Project : Pet Diaries
Part II Group Section

Team:
Ankita Manjrekar
Anne Maria
Gayatri Mestry

Project summary:

Pet Diaries aims to create a collaborative web based platform for Owners to meet prospective Caretakers who are willing to take care of the Owner's pet while the owner is away. Owners and Caretakers are two of the three User types provided by the system, and Admin is the third User type. Owner's own pets and have default caretaking responsibilities, and Caretaker's volunteer to take care of pets but don't own pets of their own. Owners can inquire for caretakers by raising requests on Pet Diaries. These requests will be notified to prospective Caretakers who get notifications based on degree of match between the Owner's and Caretaker's location. Similarly, prospective Caretaker responses will be notified to the Owner, who can then choose to communicate with a Caretaker using a communication interface provided by Pet Diaries. If Owner and Caretaker reach a consensus, they can sign an agreement to finalize a contract that is stored in the Database. A reward system is provided through a basic concept of Virtual Money to prevent selfish Owners from utilizing services without providing any Caretaking service. Virtual Money is essentially service points added to a Caretaker. Virtual Money will be paid to a Caretaker once the service is completed and Admin user has verified feedback. The Pet Diaries team envisions that a corporate scale implementation of this project can convert Virtual Money to reward points or coupons with advertisements, which at this point is beyond the scope of this project. A feedback mechanism will be provided to allow dissatisfied Users to lodge negative feedback about quality of service provided by another User. Any User who has received more than a preset quantity of negative feedback will be blocked by the Admin.

To summarize, Pet Diaries hopes to make the life of a Pet Owner easier by directly introducing them to a Caretaker without spending any money out of their pocket.

Part II of project document describes structure and behaviour of project through different UML diagrams. Part II group section consists of

1) Project Requirements
2) Use case diagrams
3) UI mockups
4) Database requirements
5) Class diagram

1) Project requirements:

a) Business Requirements Table			
ID	Requirements	Topic Area	Actor
BR - 001	Create a website for Pet Diaries	Core functionality	NA
BR - 002	System will have three categories of actors - Admin,	Core functionality	NA

	owner, caretaker		
BR - 003	Create a communication network for owners and caretakers	Core functionality	NA
BR - 004	Creating 'virtual money" to compensate for caretaking hours	Core functionality	NA
BR - 005	Managing database that stores user information and processes	Core functionality	NA

| User Requirement table			
ID	Requirements	Topic Area	Actor
UR - 001	Can create a profile with username and password	Register user	Owner, Caretaker
UR - 002	Choose a role as owner or caretaker during profile creation	Register user	Owner, Caretaker
UR - 003	Give preference of pets	Register user	Caretaker
UR - 004	Choose number of pets and type of pets during profile creation	Register user	Owner
UR - 005	Add phone number, address and email address during profile creation	Register user	Owner, Caretaker
UR - 006	Set a password for authenticating login	Register user	Owner, Caretaker, Admin
UR - 007	Login with unique username and access respective dashboards	Register user	Admin, Owner, Caretaker
UR - 008	View new user profile requests on dashboard	Authenticate Profile	Admin
UR - 009	Approve or reject new user requests	Authenticate Profile	Admin
UR - 010	View all activities between users	Maintain system	Admin
UR - 011	Give feedback to Caretaker	complete transaction	Owner
UR - 012	Request contact information of other users	start communication	Owner, caretaker
UR - 013	Generate request	manage request	Owner
UR - 014	Accept request	start communication	caretaker
UR - 015	Respond to requests that turn up in their inbox based on preferences and location set while setting up profile	manage request	Owner, caretaker
UR - 016	Generate temporary chat box	start communication	Owner, caretaker

UR - 017	Delete any previous chat box	start communication	Owner, caretaker
UR - 018	Selectively block users based on negative feedback	complete transaction	Admin
UR - 019	Delete generated request	manage request	Admin, owner
UR - 020	Edit generated request	manage request	Owner
UR - 021	Profiles can be selectively deactivated	Maintain system	Admin
UR - 022	Requests can be selectively deleted	Maintain system	Admin
UR - 023	open chat box	start communication	owner
UR - 024	Send message	start communication	Owner, caretaker
UR - 025	Accept Agreement	start communication	Owner, caretaker
UR- 026	Review feedback by Owner	complete transaction	Admin
UR- 027	Assign Virtual Money	complete transaction	Admin
UR- 028	Use Virtual Money	complete transaction	Caretaker
UR - 029	Accept caretaker request	start communication	Owner
UR - 030	Validate user inputs	Register user	Admin, Owner, Caretaker
UR - 031	Reset username/password	Login	Admin, Owner, Caretaker
UR - 032	Send recovery email	Login	Admin, Owner, Caretaker
UR - 033	Login with unique username and access respective dashboards	Login	Admin, Owner, Caretaker
UR - 034	Lock user on too many login attempts with bad password	Login	Admin, Owner, Caretaker

c) Non-functional Requirement			
ID	Requirements	Topic Area	Actor
NFR - 001	Deactivate profiles not used for a specific period of time	Maintain system	Admin
NFR - 002	Delete old requests from database	Maintain system	Admin
NFR- 003	Decide on feedback to be positive or negative	complete transaction	Admin
NFR -004	Send email to user to activate profile	maintain system	Admin

2) Use case diagrams

Use case diagrams capture functionality of system. Diagram consists of internal and external agents as actors.
Diagram shows relationship of every actor with each use case in project.
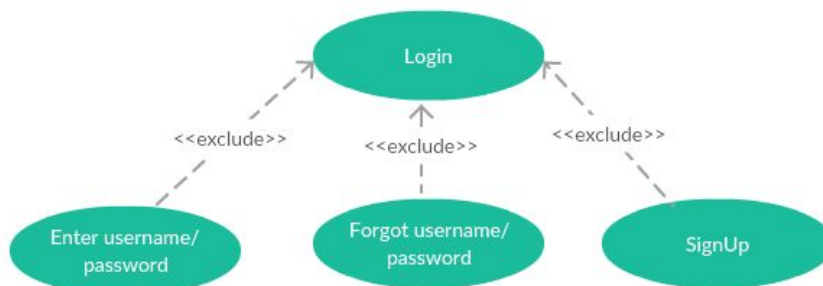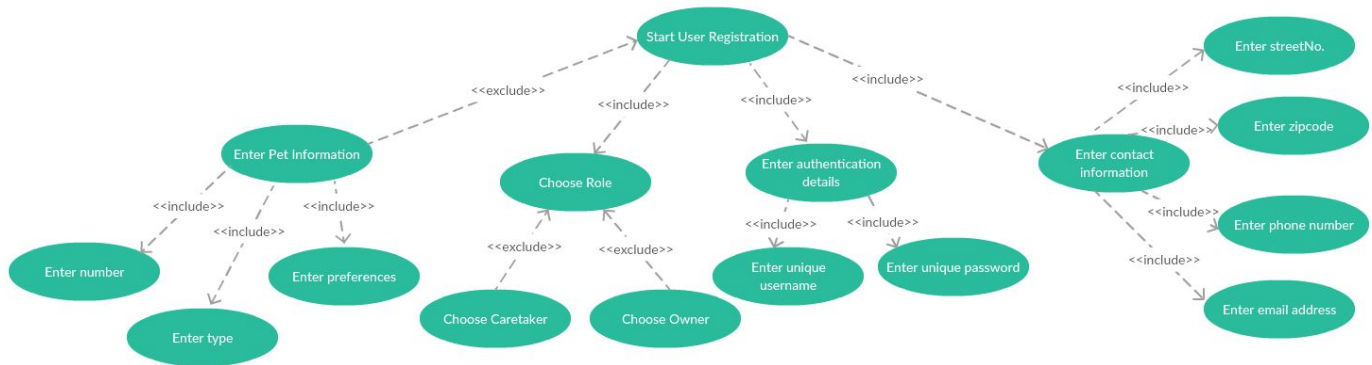Use case overview diagram:
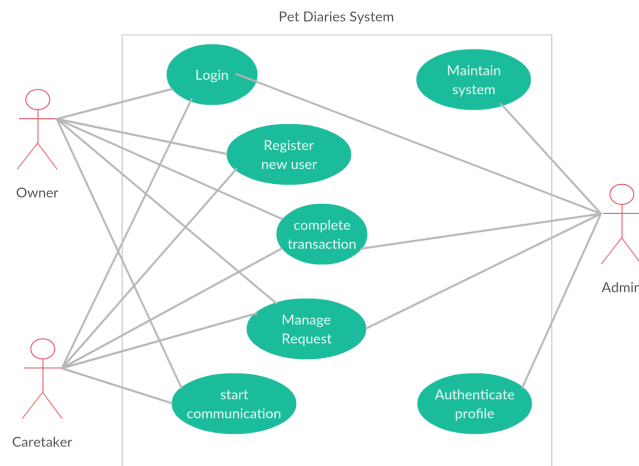
Use cases sub-diagrams:

I. UC - 01 Login

II. UC - 02 Start User Registration

III. UC- 03 start communication
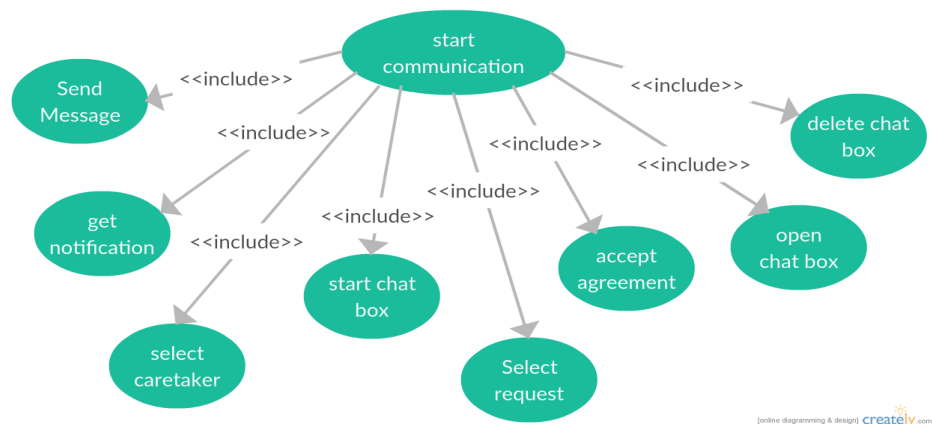
IV. UC-04 Manage Request
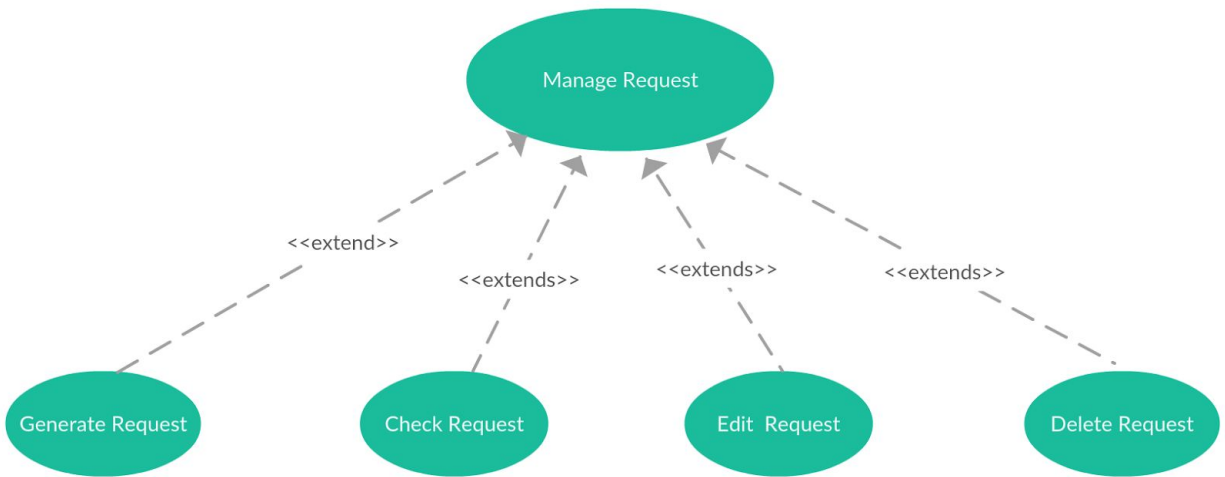
V. UC - 05 Authenticate Profile
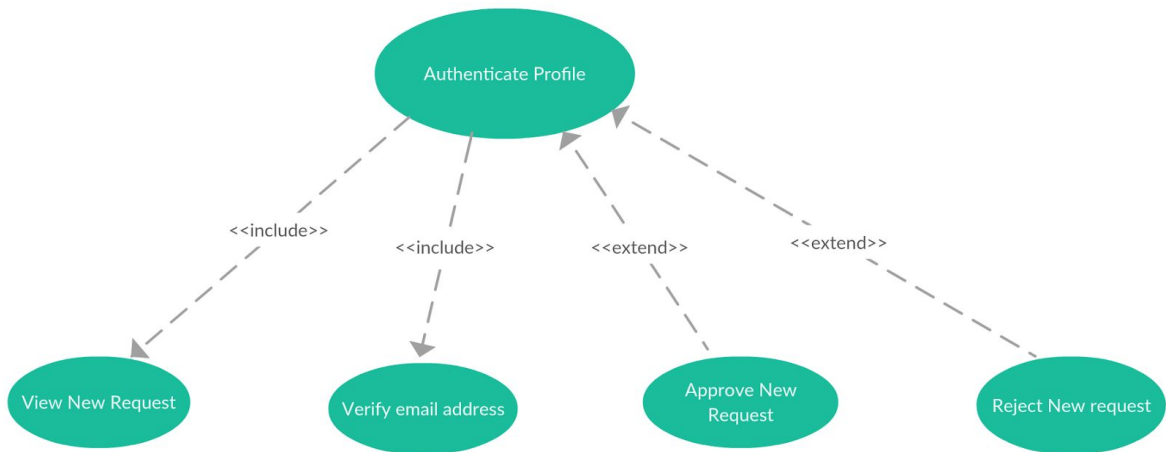
VI. UC- 06 Complete Transaction
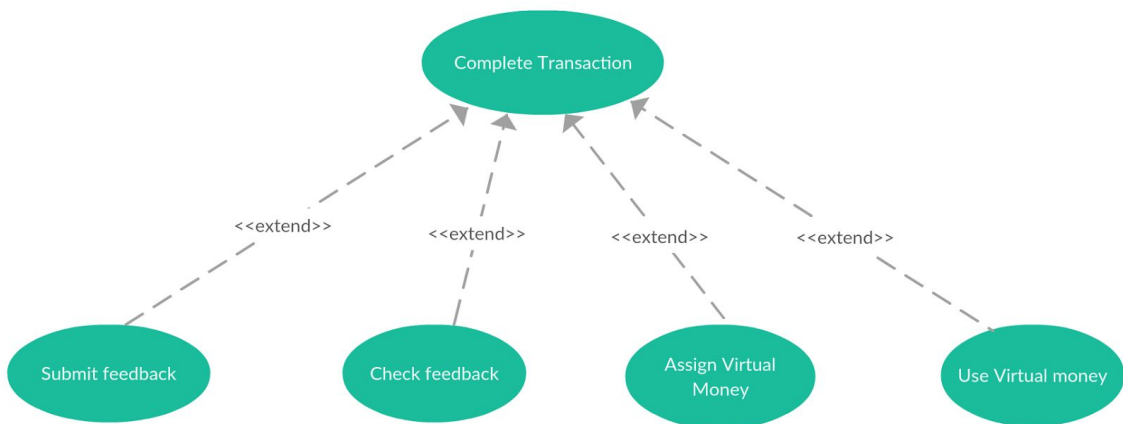
VII. C -07 Maintain System
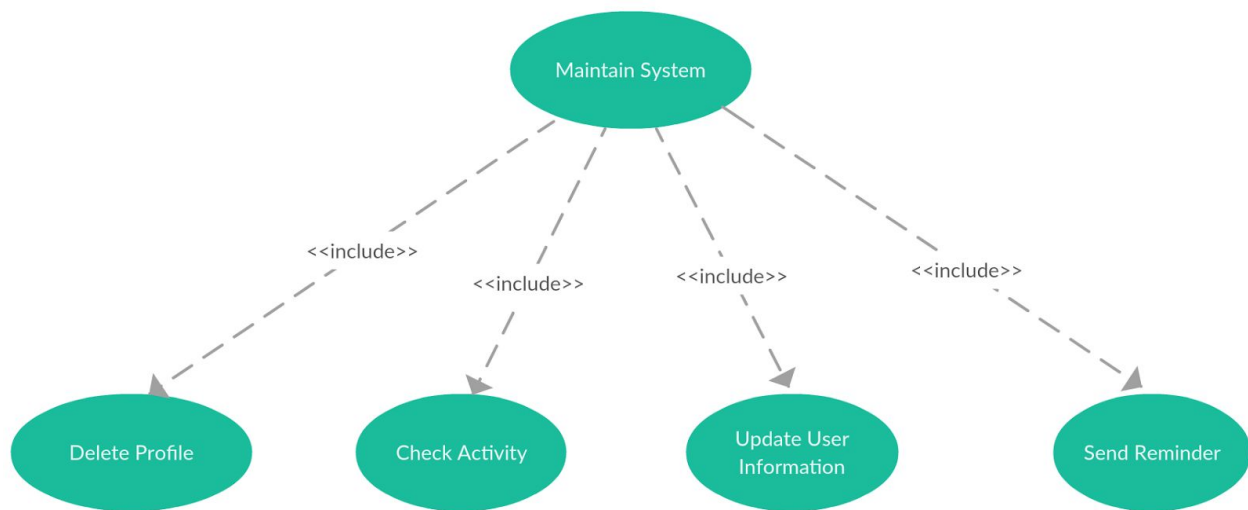
3) UI mockups:

User interface diagrams represent interface present for users to use application.

I.   Login screen

Login screen provides functionality to login to existing user account or generate new account. It also helps to recover forgotten username or password.
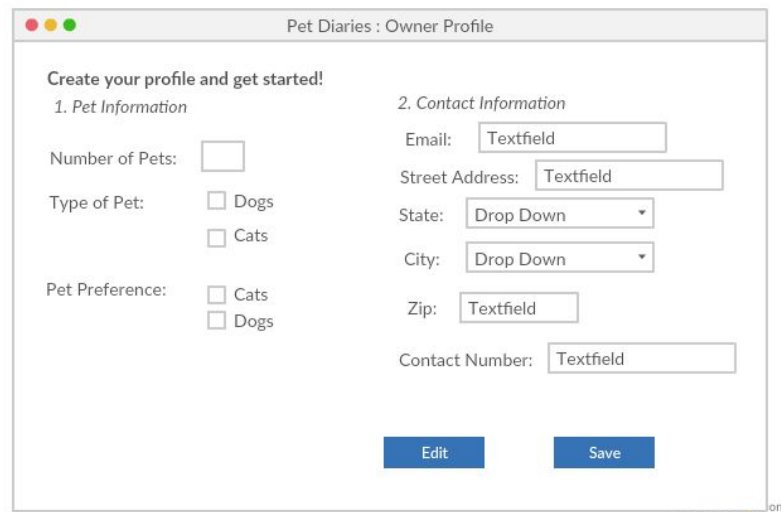
II. User profile

New user registration guides through sequence of screens required for signup such as registering address, contact information and choosing unique username. After registration is completed, users are directed to their

home screens. From home screen users can choose options to view notifications generated by system, generate request, view open communication interfaces and logout etc.

Owner Profile Creation:

Chat box

All open communication interfaces can be viewed under Send Message dialog box. Actors can choose to either open existing communication and delete communication.
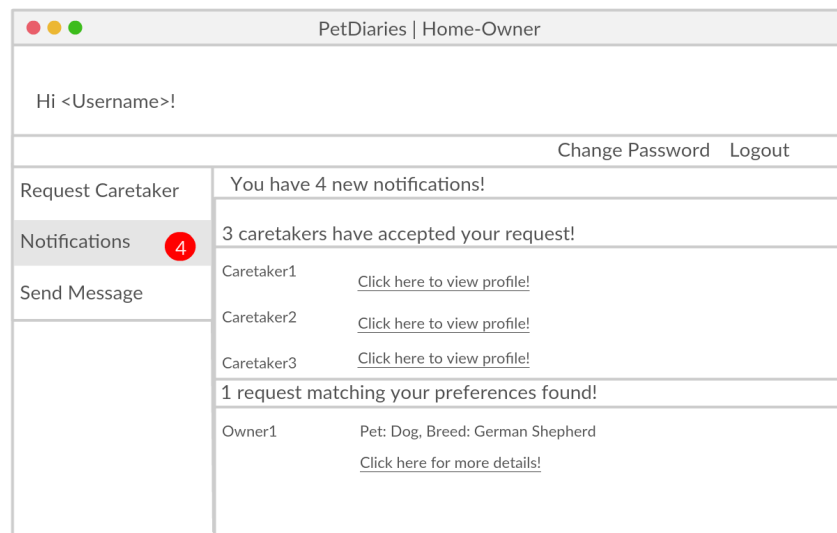
Hi <Username>!

Change Password   Logout

Request Caretaker

Notifications

Send Message

Owner 1   Delete   Open
Owner 2   Delete   Open
Owner 3   Delete   Open

Communication interface provides functionality to view received messages, send new message and accept agreement for request generated by owner.

Accept Agreement

Owner1 : Text

Caretaker1 : Text

Owner1 : Text

Textfield   Send

4) Database requirements

Database used: MySQL
Objects: Communication Interface, Contact information, User, Admin, Owner, Caretaker and Request object.

- At the time of user registration, login information such as username and password, along with System generated unique user ID is stored in database.
- When user creates profile, the profile data is stored in the database.
- Once admin verifies new user request and approves the same, Admin user will update the user as a permanent user in the database.
- After a request has been generated by the Owner, it is stored in the database and a unique request ID is created for it in the database.
- This request ID is used for editing, publishing, accepting and deleting the request.
- The feedback obtained after the request has been serviced is saved in the database corresponding to the request ID previously allocated and the userid of Owner and Caretaker
- Virtual money is credited to the Caretaker's database entry based on his userID once request is serviced and admin has verified feedback. Also, same amount is deducted from owner's user ID. This is debited when the user redeems his points.

- Once a request in generated by owner, system will send notifications to actors with same location details in database.
- When owner requests to generate communication interface, system will generate communication interface and assign it interface ID. With this interface ID actors can communicate with each other by sharing interface. Delete communication interface removes entry related to that interface ID.
- Once the agreement is established by both actors, entry regarding request ID is generated to finalize dates of request and mark them to generate virtual money.

5) Class diagram

<<Interface>> ManageDB

+ addToDb(infotype: String, info :String) : void
+ deleteFromDb(infotype:String) : boolean
+ editDb(infotype: String, info :String) : void
+ checkDb(infotype:String) : String
+ deleteProfile(userID: userId) : void
+ notifyAdmin(infoType:String):void

User

\# userId : int
\# username : String
\# password : String
\# type : userType
- infoMsg : String
\# virtualMoney: int
\#posneg : bool
\#loginTime: Date

+ getUserId() : int
+ getUsername() : String
+ getUserType() : userType
+ validateUser(username: String, password: String): bool
+ userSignUp(): void
+ userLogin(username: String, password: String): void
+ forgotNamePassword(): void
+ populateUserInput(): void
+ getValidity(emailAddr: String): bool
+ getValidity(username: String, password: String): bool
+ getValidity(contactInfo: ContactInfo, username: String, password: String): bool
+ sendResetEmail(emailAddr: String)
+ displayError(infoMsg: String): void
+ displaySuccess(infoMsg: String): void
+ displayDash(userId: int): void
+ displaySignupPage(): void
+ userRegistration(): void
+ createUser(contactInfo: ContactInfo, username: String, password: String): bool
+ submitData(contactInfo: ContactInfo, username: String, password: String): void
+ login(type: userType, userId: int): void
+ connectToContact(emailId:String) : void
+ deleteUserProfile(contactInfo) : void

<<Enumeration>> userType

ADMIN
OWNER
CARETAKER

CaretakerProfile

\# Preference : petType
- Interface_ID : int[ ]

+ setUserInfo(user:User) :int
+ setContactInfo(contactInfo :ContactInfo),userID:int : void
+ useVirtualMoney(userID:int, virtualMoney:int) ; int
+ acceptRequest(requestID: int) : void

+ openChatBox(interfaceID:int): void
+ checkOpenChatBox(username:User):int
+ startChatBox(username:User):void
+ deleteChatBox(interfaceID: int) : void
+ acceptAgreement(requestID:int) : void
+ sendMessage(msg:String):void
+ IsagreementComplete(requestID:int):int

OwnerProfile

- noOfPets : int
- petType : petType
\# preference : enum
- interface_id : int[ ]

+ setUserInfo(user:User) :int
+ setContactInfo(contactInfo: ContactInfo,userID:int) : void
+ generateNewRequest(newRequest: Request) : requestID
+ editRequest(requestID: int) : void
+ deleteRequest(requestID: int) : void
+ submitOwnerFeedback(feedback: String,requestID: int) : void
+ acceptRequest(requestID: int) : void
+ selectCaretaker(username:User, requestID: int) : void
+ openChatBox(interfaceID:int): void
+ checkOpenChatBox(username:User):int
+ startChatBox(username:User):void
+ deleteChatBox(interfaceID: int) : void
+ acceptAgreement(requestID:int) : void
+ IsagreementComplete(requestID:int):int
+ sendMessage(msg:String):void

AdminProfile

+ viewActivity() : void
+ updateUserLogin(userID: int, loginTime:Date) : void
+ sendActivityReminder(UserID:int) : void
+ deleteUserProfile(UserID:int) : void
+ validateFeedback(requestID:int): bool
+ giveVirtualMoney(userID: int) : int
+ viewRequest(requestId: int :void
+ deleteRequest(requestId: int):void
+ viewNewProfileRequest() :userID
+ verifyEmail(email:String) : void
+ acceptNewProfile(UserID:int) : void
+ rejectNewProfile(UserID:int) : void

Request

\# no_of_pets : int
\# pet_type : petType
\# date : Date
\# requestId int
\#feedback: String
\# validreq: bool

+ publishRequest(requestID: int): void
+ notifyAcceptedRequest(requestID: int) : void
+ validateRequest(requestID:int) : bool
+ notifyInvalidRequest(void): void
+ deleteOldRequests(void): void
+ sendFeedback(feedback:String, reuestID,int): void

<<Enumeration>> petType

DOG
CAT

ContactInfo

\# street : String
\# city : String
\# phoneNumber : String
\# zipcode : String
\# emailAddr : String

+ getStreet() : String
+ getCity() : String
+ getPhoneNumber() : String
+ getZipCode() : String
+ getStreet() : String
+ getEmailAddr(): String

CommInterface

\# interfaceId : int
\# ownerId : int
\# caretakerID: int

+ agreementComplete(requestID:int):void

DataBase

text