entities to the DbSet and saves all changes in a single operation.</p> <p>Parameters: IEnumerable<T> entities, CancellationToken ct = default.</p>
07	UpdateAsync	<p>Visibility: public virtual async</p>

		<p>Return type: Task</p> <p>Purpose: Updates an existing entity and persists the changes to the database.</p> <p>Parameters: T entity, CancellationToken ct = default.</p>
08	UpdateRangeAsync	<p>Visibility: public virtual async</p> <p>Return type: Task</p> <p>Purpose: Updates multiple entities and persists all changes to the database.</p> <p>Parameters: IEnumerable<T> entities, CancellationToken ct = default.</p>
09	DeleteAsync	<p>Visibility: public virtual async</p> <p>Return type: Task</p> <p>Purpose: Permanently deletes an entity from the database and saves the change.</p> <p>Parameters: T entity, CancellationToken ct = default.</p>
10	DeleteRangeAsync	<p>Visibility: public virtual async</p> <p>Return type: Task</p> <p>Purpose: Permanently deletes multiple entities from the database and saves the changes.</p> <p>Parameters: IEnumerable<T> entities, CancellationToken ct = default.</p>
11	ExistsAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<bool></p>

		<p>Purpose: Checks whether any entity matching the given predicate exists in the database.</p> <p>Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.</p>
12	CountAsync	<p>Visibility: public virtual async</p> <p>Return type: Task<int></p> <p>Purpose: Returns the total number of entities, optionally filtered by a predicate expression.</p> <p>Parameters: Expression<Func<T, bool>>? predicate = null, CancellationToken ct = default.</p>
13	SaveChangesAsync	<p>Visibility: public virtual async</p> <p>Return type: Task</p> <p>Purpose: Persists all pending changes in the DbContext to the database.</p> <p>Parameters: CancellationToken ct = default.</p>

4.5.4 BlockRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage Block entities and related user data.</p>
02	_dbSet	Visibility: protected (inherited)

		<p>Type: DbSet<Block></p> <p>Purpose: Represents the Block table in the database and provides query access for block relationships between users.</p>
Methods/Operations		
01	GetBlockedUsersA sync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all users blocked by a specific user, including basic information of the blocked users (full name, email) and the block creation timestamp.</p> <p>Parameters: int fromUserId, CancellationToken ct = default.</p>
02	GetBlockAsync	<p>Visibility: public async</p> <p>Return type: Task<Block?></p> <p>Purpose: Retrieves a specific block record between two users (blocker and blocked); returns null if no block exists.</p> <p>Parameters: int fromUserId, int toUserId, CancellationToken ct = default.</p>
03	BlockExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether a block relationship already exists between two users.</p> <p>Parameters: int fromUserId, int toUserId, CancellationToken ct = default.</p>

4.5.5 ChatExpertContentRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to access ChatExpertContent entities and related ChatExpert data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<ChatExpertContent></p> <p>Purpose: Represents the ChatExpertContent table in the database and provides query access to chat messages.</p>
Methods/Operations		
01	GetChatMessages Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves the full message history for a specific expert chat, ordered by creation time, including sender information, message content, optional expert confirmation data and timestamps.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>
02	ChatExpertExistsA sync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether a chat expert session with the given ID exists in the system.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>

4.5.6 ChatExpertRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage ChatExpert entities and related user/expert data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<ChatExpert></p> <p>Purpose: Represents the ChatExpert table in the database and provides query access to expert chat sessions.</p>
Methods/Operations		
01	GetChatsByUserId Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions for a user, including expert information, last message preview, timestamps and basic chat metadata; results are ordered by the most recent message or creation time.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsByExpertId Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat sessions assigned to a specific expert, including user and expert identification details and timestamps; primarily used by the expert dashboard or admin view.</p>

		Parameters: int expertId, CancellationToken ct = default.
03	GetChatExpertByExpertAndUserAsync	<p>Visibility: public async</p> <p>Return type: Task<ChatExpert?></p> <p>Purpose: Retrieves a single chat session for a given expert and user pair; returns null if the chat does not exist.</p> <p>Parameters: int expertId, int userId, CancellationToken ct = default.</p>
04	ChatExpertExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether an expert chat session with the specified ID exists in the system.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>

4.5.7 ChatUserContentRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to access ChatUserContent and ChatUser entities.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<ChatUserContent></p>

		Purpose: Represents the ChatUserContent table in the database and provides query access to user-to-user chat messages.
Methods/Operations		
01	GetChatMessages Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat messages for a given match, ordered by creation time; prioritizes direct user sender information and falls back to pet-based sender data for backward compatibility.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
02	ChatExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether an accepted, non-deleted user-to-user chat exists for the given match ID.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>

4.5.8 ChatUserRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to access Pets and ChatUser entities for invite/chat queries.</p>

02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<ChatUser></p> <p>Purpose: Represents the ChatUser table in the database and provides query access to match/chat relationships between pets.</p>
Methods/Operations		
01	GetInvitesAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all pending chat invitations for a user by first resolving all of the user's pets, then returning matches where those pets are the target (ToPet) of a pending invite.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves accepted chats for a user, optionally filtered by a specific pet; validates that the pet belongs to the user, loads related pet and user information, and returns chat metadata including pet and owner details.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default.</p>
03	GetChatUserByPet sAsync	<p>Visibility: public async</p> <p>Return type: Task<ChatUser?></p> <p>Purpose: Retrieves a chat relationship between two pets if it exists and is not deleted.</p> <p>Parameters: int fromPetId, int toPetId, CancellationToken ct = default.</p>

04	GetChatUserByMatchIdAsync	<p>Visibility: public async</p> <p>Return type: Task<ChatUser?></p> <p>Purpose: Retrieves a chat record by match ID where the chat is still in Pending status (typically used when accepting/declining an invite).</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
----	---------------------------	--

4.5.9 ExpertConfirmationRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage ExpertConfirmation entities and related User, Expert and ChatAi data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<ExpertConfirmation></p> <p>Purpose: Represents the ExpertConfirmation table in the database and provides query access for expert confirmation records.</p>
Methods/Operations		
01	GetAllExpertConfirmationsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation records, including related User, Expert and ChatAi data, and maps them to ExpertConfirmationDTO objects.</p>

		Parameters: CancellationToken ct = default.
02	GetExpertConfirmationAsync	<p>Visibility: public async</p> <p>Return type: Task<ExpertConfirmation?></p> <p>Purpose: Retrieves a single expert confirmation for a specific expert, user and chat session (composite key lookup), including related User, Expert and ChatAi entities.</p> <p>Parameters: int expertId, int userId, int chatId, CancellationToken ct = default.</p>
03	GetUserExpertConfirmationsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation records for a specific user, including related Expert and ChatAi data, and maps them to ExpertConfirmationDTO objects.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
04	GetExpertConfirmationByUserAndChatAsync	<p>Visibility: public async</p> <p>Return type: Task<ExpertConfirmation?></p> <p>Purpose: Retrieves a single expert confirmation record based on userId and chatAId, without including related navigation properties.</p> <p>Parameters: int userId, int chatId, CancellationToken ct = default.</p>
05	UpdateAsync	<p>Visibility: public override async</p> <p>Return type: Task</p> <p>Purpose: Overrides the base UpdateAsync method to correctly handle ExpertConfirmation entities with a composite key</p>

		(ExpertId, UserId, ChatAId) by manually loading the existing record, updating its fields and saving changes; falls back to the base implementation if no existing record is found. Parameters: ExpertConfirmation entity, CancellationToken ct = default.
--	--	--

4.5.10 NotificationRepository

No	Name	Description
Attributes		
01	_context	Visibility: protected (inherited) Type: PawnderDatabaseContext Purpose: Database context inherited from BaseRepository, used to manage Notification entities and persist state changes (e.g. marking notifications as read).
02	_dbSet	Visibility: protected (inherited) Type: DbSet<Notification> Purpose: Represents the Notification table in the database and provides query access for user notification data.
Methods/Operations		
01	GetAllNotifications Async	Visibility: public async Return type: Task<IEnumerable<NotificationDto>> Purpose: Retrieves all notifications in the system, including associated user information, and maps them to NotificationDto objects. Parameters: CancellationToken ct = default.

02	GetNotificationByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<NotificationDto?></p> <p>Purpose: Retrieves a single notification by its ID, including user information; returns null if the notification does not exist.</p> <p>Parameters: int notificationId, CancellationToken ct = default.</p>
03	GetNotificationsByUserIdAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<Notification>></p> <p>Purpose: Retrieves all notifications for a specific user, ordered by creation time (most recent first).</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
04	MarkAllAsReadAsync	<p>Visibility: public async</p> <p>Return type: Task<int></p> <p>Purpose: Marks all unread notifications of a user as read, updates their timestamps, persists the changes and returns the number of updated notifications.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
05	GetUnreadCountAsync	<p>Visibility: public async</p> <p>Return type: Task<int></p> <p>Purpose: Counts the number of unread notifications of specific types (system and expert confirmation) for a given user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

4.5.11 PaymentHistoryRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage PaymentHistory entities and related User data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<PaymentHistory></p> <p>Purpose: Represents the PaymentHistory table in the database and provides query access for subscription and payment history records.</p>
Methods/Operations		
01	GetPaymentHistoriesByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all payment history records for a specific user, ordered by creation time (most recent first), and returns basic subscription information such as status and start/end dates.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetAllPaymentHistoriesAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all payment history records in the system, including associated user information (name and email), subscription status, amount, dates and timestamps.</p> <p>Parameters: CancellationToken ct = default.</p>

03	GetVipStatusAsync	<p>Visibility: public async</p> <p>Return type: Task<object?></p> <p>Purpose: Retrieves the current active VIP subscription of a user (if any), based on the current date and subscription status, and calculates the remaining number of days until expiration.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
----	-------------------	---

4.5.12 PetCharacteristicRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage PetCharacteristic entities and related Attribute and Option data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<PetCharacteristic></p> <p>Purpose: Represents the PetCharacteristic table in the database and provides query access for pet attribute values.</p>
Methods/Operations		
01	GetPetCharacteristicsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p>

		<p>Purpose: Retrieves all characteristics of a specific pet, including attribute metadata (name, unit, type) and selected option or numeric value.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	GetPetCharacteristicAsync	<p>Visibility: public async</p> <p>Return type: Task<PetCharacteristic?></p> <p>Purpose: Retrieves a single pet characteristic by pet ID and attribute ID, including related Attribute and Option entities; returns null if it does not exist.</p> <p>Parameters: int petId, int attributeId, CancellationToken ct = default.</p>
03	ExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether a pet characteristic already exists for a given pet and attribute.</p> <p>Parameters: int petId, int attributeId, CancellationToken ct = default.</p>

4.5.13 PetPhotoRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage PetPhoto entities and persist photo-related changes.</p>

02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<PetPhoto></p> <p>Purpose: Represents the PetPhoto table in the database and provides query access to pet image data.</p>
Methods/Operations		
01	GetPhotosByPetId Async	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<PetPhotoResponse>></p> <p>Purpose: Retrieves all non-deleted photos of a specific pet, ordered by sort order and photo ID, and maps them to PetPhotoResponse DTOs.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	GetPhotoCountBy PetIdAsync	<p>Visibility: public async</p> <p>Return type: Task<int></p> <p>Purpose: Returns the number of non-deleted photos associated with a specific pet.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
03	GetMaxSortOrder Async	<p>Visibility: public async</p> <p>Return type: Task<int?></p> <p>Purpose: Retrieves the maximum sort order value among non-deleted photos of a pet; returns null if the pet has no photos.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>

4.5.14 PetRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDbContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage Pet entities and related User, Address, Photo and Characteristic data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<Pet></p> <p>Purpose: Represents the Pet table in the database and provides query access to pet records.</p>
Methods/Operations		
01	GetPetsByUserIdA sync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<PetDto>></p> <p>Purpose: Retrieves all non-deleted pets belonging to a specific user, including their avatar image (first non-deleted photo ordered by sort order) and basic pet information, mapped to PetDto.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetPetsForMatchi ngAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves pets that are eligible for matching: active, non-deleted pets belonging to other users, excluding users already matched or blocked; includes photo list and basic owner info with address and coordinates.</p>

		Parameters: int userId, List<int> excludedUserIds, List<int> blockedUserIds, CancellationToken ct = default.
03	GetPetByIdWithDetailsAsync	<p>Visibility: public async</p> <p>Return type: Task<Pet?></p> <p>Purpose: Retrieves a single non-deleted pet by ID including photos, characteristics (with attributes) and owner information with address details.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
04	IsPetOwnerAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether a given user is the owner of a non-deleted pet.</p> <p>Parameters: int petId, int userId, CancellationToken ct = default.</p>
05	DeactivateOtherPetsAsync	<p>Visibility: public async</p> <p>Return type: Task</p> <p>Purpose: Deactivates all other pets of a user and activates only the specified pet by updating IsActive and UpdatedAt for all the user's non-deleted pets.</p> <p>Parameters: int userId, int activePetId, CancellationToken ct = default.</p>

4.5.15 ReportRepository

No	Name	Description
Attributes		

01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage Report entities and related user/content data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<Report></p> <p>Purpose: Represents the Report table in the database and provides query access for report records.</p>
Methods/Operations		
01	GetAllReportsAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<ReportDto>></p> <p>Purpose: Retrieves all reports with reporter information and reported user information, resolving the reported user by prioritizing FromUser and falling back to FromPet.User, then maps the data to ReportDto.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetReportByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<ReportDto?></p> <p>Purpose: Retrieves a single report by ID with reporter and reported user details using the same FromUser / FromPet.User priority logic, and maps it to ReportDto; returns null if not found.</p> <p>Parameters: int reportId, CancellationToken ct = default.</p>
03	GetReportsByUserIdAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<object>></p>

		<p>Purpose: Retrieves all reports created by a specific user (reporter), ordered by creation time, and returns lightweight objects including report info, content info and reported user information (via FromPet.User).</p> <p>Parameters: int userReportId, CancellationToken ct = default.</p>
--	--	---

4.5.16 UserPreferenceRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected (inherited)</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage UserPreference entities and related Attribute / Option data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<UserPreference></p> <p>Purpose: Represents the UserPreference table in the database and provides query access for user preference records.</p>
Methods/Operations		
01	GetUserPreferencesAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<UserPreferenceResponse>></p> <p>Purpose: Retrieves all preferences of a specific user, including attribute details and option information (if any), ordered by AttributeId, and maps results to UserPreferenceResponse DTOs.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

02	GetUserPreferenceAsync	<p>Visibility: public async</p> <p>Return type: Task<UserPreference?></p> <p>Purpose: Retrieves a single user preference by user ID and attribute ID, including attribute data; returns null if the preference does not exist.</p> <p>Parameters: int userId, int attributeId, CancellationToken ct = default.</p>
03	ExistsAsync	<p>Visibility: public async</p> <p>Return type: Task<bool></p> <p>Purpose: Checks whether a preference already exists for a given user and attribute combination.</p> <p>Parameters: int userId, int attributeId, CancellationToken ct = default.</p>
04	GetUserPreferencesByUserIdAsync	<p>Visibility: public async</p> <p>Return type: Task<IEnumerable<UserPreference>></p> <p>Purpose: Retrieves all UserPreference entities for a specific user without additional joins or DTO mapping, typically used for internal logic such as batch delete or upsert.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

4.5.17 UserRepository

No	Name	Description
Attributes		
01	_context	Visibility: protected (inherited)

		<p>Type: PawnderDatabaseContext</p> <p>Purpose: Database context inherited from BaseRepository, used to manage User entities and related Role data.</p>
02	_dbSet	<p>Visibility: protected (inherited)</p> <p>Type: DbSet<User></p> <p>Purpose: Represents the User table in the database and provides query access to user records.</p>
Methods/Operations		
01	GetUsersAsync	<p>Visibility: public async</p> <p>Return type: Task<PagedResult<UserResponse>></p> <p>Purpose: Retrieves a paginated list of users with optional search, role and status filters, and the option to include deleted users; automatically excludes Admin users from the result set and maps to UserResponse DTOs.</p> <p>Parameters: string? search, int? roleId, int? statusId, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.</p>
02	GetUserByIdAsync	<p>Visibility: public async</p> <p>Return type: Task<User?></p> <p>Purpose: Retrieves a single user by ID with an option to include or exclude soft-deleted users; returns null if the user does not exist.</p> <p>Parameters: int userId, bool includeDeleted = false, CancellationToken ct = default.</p>
03	EmailExistsAsync	Visibility: public async

		<p>Return type: Task<bool></p> <p>Purpose: Checks whether an active (non-deleted) user with the given email already exists in the system.</p> <p>Parameters: string email, CancellationToken ct = default.</p>
04	GetUserByEmailAs ync	<p>Visibility: public async</p> <p>Return type: Task<User?></p> <p>Purpose: Retrieves a non-deleted user by email including the related Role entity; returns null if not found.</p> <p>Parameters: string email, CancellationToken ct = default.</p>

4.5.18 AppointmentRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected</p> <p>Type: PawnderDbContext</p> <p>Purpose: Provides access to the database context for querying and persisting appointment-related data.</p>
02	_dbSet	<p>Visibility: protected</p> <p>Type: DbSet<PetAppointment></p> <p>Purpose: Represents the PetAppointment entity set used for database operations.</p>
Methods/Operations		
01	GetByMatchIdAsy nc	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p>

		<p>Purpose: Retrieves all appointments associated with a specific match, including related pets, users, and location information.</p> <p>Parameters: matchId: int, ct: CancellationToken</p>
02	GetByUserIdAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves all appointments in which the specified user participates (as inviter or invitee).</p> <p>Parameters: userId: int, ct: CancellationToken</p>
03	GetByStatusAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves appointments filtered by appointment status.</p> <p>Parameters: status: string, ct: CancellationToken</p>
04	GetByIdWithDetailsAsync	<p>Visibility: public</p> <p>Return: Task<PetAppointment?></p> <p>Purpose: Retrieves a specific appointment by ID along with full related details (match, pets with primary photos, users, and location).</p> <p>Parameters: appointmentId: int, ct: CancellationToken</p>
05	GetUpcomingAppointmentsForReminderAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves confirmed appointments occurring within a defined time window for reminder notifications.</p>

		Parameters: reminderTime: DateTime, ct: CancellationToken
06	CountMessagesBetweenUsersAsync	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Counts the number of chat messages exchanged within a specific match to support appointment precondition checks..</p> <p>Parameters: matchId: int, ct: CancellationToken</p>
07	IsPetProfileCompleteAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a pet profile is complete (has name, breed, and at least one valid photo) before allowing appointment creation.</p> <p>Parameters: petId: int, ct: CancellationToken</p>

4.5.19 AppointmentLocationRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Provides access to the database context for querying and managing appointment location data.</p>
02	_dbSet	<p>Visibility: protected</p> <p>Type: DbSet<PetAppointmentLocation></p>

		Purpose: Represents the PetAppointmentLocation entity set used for database operations.
Methods/Operations		
01	GetByCityAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointmentLocation>></p> <p>Purpose: Retrieves all pet-friendly appointment locations filtered by city name.</p> <p>Parameters: city: string, ct: CancellationToken</p>
02	GetNearbyLocationsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointmentLocation>></p> <p>Purpose: Retrieves nearby pet-friendly locations within a specified radius using a simplified Haversine-based geographic calculation.</p> <p>Parameters: latitude: decimal, longitude: decimal, radiusKm: decimal, ct: CancellationToken</p>
03	GetByGooglePlaceIdAsync	<p>Visibility: public</p> <p>Return: Task<PetAppointmentLocation?></p> <p>Purpose: Retrieves an appointment location by its Google Place ID to prevent duplicate location entries.</p> <p>Parameters: googlePlaceId: string, ct: CancellationToken</p>

4.5.20 EventRepository

No	Name	Description
Attributes		
01	_context	Visibility: protected

		<p>Type: PawnderDatabaseContext</p> <p>Purpose: Provides access to the database context for querying and managing event-related data.</p>
02	_dbSet	<p>Visibility: protected</p> <p>Type: DbSet<PetEvent></p> <p>Purpose: Represents the PetEvent entity set used for database operations.</p>
Methods/Operations		
01	GetAllEventsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves all events (admin view), including creator and non-deleted submissions.</p> <p>Parameters: ct: CancellationToken</p>
02	GetActiveEventsAs ync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events available for users (excluding cancelled), prioritizing ongoing events and including creator and non-deleted submissions.</p> <p>Parameters: ct: CancellationToken</p>
03	GetEventWithSub missionsAsync	<p>Visibility: public</p> <p>Return: Task<PetEvent?></p> <p>Purpose: Retrieves event details by ID with full submission information (user, pet, pet photos, votes) for leaderboard and event detail views.</p>

		Parameters: eventId: int, ct: CancellationToken
04	GetEventsToTransitionAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events that need automatic status transitions based on current time (upcoming→active, active→submission_closed, submission_closed→voting_ended).</p> <p>Parameters: ct: CancellationToken</p>
05	GetEventsByStatusAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events filtered by a specific status, including creator information.</p> <p>Parameters: status: string, ct: CancellationToken</p>

4.5.21 SubmissionRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected</p> <p>Type: PawnderDbContext</p> <p>Purpose: Provides access to the database context for managing event submissions and voting data.</p>
02	_dbSet	<p>Visibility: protected</p> <p>Type: DbSet<EventSubmission></p> <p>Purpose: Represents the EventSubmission entity set used for database operations.</p>

Methods/Operations		
01	GetByIdAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<EventSubmission>></p> <p>Purpose: Retrieves all non-deleted submissions of a specific event, including user and pet details, ordered by vote count.</p> <p>Parameters: eventId: int, ct: CancellationToken</p>
02	GetLeaderboardAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<EventSubmission>></p> <p>Purpose: Retrieves top-ranked submissions of an event for leaderboard display, sorted by vote count and submission time.</p> <p>Parameters: eventId: int, top: int, ct: CancellationToken</p>
03	HasUserSubmittedAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a user has already submitted an entry to a specific event.</p> <p>Parameters: eventId: int, userId: int, ct: CancellationToken</p>
04	GetByIdWithDetailsAsync	<p>Visibility: public</p> <p>Return: Task<EventSubmission?></p> <p>Purpose: Retrieves a submission by ID with full related details (event, user, pet, pet photos, and votes).</p> <p>Parameters: submissionId: int, ct: CancellationToken</p>

05	HasUserVotedAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a user has already voted for a specific submission.</p> <p>Parameters: submissionId: int, userId: int, ct: CancellationToken</p>
06	AddVoteAsync	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Adds a new vote to a submission and updates the vote count accordingly.</p> <p>Parameters: submissionId: int, userId: int, ct: CancellationToken</p>
07	RemoveVoteAsync	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Removes an existing vote from a submission and updates the vote count accordingly.</p> <p>Parameters: submissionId: int, userId: int, ct: CancellationToken</p>
08	UpdateVoteCount Async	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Recalculates and updates the vote count of a submission based on current vote records.</p> <p>Parameters: submissionId: int, ct: CancellationToken</p>

4.5.22 PolicyRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: private readonly</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Provides access to the database context for querying and managing policy, policy versions, and user acceptance audit data.</p>
Methods/Operations		
01	GetAllPoliciesAsync	<p>Visibility: public</p> <p>Return: Task<List<Policy>></p> <p>Purpose: Retrieves all non-deleted policies ordered by display order and policy code.</p> <p>Parameters: ct: CancellationToken</p>
02	GetPolicyByIdAsync	<p>Visibility: public</p> <p>Return: Task<Policy?></p> <p>Purpose: Retrieves a policy by ID including its ACTIVE versions.</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
03	GetPolicyByCodeA sync	<p>Visibility: public</p> <p>Return: Task<Policy?></p> <p>Purpose: Retrieves a policy by code including its ACTIVE versions.</p> <p>Parameters: policyCode: string, ct: CancellationToken</p>

04	CreatePolicyAsync	<p>Visibility: public</p> <p>Return: Task<Policy></p> <p>Purpose: Creates a new policy and sets CreatedAt/UpdatedAt timestamps.</p> <p>Parameters: policy: Policy, ct: CancellationToken</p>
05	UpdatePolicyAsync	<p>Visibility: public</p> <p>Return: Task<Policy></p> <p>Purpose: Updates an existing policy and sets UpdatedAt timestamp.</p> <p>Parameters: policy: Policy, ct: CancellationToken</p>
06	DeletePolicyAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Soft-deletes a policy (marks IsDeleted=true, IsActive=false) by policy ID.</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
07	GetVersionsByPolicyIdAsync	<p>Visibility: public</p> <p>Return: Task<List<PolicyVersion>></p> <p>Purpose: Retrieves all versions of a policy, including creator information, ordered by version number descending.</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
08	GetVersionByIdAsync	<p>Visibility: public</p> <p>Return: Task<PolicyVersion?></p>

		<p>Purpose: Retrieves a policy version by ID including policy and creator details.</p> <p>Parameters: policyVersionId: int, ct: CancellationToken</p>
09	GetActiveVersionByPolicyIdAsync	<p>Visibility: public</p> <p>Return: Task<PolicyVersion?></p> <p>Purpose: Retrieves the ACTIVE version for a given policy ID.</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
10	GetActiveVersionByPolicyCodeAsync	<p>Visibility: public</p> <p>Return: Task<PolicyVersion?></p> <p>Purpose: Retrieves the ACTIVE version for a given policy code (excluding deleted policies).</p> <p>Parameters: policyCode: string, ct: CancellationToken</p>
11	GetNextVersionNumberAsync	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Calculates the next version number for a policy based on the current maximum version.</p> <p>Parameters: policyId: int, ct: CancellationToken</p>
12	CreateVersionAsync	<p>Visibility: public</p> <p>Return: Task<PolicyVersion></p> <p>Purpose: Creates a new policy version and sets CreatedAt/UpdatedAt timestamps.</p> <p>Parameters: version: PolicyVersion, ct: CancellationToken</p>

13	UpdateVersionAsync	<p>Visibility: public</p> <p>Return: Task<PolicyVersion></p> <p>Purpose: Updates an existing policy version and sets UpdatedAt timestamp.</p> <p>Parameters: version: PolicyVersion, ct: CancellationToken</p>
14	DeleteVersionAsync	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Deletes a policy version only when status is DRAFT.</p> <p>Parameters: policyVersionId: int, ct: CancellationToken</p>
15	GetRequiredPoliciesAsync	<p>Visibility: public</p> <p>Return: Task<List<Policy>></p> <p>Purpose: Retrieves active policies that require user consent, including ACTIVE versions.</p> <p>Parameters: ct: CancellationToken</p>
16	GetActiveRequiredVersionsAsync	<p>Visibility: public</p> <p>Return: Task<List<PolicyVersion>></p> <p>Purpose: Retrieves ACTIVE policy versions that require consent and belong to active, non-deleted policies.</p> <p>Parameters: ct: CancellationToken</p>
17	HasUserAcceptedVersionAsync	<p>Visibility: public</p> <p>Return: Task<bool></p>

		<p>Purpose: Checks whether a user has a valid acceptance record for a specific policy version.</p> <p>Parameters: userId: int, policyVersionId: int, ct: CancellationToken</p>
18	GetUserValidAcceptsAsync	<p>Visibility: public</p> <p>Return: Task<List<UserPolicyAccept>></p> <p>Purpose: Retrieves valid acceptance records of a user (IsValid=true) with policy and version details.</p> <p>Parameters: userId: int, ct: CancellationToken</p>
19	GetUserAcceptHistoryAsync	<p>Visibility: public</p> <p>Return: Task<List<UserPolicyAccept>></p> <p>Purpose: Retrieves full acceptance history of a user ordered by accepted time descending.</p> <p>Parameters: userId: int, ct: CancellationToken</p>
20	CreateAcceptAsync	<p>Visibility: public</p> <p>Return: Task<UserPolicyAccept></p> <p>Purpose: Creates a new acceptance record and sets CreatedAt/AcceptedAt timestamps.</p> <p>Parameters: accept: UserPolicyAccept, ct: CancellationToken</p>
21	InvalidateOldAcceptsAsync	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Invalidates previous valid acceptance records of a user for a given policy when a new version is accepted.</p>

		Parameters: userId: int, policyId: int, ct: CancellationToken
22	InvalidateAllAcceptsForVersionAsync	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Invalidates all valid acceptance records for a specific policy version (e.g., when version is superseded).</p> <p>Parameters: policyVersionId: int, ct: CancellationToken</p>
23	CountAcceptedUsersAsync	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Counts distinct active users (RoleId=3) who have valid acceptance for a policy version.</p> <p>Parameters: policyVersionId: int, ct: CancellationToken</p>
24	CountActiveUsersAsync	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Counts active normal users (RoleId=3) excluding Admin/Expert, used for acceptance rate calculation.</p> <p>Parameters: ct: CancellationToken</p>

4.5.23 BadWordRepository

No	Name	Description
Attributes		
01	_context	<p>Visibility: protected</p> <p>Type: PawnderDatabaseContext</p> <p>Purpose: Provides access to the database context for managing prohibited word data used in content moderation.</p>

02	_dbSet	<p>Visibility: protected</p> <p>Type: DbSet<BadWord></p> <p>Purpose: Represents the BadWord entity set used for database operations.</p>
Methods/Operations		
01	GetActiveBadWordsAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<BadWord>></p> <p>Purpose: Retrieves all active prohibited words, ordered by severity level and category, for content filtering and moderation.</p> <p>Parameters: ct: CancellationToken</p>
02	GetBadWordsByLevelAsync	<p>Visibility: public</p> <p>Return: Task<IEnumerable<BadWord>></p> <p>Purpose: Retrieves active prohibited words filtered by a specific severity level.</p> <p>Parameters: level: int, ct: CancellationToken</p>

4.6 IRepository

4.6.1 IAddressRepository

No	Name	Description
Methods/Operations		

01	GetAddressByIdAs ync	<p>Return type: Task<Address?></p> <p>Purpose: Retrieves an address by its ID from the database; returns null if the address does not exist.</p> <p>Parameters: int addressId, CancellationToken ct = default.</p>
----	-------------------------	--

4.6.2 IAttributeOptionRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllOptionsAsyn c	<p>Return type: Task<IEnumerable<OptionResponse>></p> <p>Purpose: Retrieves all non-deleted attribute options from the database.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetOptionsByAttri butIdAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all non-deleted options associated with a specific attribute.</p> <p>Parameters: int attributeId, CancellationToken ct = default.</p>

4.6.3 IAttributeRepository

No	Name	Description
Attributes		
Methods/Operations		

01	GetAttributesAsync	<p>Return type: Task<PagedResult<AttributeResponse>></p> <p>Purpose: Retrieves attributes with pagination, optional search keyword, and ability to include or exclude soft-deleted records.</p> <p>Parameters: string? search, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.</p>
02	GetAttributeByIdAsync	<p>Return type: Task<AttributeEntity?></p> <p>Purpose: Retrieves a single attribute by its identifier from the database. Returns null if not found.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
03	NameExistsAsync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether an attribute name already exists in the database, with an optional exclusion of a specific attribute ID (used when updating).</p> <p>Parameters: string name, int? excludeId = null, CancellationToken ct = default.</p>
04	GetAttributesForFilterAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all active attributes along with their non-deleted options, formatted for use in filtering or matching features.</p> <p>Parameters: CancellationToken ct = default.</p>

4.6.4 IBaseRepository

No	Name	Description
Attributes		

Methods/Operations		
01	GetByIdAsync	<p>Return type: Task<T?></p> <p>Purpose: Retrieves a single entity by its identifier from the database. Returns null if the entity does not exist.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
02	GetAllAsync	<p>Return type: Task<IEnumerable<T>></p> <p>Purpose: Retrieves all entities of type T from the database.</p> <p>Parameters: CancellationToken ct = default.</p>
03	FindAsync	<p>Return type: Task<IEnumerable<T>></p> <p>Purpose: Retrieves entities that match the specified predicate condition.</p> <p>Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.</p>
04	FirstOrDefaultAsync	<p>Return type: Task<T?></p> <p>Purpose: Retrieves the first entity that matches the specified condition, or returns null if no match is found.</p> <p>Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.</p>
05	AddAsync	<p>Return type: Task<T></p> <p>Purpose: Adds a new entity to the database and persists the change.</p> <p>Parameters: T entity, CancellationToken ct = default.</p>
06	AddRangeAsync	Return type: Task

		<p>Purpose: Adds multiple entities to the database in a single operation.</p> <p>Parameters: <code>IEnumerable<T> entities, CancellationToken ct = default.</code></p>
07	UpdateAsync	<p>Return type: Task</p> <p>Purpose: Updates an existing entity in the database and persists the change.</p> <p>Parameters: <code>T entity, CancellationToken ct = default.</code></p>
08	UpdateRangeAsync	<p>Return type: Task</p> <p>Purpose: Updates multiple entities in a single operation.</p> <p>Parameters: <code>IEnumerable<T> entities, CancellationToken ct = default.</code></p>
09	DeleteAsync	<p>Return type: Task</p> <p>Purpose: Removes an entity from the database.</p> <p>Parameters: <code>T entity, CancellationToken ct = default.</code></p>
10	DeleteRangeAsync	<p>Return type: Task</p> <p>Purpose: Removes multiple entities from the database in a single operation.</p> <p>Parameters: <code>IEnumerable<T> entities, CancellationToken ct = default.</code></p>
11	ExistsAsync	<p>Return type: <code>Task<bool></code></p> <p>Purpose: Checks whether any entity exists that matches the specified predicate condition.</p>

		Parameters: Expression<Func<T, bool>> predicate, CancellationToken ct = default.
12	CountAsync	<p>Return type: Task<int></p> <p>Purpose: Counts the number of entities that match the specified condition. If no predicate is provided, counts all entities.</p> <p>Parameters: Expression<Func<T, bool>>? predicate = null, CancellationToken ct = default.</p>
13	SaveChangesAsync	<p>Return type: Task</p> <p>Purpose: Persists all pending changes to the database.</p> <p>Parameters: CancellationToken ct = default.</p>

4.6.5 IBlockRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetBlockedUsersA sync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves a list of users that have been blocked by a specific user. Includes basic information about the blocked users.</p> <p>Parameters: int fromUserId, CancellationToken ct = default.</p>
02	GetBlockAsync	<p>Return type: Task<Block?></p> <p>Purpose: Retrieves a specific block relationship between two users. Returns null if the block does not exist.</p>

		Parameters: int fromUserId, int toUserId, CancellationToken ct = default.
03	BlockExistsAsync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether a block relationship already exists between two users.</p> <p>Parameters: int fromUserId, int toUserId, CancellationToken ct = default.</p>

4.6.6 IChatExpertContentRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetChatMessages Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all messages associated with a specific expert chat session, ordered by creation time.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>
02	ChatExpertExistsA sync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether an expert chat session exists in the system.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>

4.6.7 IChatExpertRepository

No	Name	Description
Attributes		

Methods/Operations		
01	GetChatsByUserId Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions associated with a specific user. Includes summary information such as expert details and last message metadata.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsByExpertId dAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat sessions assigned to a specific expert, along with related user information.</p> <p>Parameters: int expertId, CancellationToken ct = default.</p>
03	GetChatExpertByExpertAndUserAsync	<p>Return type: Task<ChatExpert?></p> <p>Purpose: Retrieves a chat session between a specific expert and a specific user. Returns null if the chat does not exist.</p> <p>Parameters: int expertId, int userId, CancellationToken ct = default.</p>
04	ChatExpertExistsA sync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether a chat expert session exists based on its identifier.</p> <p>Parameters: int chatExpertId, CancellationToken ct = default.</p>

4.6.8 IChatUserContentRepository

No	Name	Description
Attributes		
Methods/Operations		

01	GetChatMessages Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat messages belonging to a specific match, ordered by creation time. Includes sender information and message content.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
02	ChatExistsAsync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether an active chat session exists for a given match identifier. Typically used to validate access before performing chat operations.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>

4.6.9 IChatUserRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetInvitesAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all pending chat invitations for a user based on the user's pets. These invitations represent incoming match requests waiting for acceptance.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all active chat sessions for a user. Optionally filters chats by a specific pet owned by the user.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default.</p>

03	GetChatUserByPet sAsync	<p>Return type: Task<ChatUser?></p> <p>Purpose: Retrieves a chat session between two specific pets. Returns null if no active chat exists.</p> <p>Parameters: int fromPetId, int toPetId, CancellationToken ct = default.</p>
04	GetChatUserByMa tchIdAsync	<p>Return type: Task<ChatUser?></p> <p>Purpose: Retrieves a chat session using its match identifier. Commonly used to validate or process pending match requests.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>

4.6.10 IExpertConfirmationRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllExpertConfir mationsAsync	<p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation records, including related user, expert, and chat information.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetExpertConfirm ationAsync	<p>Return type: Task<ExpertConfirmation?></p> <p>Purpose: Retrieves a specific expert confirmation based on expert, user, and chat identifiers. Returns null if no record is found.</p> <p>Parameters: int expertId, int userId, int chatId, CancellationToken ct = default.</p>

03	GetUserExpertConfirmationsAsync	<p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation records associated with a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
04	GetExpertConfirmationByUserAndChatAsync	<p>Return type: Task<ExpertConfirmation?></p> <p>Purpose: Retrieves an expert confirmation record based on user and chat identifiers. Returns null if not found.</p> <p>Parameters: int userId, int chatId, CancellationToken ct = default</p>

4.6.11 INotificationRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllNotificationsAsync	<p>Return type: Task<IEnumerable<NotificationDto>></p> <p>Purpose: Retrieves all notifications in the system, including related user information.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetNotificationByIdAsync	<p>Return type: Task<NotificationDto?></p> <p>Purpose: Retrieves a specific notification by its identifier. Returns null if the notification does not exist.</p> <p>Parameters: int notificationId, CancellationToken ct = default.</p>

03	GetNotificationsByUserIdAsync	<p>Return type: Task<IEnumerable<Notification>></p> <p>Purpose: Retrieves all notifications belonging to a specific user, ordered by creation time.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
04	MarkAllAsReadAsync	<p>Return type: Task<int></p> <p>Purpose: Marks all unread notifications of a specific user as read and returns the number of affected notifications.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
05	GetUnreadCountAsync	<p>Return type: Task<int></p> <p>Purpose: Returns the number of unread notifications for a specific user, typically used for badge or alert display.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

4.6.12 IPaymentHistoryRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetPaymentHistoriesByUserIdAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all payment history records associated with a specific user, ordered by creation date.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetAllPaymentHistoriesAsync	Return type: Task<IEnumerable<object>>

		<p>Purpose: Retrieves all payment history records across all users, including basic user information. Useful for administrative reporting.</p> <p>Parameters: CancellationToken ct = default.</p>
03	GetVipStatusAsync	<p>Return type: Task<object?></p> <p>Purpose: Retrieves the current active VIP subscription status for a user, if any. Returns null if the user does not have an active VIP service.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

4.6.13 IPetCharacteristicRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetPetCharacteristicsAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all characteristics associated with a specific pet, including attribute name, type, unit, and value or option.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	GetPetCharacteristicAsync	<p>Return type: Task<PetCharacteristic?></p> <p>Purpose: Retrieves a specific characteristic for a pet based on the attribute identifier. Returns null if the characteristic does not exist.</p> <p>Parameters: int petId, int attributeId, CancellationToken ct = default.</p>

03	ExistsAsync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether a pet characteristic already exists for a given pet and attribute combination.</p> <p>Parameters: int petId, int attributeId, CancellationToken ct = default.</p>
----	-------------	---

4.6.14 IReportRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllReportsAsync	<p>Return type: Task<IEnumerable<ReportDto>></p> <p>Purpose: Retrieves all reports in the system, including reporter and reported user information for administrative review.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetReportByIdAsync	<p>Return type: Task<ReportDto?></p> <p>Purpose: Retrieves a single report by its identifier, including detailed context about the report and involved users. Returns null if not found.</p> <p>Parameters: int reportId, CancellationToken ct = default.</p>

4.6.15 IPetPhotoRepository

No	Name	Description
Attributes		
Methods/Operations		

01	GetPhotosByPetId Async	<p>Return type: Task<IEnumerable<PetPhotoResponse>></p> <p>Purpose: Retrieves all non-deleted photos associated with a specific pet, ordered by sort order and creation sequence.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	GetPhotoCountBy PetIdAsync	<p>Return type: Task<int></p> <p>Purpose: Counts the number of active (non-deleted) photos associated with a specific pet.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
03	GetMaxSortOrder Async	<p>Return type: Task<int?></p> <p>Purpose: Retrieves the highest sort order value among existing pet photos. Returns null if no photos exist. Used when assigning a new photo order.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>

4.6.16 IUserRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetUsersAsync	<p>Return type: Task<PagedResult<UserResponse>></p> <p>Purpose: Retrieves users with pagination support, optional search keyword, role filtering, status filtering, and the ability to include or exclude soft-deleted records.</p>

		Parameters: string? search, int? roleId, int? statusId, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.
02	GetUserByIdAsync	<p>Return type: Task<User?></p> <p>Purpose: Retrieves a user by its identifier, with optional inclusion of soft-deleted users. Returns null if the user does not exist.</p> <p>Parameters: int userId, bool includeDeleted = false, CancellationToken ct = default.</p>
03	EmailExistsAsync	<p>Return type: Task<bool></p> <p>Purpose: Checks whether an email address already exists in the system for a non-deleted user. Used during registration and validation workflows.</p> <p>Parameters: string email, CancellationToken ct = default.</p>
04	GetUserByEmailAs ync	<p>Return type: Task<User?></p> <p>Purpose: Retrieves a user by email address, including role information. Returns null if no matching user is found.</p> <p>Parameters: string email, CancellationToken ct = default.</p>

4.6.17 IAppointmentRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetByIdAsy nc()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p>

		<p>Purpose: Retrieves all appointments associated with a specific match.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • matchId: int – Identifier of the match • ct: CancellationToken (optional)
02	GetByUserIdAsync() ()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves all appointments related to a user, including both inviter and invitee roles.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • userId: int – Identifier of the user • ct: CancellationToken (optional)
03	GetByStatusAsync() ()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves appointments filtered by their current status.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • status: string – Appointment status • ct: CancellationToken (optional)
04	GetByIdWithDetailsAsync() ()	<p>Visibility: public</p> <p>Return: Task<PetAppointment?></p> <p>Purpose: Retrieves an appointment by ID including all related navigation properties.</p>

		<p>Parameters:</p> <ul style="list-style-type: none"> • appointmentId: int – Identifier of the appointment • ct: CancellationToken (optional)
05	GetUpcomingAppointmentsForReminderAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointment>></p> <p>Purpose: Retrieves upcoming appointments that require reminder notifications.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • reminderTime: DateTime – Reminder trigger time • ct: CancellationToken (optional)
06	CountMessagesBetweenUsersAsync()	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Counts the number of messages exchanged between two users within a specific match.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • matchId: int – Identifier of the match • ct: CancellationToken (optional)
07	IsPetProfileCompleteAsync()	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a pet profile contains all required information before allowing appointment-related actions.</p> <p>Parameters:</p>

		<ul style="list-style-type: none"> • petId: int – Identifier of the pet • ct: CancellationToken (optional)
--	--	--

4.6.18 IAppointmentLocationRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetByCityAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointmentLocation>></p> <p>Purpose: Retrieves a list of pet-friendly appointment locations filtered by city.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • city: string – Name of the city • ct: CancellationToken (optional)
02	GetNearbyLocationsAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetAppointmentLocation>></p> <p>Purpose: Retrieves nearby pet-friendly locations based on the user's current geographic coordinates within a specified radius.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • latitude: decimal – Current latitude • longitude: decimal – Current longitude • radiusKm: decimal – Search radius in kilometers (default: 10 km)

		<ul style="list-style-type: none"> • ct: CancellationToken (optional)
03	GetByGooglePlaceIdAsync()	<p>Visibility: public</p> <p>Return: Task<PetAppointmentLocation?></p> <p>Purpose: Retrieves a pet-friendly location using its Google Place ID for external location data synchronization.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • googlePlaceId: string – Google Place identifier • ct: CancellationToken (optional)

4.6.19 IEventRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllEventsAsync() ()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves all events regardless of status for administrative management purposes.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ct: CancellationToken (optional)
02	GetActiveEventsAs ync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events that are currently active or scheduled to occur in the near future.</p>

		<p>Parameters:</p> <ul style="list-style-type: none"> • ct: CancellationToken (optional)
03	GetEventWithSubmissionsAsync()	<p>Visibility: public</p> <p>Return: Task<PetEvent?></p> <p>Purpose: Retrieves a specific event along with its associated submission records.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • eventId: int – Identifier of the event • ct: CancellationToken (optional)
04	GetEventsToTransitionAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events that require automatic status transitions, typically processed by background jobs.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ct: CancellationToken (optional)
05	GetEventsByStatusAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<PetEvent>></p> <p>Purpose: Retrieves events filtered by their current status.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • status: string – Event status • ct: CancellationToken (optional)

4.6.20 ISubmissionRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetByIdAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<EventSubmission>></p> <p>Purpose: Retrieves all submissions associated with a specific event.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • eventId: int – Identifier of the event • ct: CancellationToken (optional)
02	GetLeaderboardSync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<EventSubmission>></p> <p>Purpose: Retrieves the top submissions of an event, sorted by vote count.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • eventId: int – Identifier of the event • top: int – Number of top submissions to retrieve (default: 10) • ct: CancellationToken (optional)
03	HasUserSubmittedAsync()	<p>Visibility: public</p> <p>Return: Task<bool></p>

		<p>Purpose: Checks whether a user has already submitted an entry for a specific event.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • eventId: int – Identifier of the event • userId: int – Identifier of the user • ct: CancellationToken (optional)
04	GetByIdWithDetail sAsync()	<p>Visibility: public</p> <p>Return: Task<EventSubmission?></p> <p>Purpose: Retrieves a submission by ID including related user, pet, and vote information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • submissionId: int – Identifier of the submission • ct: CancellationToken (optional)
05	HasUserVotedAsyn c()	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a user has already voted for a specific submission.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • submissionId: int – Identifier of the submission • userId: int – Identifier of the user • ct: CancellationToken (optional)
06	AddVoteAsync()	Visibility: public

		<p>Return: Task</p> <p>Purpose: Adds a vote from a user to a specific submission.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • submissionId: int – Identifier of the submission • userId: int – Identifier of the user • ct: CancellationToken (optional)
07	RemoveVoteAsync() ()	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Removes an existing vote from a user for a specific submission.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • submissionId: int – Identifier of the submission • userId: int – Identifier of the user • ct: CancellationToken (optional)
08	UpdateVoteCount Async()	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Updates the denormalized vote count of a submission to ensure data consistency and performance.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • submissionId: int – Identifier of the submission • ct: CancellationToken (optional)

4.6.21 IPolicyRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllPoliciesAsyn c()	<p>Visibility: public</p> <p>Return: Task<List<Policy>></p> <p>Purpose: Retrieves all policies excluding soft-deleted records.</p> <p>Parameters: ct (optional)</p>
02	GetPolicyByIdAsyn c()	<p>Visibility: public</p> <p>Return: Task<Policy?></p> <p>Purpose: Retrieves a policy by its unique identifier.</p> <p>Parameters: policyId: int, ct (optional)</p>
03	GetPolicyByCodeA sync()	<p>Visibility: public</p> <p>Return: Task<Policy?></p> <p>Purpose: Retrieves a policy by its unique code.</p> <p>Parameters: policyCode: string, ct (optional)</p>
04	CreatePolicyAsync()	<p>Visibility: public</p> <p>Return: Task<Policy></p> <p>Purpose: Creates a new policy record.</p> <p>Parameters: policy: Policy, ct (optional)</p>

05	UpdatePolicyAsync() ()	Visibility: public Return: Task<Policy> Purpose: Updates an existing policy record. Parameters: policy: Policy, ct (optional)
06	DeletePolicyAsync() ()	Visibility: public Return: Task<bool> Purpose: Performs a soft delete on a policy. Parameters: policyId: int, ct (optional)
07	GetVersionsByPolicyIdAsync() ()	Visibility: public Return: Task<List<PolicyVersion>> Purpose: Retrieves all versions belonging to a specific policy. Parameters: policyId: int, ct (optional)
08	GetVersionByIdAsync() ()	Visibility: public Return: Task<PolicyVersion?> Purpose: Retrieves a specific policy version by ID. Parameters: policyVersionId: int, ct (optional)
09	GetActiveVersionByPolicyIdAsync() ()	Visibility: public Return: Task<PolicyVersion?> Purpose: Retrieves the currently active version of a policy using policy ID. Parameters: policyId: int, ct (optional)

10	GetActiveVersionByPolicyCodeAsync()	<p>Visibility: public Return: Task<PolicyVersion?></p> <p>Purpose: Retrieves the currently active version of a policy using policy code.</p> <p>Parameters: policyCode: string, ct (optional)</p>
11	GetNextVersionNumberAsync()	<p>Visibility: public Return: Task<int></p> <p>Purpose: Generates the next sequential version number for a policy.</p> <p>Parameters: policyId: int, ct (optional)</p>
12	CreateVersionAsync()	<p>Visibility: public Return: Task<PolicyVersion></p> <p>Purpose: Creates a new policy version (typically in DRAFT state).</p> <p>Parameters: version: PolicyVersion, ct (optional)</p>
13	UpdateVersionAsync()	<p>Visibility: public Return: Task<PolicyVersion></p> <p>Purpose: Updates an existing policy version.</p> <p>Parameters: version: PolicyVersion, ct (optional)</p>
14	DeleteVersionAsync()	<p>Visibility: public Return: Task<bool></p>

		<p>Purpose: Deletes a policy version; only allowed if the version is in DRAFT state.</p> <p>Parameters: policyVersionId: int, ct (optional)</p>
15	GetRequiredPoliciesAsync()	<p>Visibility: public</p> <p>Return: Task<List<Policy>></p> <p>Purpose: Retrieves all policies that are active and require user consent.</p> <p>Parameters: ct (optional)</p>
16	GetActiveRequiredVersionsAsync()	<p>Visibility: public</p> <p>Return: Task<List<PolicyVersion>></p> <p>Purpose: Retrieves all active versions of policies that require user consent.</p> <p>Parameters: ct (optional)</p>
17	HasUserAcceptedVersionAsync()	<p>Visibility: public</p> <p>Return: Task<bool></p> <p>Purpose: Checks whether a user has accepted a specific policy version.</p> <p>Parameters: userId: int, policyVersionId: int, ct (optional)</p>
18	GetUserValidAcceptsAsync()	<p>Visibility: public</p> <p>Return: Task<List<UserPolicyAccept>></p> <p>Purpose: Retrieves the user's valid (currently effective) acceptances.</p> <p>Parameters: userId: int, ct (optional)</p>

19	GetUserAcceptHistoryAsync()	<p>Visibility: public</p> <p>Return: Task<List<UserPolicyAccept>></p> <p>Purpose: Retrieves the full acceptance history of a user for audit purposes.</p> <p>Parameters: userId: int, ct (optional)</p>
20	CreateAcceptAsync()	<p>Visibility: public</p> <p>Return: Task<UserPolicyAccept></p> <p>Purpose: Creates a new acceptance record when a user consents to a policy version.</p> <p>Parameters: accept: UserPolicyAccept, ct (optional)</p>
21	InvalidateOldAcceptsAsync()	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Invalidates previous acceptances of the same policy for a user when a new version becomes effective.</p> <p>Parameters: userId: int, policyId: int, ct (optional)</p>
22	InvalidateAllAcceptsForVersionAsync()	<p>Visibility: public</p> <p>Return: Task</p> <p>Purpose: Invalidates all acceptance records linked to a specific policy version (commonly when publishing a newer version).</p> <p>Parameters: policyVersionId: int, ct (optional)</p>
23	CountAcceptedUsersAsync()	<p>Visibility: public</p> <p>Return: Task<int></p>

		<p>Purpose: Counts how many users have accepted a specific policy version.</p> <p>Parameters: policyVersionId: int, ct (optional)</p>
24	CountActiveUsers Async()	<p>Visibility: public</p> <p>Return: Task<int></p> <p>Purpose: Counts the total number of active users in the system for compliance metrics.</p> <p>Parameters: ct (optional)</p>

4.6.22 IBadWordRepository

No	Name	Description
Attributes		
Methods/Operations		
01	GetActiveBadWordsAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<BadWord>></p> <p>Purpose: Retrieves all active bad words currently applied in the content moderation system.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ct: CancellationToken (optional)
02	GetBadWordsByLevelAsync()	<p>Visibility: public</p> <p>Return: Task<IEnumerable<BadWord>></p> <p>Purpose: Retrieves bad words filtered by severity level to support tiered moderation logic.</p> <p>Parameters:</p>

		<ul style="list-style-type: none"> • level: int – Severity level of the bad words • ct: CancellationToken (optional)
--	--	--

4.7 IService

4.7.1 IAddressService

No	Name	Description
Attributes		
Methods/Operations		
01	CreateAddressForUserAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new address for a user based on location information.</p> <p>Parameters: int userId, LocationDto locationDto, CancellationToken ct = default.</p>
02	UpdateAddressAsync	<p>Return type: Task<object></p> <p>Purpose: Updates address information by ID based on new location data.</p> <p>Parameters: int addressId, LocationDto locationDto, CancellationToken ct = default.</p>
03	UpdateAddressManualAsync	<p>Return type: Task<object></p> <p>Purpose: Manually updates address by ID with manually entered address information.</p>

		Parameters: int addressId, ManualAddressDto dto, CancellationToken ct = default.
04	GetAddressByIdAs ync	Return type: Task<object> Purpose: Retrieves address information by ID. Parameters: int addressId, CancellationToken ct = default.

4.7.2 IAdminService

No	Name	Description
Attributes		
Methods/Operations		
01	ReassignExpertCo nfirmationAsync	Return type: Task<object> Purpose: Reassigns expert confirmation request to another expert. Parameters: ReassignExpertConfirmationRequest req, CancellationToken ct = default.
02	BanUserAsync	Return type: Task<object> Purpose: Bans a user from the platform based on the provided ban request. Parameters: int userId, BanUserRequest req, CancellationToken ct = default.
03	UnbanUserAsync	Return type: Task<object> Purpose: Unbans a previously banned user from the platform.

		Parameters: int userId, UnbanUserRequest? req, CancellationToken ct = default.
04	GetUserBansAsync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all ban records for a specific user. Parameters: int userId, CancellationToken ct = default.
05	UpdateUserByAd minAsync	Return type: Task<bool> Purpose: Updates user information by administrator. Parameters: int userId, AdUserUpdateRequest request, CancellationToken ct = default.
06	RegisterUserByAd minAsync	Return type: Task<UserResponse> Purpose: Registers a new user account by administrator. Parameters: AdUserCreateRequest req, CancellationToken ct = default.

No	Name	Description
Attributes		
Methods/Operations		
01	ReassignExpertCo nfirmationAsync	Return type: Task<object> Purpose: Reassigns expert confirmation request to another expert.

		Parameters: ReassignExpertConfirmationRequest req, CancellationToken ct = default.
02	BanUserAsync	<p>Return type: Task<object></p> <p>Purpose: Bans a user from the platform based on the provided ban request.</p> <p>Parameters: int userId, BanUserRequest req, CancellationToken ct = default.</p>
03	UnbanUserAsync	<p>Return type: Task<object></p> <p>Purpose: Unbans a previously banned user from the platform.</p> <p>Parameters: int userId, UnbanUserRequest? req, CancellationToken ct = default.</p>
04	GetUserBansAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all ban records for a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
05	UpdateUserByAdminAsync	<p>Return type: Task<bool></p> <p>Purpose: Updates user information by administrator.</p> <p>Parameters: int userId, AdUserUpdateRequest request, CancellationToken ct = default.</p>

4.7.3 IAttributeOptionService

No	Name	Description
----	------	-------------

Attributes		
Methods/Operations		
01	GetAllOptionsAsync	<p>Return type: Task<IEnumerable<OptionResponse>></p> <p>Purpose: Retrieves all non-deleted attribute options from the database.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetOptionsByAttributeIdAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all non-deleted options associated with a specific attribute.</p> <p>Parameters: int attributeId, CancellationToken ct = default.</p>
03	CreateOptionAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new option for a specific attribute.</p> <p>Parameters: int attributeId, string optionName, CancellationToken ct = default.</p>
04	UpdateOptionAsync	<p>Return type: Task<bool></p> <p>Purpose: Updates the name of an existing option.</p> <p>Parameters: int optionId, string optionName, CancellationToken ct = default.</p>
05	DeleteOptionAsync	<p>Return type: Task<bool></p> <p>Purpose: Soft deletes an option by its ID.</p>

		Parameters: int optionId, CancellationToken ct = default.
--	--	---

4.7.4 IAttributeService

No	Name	Description
Attributes		
Methods/Operations		
01	GetAttributesAsyn c	<p>Return type: Task<PagedResult<AttributeResponse>></p> <p>Purpose: Retrieves a paginated list of attributes with optional search and filtering.</p> <p>Parameters: string? search, int page, int pageSize, bool includeDeleted, CancellationToken ct = default.</p>
02	GetAttributeByIdA sync	<p>Return type: Task<AttributeResponse?></p> <p>Purpose: Retrieves a specific attribute by its ID.</p> <p>Parameters: int id, CancellationToken ct = default.</p>
03	CreateAttributeAs ync	<p>Return type: Task<AttributeResponse></p> <p>Purpose: Creates a new attribute in the system.</p> <p>Parameters: AttributeCreateRequest request, CancellationToken ct = default.</p>
04	UpdateAttributeAs ync	<p>Return type: Task<bool></p> <p>Purpose: Updates an existing attribute by its ID.</p>

		Parameters: int id, AttributeUpdateRequest request, CancellationToken ct = default.
05	DeleteAttributeAs ync	<p>Return type: Task<bool></p> <p>Purpose: Deletes an attribute by its ID, either soft or hard delete.</p> <p>Parameters: int id, bool hard = false, CancellationToken ct = default.</p>
06	GetAttributesForFi lterAsync	<p>Return type: Task<object></p> <p>Purpose: Retrieves attributes formatted for filtering purposes.</p> <p>Parameters: CancellationToken ct = default.</p>

4.7.5 IAuthService

No	Name	Description
Attributes		
Methods/Operations		
01	LoginAsync	<p>Return type: Task<object></p> <p>Purpose: Authenticates a user and returns login credentials.</p> <p>Parameters: LoginRequest request, CancellationToken ct = default.</p>
02	RefreshTokenAsyn c	Return type: Task<object>

		Purpose: Refreshes the authentication token using a refresh token. Parameters: RefreshTokenRequest request, CancellationToken ct = default.
03	LogoutAsync	Return type: Task<bool>Purpose: Logs out a user and invalidates their session.Parameters: int userId, CancellationToken ct = default.
04	ChangePasswordA sync	Return type: Task<object>Purpose: Changes the password for a user account.Parameters: int userId, ChangePasswordRequest request, CancellationToken ct = default.

4.7.6 IBlockService

No	Name	Description
Attributes		
Methods/Operations		
01	GetBlockedUsersA sync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all users blocked by a specific user. Parameters: int fromUserId, CancellationToken ct = default.
02	CreateBlockAsync	Return type: Task<object> Purpose: Creates a new block relationship between two users.

		Parameters: int fromUserId, int toUserId, CancellationToken ct = default.
03	DeleteBlockAsync	Return type: Task<bool> Purpose: Removes a block relationship between two users. Parameters: int fromUserId, int toUserId, CancellationToken ct = default.

4.7.7 IChatAIService

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllChatsAsync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all AI chat sessions for a specific user. Parameters: int userId, CancellationToken ct = default.
02	CreateChatAsync	Return type: Task<object> Purpose: Creates a new AI chat session for a user. Parameters: int userId, string? title, CancellationToken ct = default.
03	UpdateChatTitleAs ync	Return type: Task<bool> Purpose: Updates the title of an existing AI chat session. Parameters: int chatAId, int userId, string title, CancellationToken ct = default.

04	DeleteChatAsync	Return type: Task<bool> Purpose: Deletes an AI chat session for a user. Parameters: int chatAild, int userId, CancellationToken ct = default.
05	GetChatHistoryAsync	Return type: Task<object> Purpose: Retrieves the message history of an AI chat session. Parameters: int chatAild, int userId, CancellationToken ct = default.
06	SendMessageAsync	Return type: Task<object> Purpose: Sends a message to the AI and receives a response. Parameters: int chatAild, int userId, string question, CancellationToken ct = default.
07	GetTokenUsageAsync	Return type: Task<object> Purpose: Retrieves the AI token usage statistics for a user. Parameters: int userId, CancellationToken ct = default.

4.7.8 IChatExpertContentService

No	Name	Description
Attributes		
Methods/Operations		
01	GetChatMessagesAsync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all messages from an expert chat session. Parameters: int chatExpertId, CancellationToken ct = default.
02	SendMessageAsync	Return type: Task<object> Purpose: Sends a message in an expert chat session.

		Parameters: int chatExpertId, int fromId, string message, int? expertId, int? userId, int? chatAiid, CancellationToken ct = default.
--	--	--

4.7.9 IChatExpertService

No	Name	Description
Attributes		
Methods/Operations		
01	GetChatsByUserId Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions for a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsByExpertId Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat sessions for a specific expert.</p> <p>Parameters: int expertId, CancellationToken ct = default.</p>
03	CreateChatAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new chat session between an expert and a user.</p> <p>Parameters: int expertId, int userId, CancellationToken ct = default.</p>

4.7.10 IChatUserContentService

No	Name	Description
Attributes		

Methods/Operations		
01	GetChatMessages Async	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all messages from a matched user chat session.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
02	SendMessageAsyn c	<p>Return type: Task<object></p> <p>Purpose: Sends a message in a matched user chat session.</p> <p>Parameters: int matchId, int fromUserId, string message, CancellationToken ct = default.</p>

4.7.11 IChatUserService

No	Name	Description
Attributes		
Methods/Operations		
01	GetInvitesAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all pending friend request invites for a user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
02	GetChatsAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all chat sessions for a user, optionally filtered by pet.</p>

		Parameters: int userId, int? petId, CancellationToken ct = default.
03	CreateFriendRequestAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new friend request between two pets.</p> <p>Parameters: int fromPetId, int toPetId, CancellationToken ct = default.</p>
04	UpdateFriendRequestAsync	<p>Return type: Task<object></p> <p>Purpose: Accepts or updates a friend request status.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
05	DeleteFriendRequestAsync	<p>Return type: Task<bool></p> <p>Purpose: Deletes or rejects a friend request.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>
06	DeleteChatAsync	<p>Return type: Task<bool></p> <p>Purpose: Deletes a chat session between matched users.</p> <p>Parameters: int matchId, CancellationToken ct = default.</p>

4.7.12 IDailyLimitService

No	Name	Description
Attributes		
Methods/Operations		

01	CanPerformAction	<p>Return type: Task<bool></p> <p>Purpose: Checks if a user can perform a specific action based on daily limits.</p> <p>Parameters: int userId, string actionType.</p>
02	RecordAction	<p>Return type: Task<bool></p> <p>Purpose: Records an action performed by a user for daily limit tracking.</p> <p>Parameters: int userId, string actionType.</p>
03	GetActionCountToday	<p>Return type: Task<int></p> <p>Purpose: Gets the number of times a user has performed an action today.</p> <p>Parameters: int userId, string actionType.</p>
04	GetRemainingCount	<p>Return type: Task<int></p> <p>Purpose: Gets the remaining number of allowed actions for a user today.</p> <p>Parameters: int userId, string actionType.</p>
05	GetRemainingCountAsync	<p>Return type: Task<int></p> <p>Purpose: Gets the remaining number of allowed actions for a user today with cancellation support.</p>

		Parameters: int userId, string actionType, CancellationToken ct = default.
06	GetFreeQuotaForAction	<p>Return type: Task<int></p> <p>Purpose: Gets the free quota limit for a specific action type.</p> <p>Parameters: string actionType.</p>
07	GetFreeTokensUsedToday	<p>Return type: Task<int></p> <p>Purpose: Gets the number of free tokens a user has used today.</p> <p>Parameters: int userId.</p>
08	RecordTokenUsage	<p>Return type: Task<bool></p> <p>Purpose: Records the token usage for a user.</p> <p>Parameters: int userId, int tokensUsed.</p>

4.7.13 IExpertConfirmationService

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllExpertConfirmationsAsync	<p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmation requests from the system.</p> <p>Parameters: CancellationToken ct = default.</p>

02	GetExpertConfirmationAsync	<p>Return type: Task<ExpertConfirmationDTO?></p> <p>Purpose: Retrieves a specific expert confirmation by expert, user, and chat IDs.</p> <p>Parameters: int expertId, int userId, int chatId, CancellationToken ct = default.</p>
03	GetUserExpertConfirmationsAsync	<p>Return type: Task<IEnumerable<ExpertConfirmationDTO>></p> <p>Purpose: Retrieves all expert confirmations for a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
04	CreateExpertConfirmationAsync	<p>Return type: Task<ExpertConfirmationResponseDTO></p> <p>Purpose: Creates a new expert confirmation request.</p> <p>Parameters: int userId, int chatId, ExpertConfirmationCreateDTO dto, CancellationToken ct = default.</p>
05	UpdateExpertConfirmationAsync	<p>Return type: Task<ExpertConfirmationResponseDTO></p> <p>Purpose: Updates an existing expert confirmation request.</p> <p>Parameters: int expertId, int userId, int chatId, ExpertConfirmationUpdateDto dto, CancellationToken ct = default.</p>
06	GetUserExpertChatsAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all expert chat sessions for a specific user.</p>

		Parameters: int userId, CancellationToken ct = default.
--	--	---

4.7.14 IMatchService

No	Name	Description
Attributes		
Methods/Operations		
01	GetLikesReceivedA sync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all likes received by a user, optionally filtered by pet.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default.</p>
02	GetStatsAsync	<p>Return type: Task<object></p> <p>Purpose: Retrieves matching statistics for a user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
03	SendLikeAsync	<p>Return type: Task<object></p> <p>Purpose: Sends a like from one pet to another pet.</p> <p>Parameters: LikeRequest request, CancellationToken ct = default.</p>
04	RespondToLikeAsy nc	<p>Return type: Task<object></p> <p>Purpose: Responds to a received like with accept or reject action.</p>

		Parameters: RespondRequest request, CancellationToken ct = default.
05	GetBadgeCountsA sync	<p>Return type: Task<object></p> <p>Purpose: Retrieves badge notification counts for a user.</p> <p>Parameters: int userId, int? petId, CancellationToken ct = default.</p>

4.7.15 INotificationService

No	Name	Description
Attributes		
Methods/Operations		
01	GetAllNotifications Async	<p>Return type: Task<IEnumerable<NotificationDto>></p> <p>Purpose: Retrieves all notifications from the system.</p> <p>Parameters: CancellationToken ct = default.</p>
02	GetNotificationByldAsync	<p>Return type: Task<NotificationDto?></p> <p>Purpose: Retrieves a specific notification by its ID.</p> <p>Parameters: int notificationId, CancellationToken ct = default.</p>
03	GetNotificationsByUserIdAsync	<p>Return type: Task<IEnumerable<Notification>></p> <p>Purpose: Retrieves all notifications for a specific user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>

04	CreateNotification Async	<p>Return type: Task<Notification></p> <p>Purpose: Creates a new notification for a user.</p> <p>Parameters: NotificationDto_1 notificationDto, CancellationToken ct = default.</p>
05	MarkAsReadAsync	<p>Return type: Task<bool></p> <p>Purpose: Marks a specific notification as read.</p> <p>Parameters: int notificationId, CancellationToken ct = default.</p>
06	MarkAllAsReadAsy nc	<p>Return type: Task<int></p> <p>Purpose: Marks all notifications for a user as read and returns the count.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
07	GetUnreadCountA sync	<p>Return type: Task<int></p> <p>Purpose: Gets the count of unread notifications for a user.</p> <p>Parameters: int userId, CancellationToken ct = default.</p>
08	DeleteNotification Async	<p>Return type: Task<bool></p> <p>Purpose: Deletes a specific notification by its ID.</p> <p>Parameters: int notificationId, CancellationToken ct = default.</p>

4.7.16 IOtpService

No	Name	Description
----	------	-------------

Attributes		
Methods/Operations		
01	SendOtpAsync	<p>Return type: Task<object></p> <p>Purpose: Sends an OTP code to the specified email address.</p> <p>Parameters: string email, string purpose = "register", CancellationToken ct = default.</p>
02	CheckOtpAsync	<p>Return type: Task<bool></p> <p>Purpose: Validates the OTP code entered by the user.</p> <p>Parameters: string email, string otp, CancellationToken ct = default.</p>

4.7.17 IPaymentHistoryService

No	Name	Description
Attributes		
Methods/Operations		
01	GenerateQrAsync	<p>Return type: Task<byte[]></p> <p>Purpose: Generates a QR code for payment with specified amount and info.</p> <p>Parameters: decimal amount, string addInfo, CancellationToken ct = default.</p>
02	CreatePaymentHistoryAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new payment history record.</p>

		Parameters: CreatePaymentHistoryRequest request, CancellationToken ct = default.
03	GetPaymentHistoriesByIdUserAsync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all payment histories for a specific user. Parameters: int userId, CancellationToken ct = default.
04	GetAllPaymentHistoriesAsync	Return type: Task<IEnumerable<object>> Purpose: Retrieves all payment histories in the system. Parameters: CancellationToken ct = default.
05	GetVipStatusAsync	Return type: Task<object> Purpose: Gets the VIP subscription status for a user. Parameters: int userId, CancellationToken ct = default.
06	ProcessPaymentCallbackAsync	Return type: Task<object> Purpose: Processes payment callback notification from payment gateway. Parameters: JsonElement notification, int userIdFromToken, CancellationToken ct = default.
07	CheckPaymentInLastHourAsync	Return type: Task<object> Purpose: Checks for payment transactions in the last hour.

		Parameters: int userId, decimal transferAmount, string content, CancellationToken ct = default.
08	ValidateWebhook Async	<p>Return type: Task<bool></p> <p>Purpose: Validates the webhook authentication header.</p> <p>Parameters: string? authHeader, CancellationToken ct = default.</p>

4.7.18 IPetCharacteristicService

No	Name	Description
Attributes		
Methods/Operations		
01	GetPetCharacteristicsAsync	<p>Return type: Task<IEnumerable<object>></p> <p>Purpose: Retrieves all characteristics for a specific pet.</p> <p>Parameters: int petId, CancellationToken ct = default.</p>
02	CreatePetCharacteristicAsync	<p>Return type: Task<object></p> <p>Purpose: Creates a new characteristic for a pet with a specific attribute.</p> <p>Parameters: int petId, int attributeId, PetCharacteristicDTO dto, CancellationToken ct = default.</p>
03	UpdatePetCharacteristicAsync	<p>Return type: Task<object></p> <p>Purpose: Updates an existing characteristic for a pet.</p>

		Parameters: int petId, int attributeId, PetCharacteristicDTO dto, CancellationToken ct = default.
--	--	--

4.7.19 IPetPhotoService

No	Name	Description
Attributes		
Methods/Operations		
01	GetPhotosByPetId Async	Return type: Task<IEnumerable<PetPhotoResponse>> Purpose: Retrieves all photos for a specific pet. Parameters: int petId, CancellationToken ct = default.
02	UploadPhotosAsyn c	Return type: Task<IEnumerable<PetPhotoResponse>> Purpose: Uploads multiple photos for a pet. Parameters: int petId, List<IFormFile> files, CancellationToken ct = default.
03	ReorderPhotosAsy nc	Return type: Task<bool> Purpose: Reorders the display order of pet photos. Parameters: List<ReorderPhotoRequest> items, CancellationToken ct = default.
04	DeletePhotoAsync	Return type: Task<bool> Purpose: Deletes a pet photo, either soft or hard delete.

		Parameters: int photoid, bool hard = false, CancellationToken ct = default.
--	--	--