MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

PET DATING

VERSION 1.0

Supervisor	**Phan Duy Hùng**
Group Members	**Nguyễn Đức Thịnh – SE06149** **Nguyễn Đỗ Anh Khương – SE05608** **Khương Trung Hiếu– SE06020** **Trịnh Văn Anh – SE05308** **Nguyễn Hoàng Phong – SE05668**
Capstone Project code	**PD**

Ha Noi, May 31, 2020

Contents

CHAPTER 1: INTRODUCTION

1.1. The problem

Most of people who own pets always care about life of their pets such as food, morale and mates. However, the network where puppies can find a friend or partner is very rare in Viet Nam. Facebook – a social network that is popular had a lot of groups about this problem. But in there,

we also lack of specific features that focus in users that have pets. Some other communities are not really popular or have stopped working. In conclusion, there are not really any online place or community for people that have pets in Vietnam right now.

1.2. Existing solutions

We consulted some websites/application about pet before give our proposal. Each website/application has a lot of advantages but still has some limitations.

- **App Get Bone**

Figure 1 - Get bone (app)

Advantages	Disadvantages
• Provide news and more information about pets. • Interface is easy to use and be like Tinder • Support finding mates.	• Only news, no communication between users

- **Website Datemypet.com**

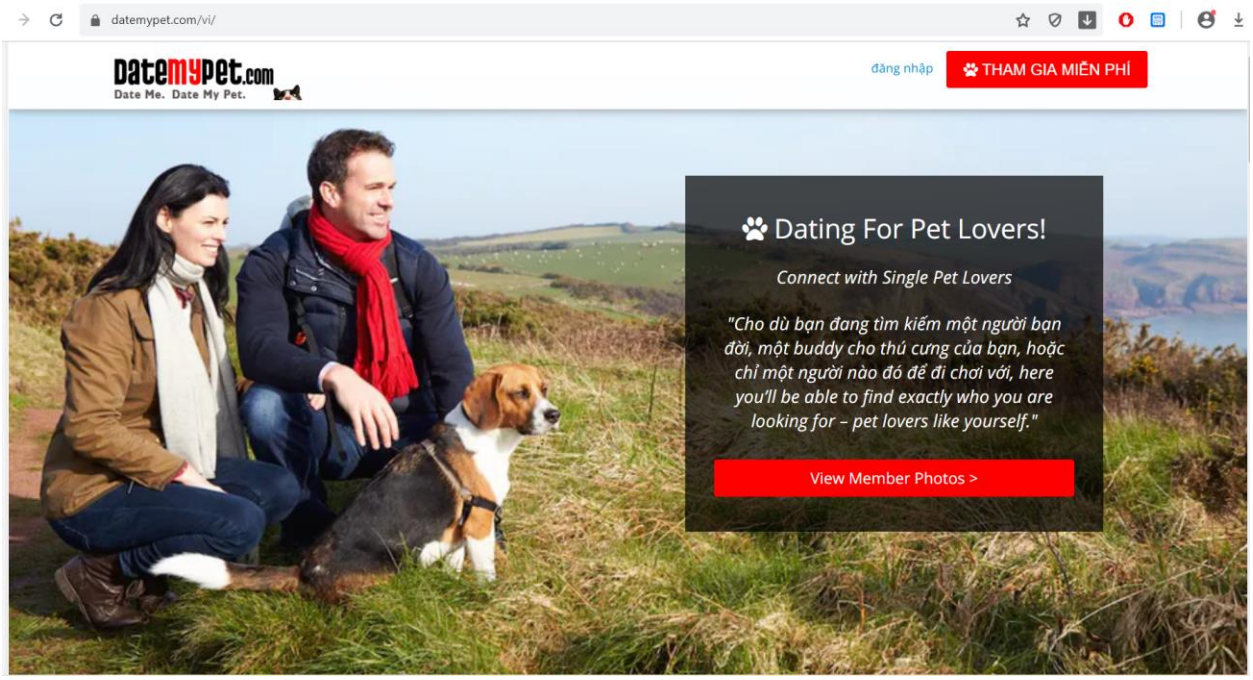


Figure 2 - Datemypet.com

Advantages	Disadvantages
• Provide news and more information about pets.	• Not support finding mates. • Only news, no communication between users

1.3. Proposal

1.3.1. The idea

We want to build a mobile application where everyone can find the half of their pets, or simply want to know about other pets. Major of application is finding the mates for their pets based on automation mode or manual mode. In the app, people maybe complete the user's profile; pet's profile; upload images; match what pets they want. In addition, everyone can connect and chat about pets. It helps making pets dating becomes easy, safe and enhance the community. Our idea has two part: mobile application for user and website for manager.

1.3.2. The Proposal

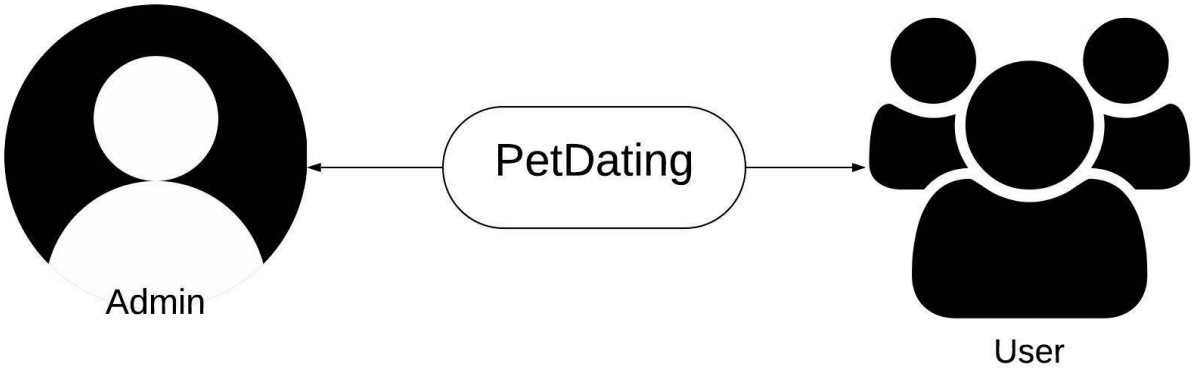


Figure 2 - Roles in PD system

1.3.3. Business flow

- **Review report process**

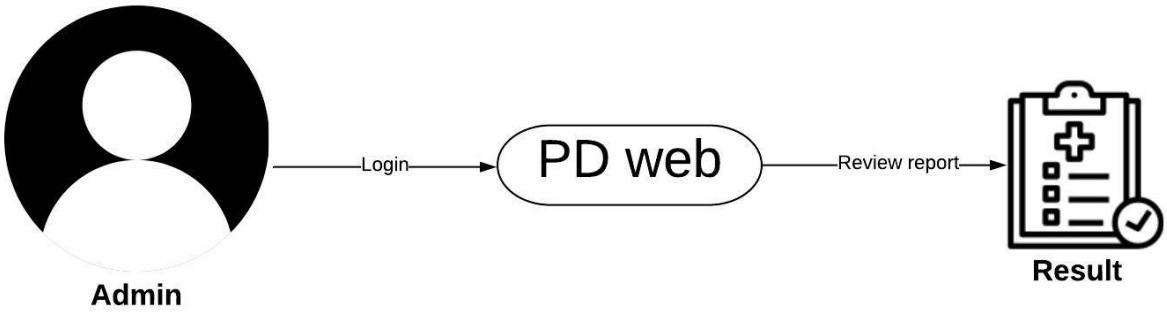


Figure 4 - Process of review report on PD

- **Ban and unban user**

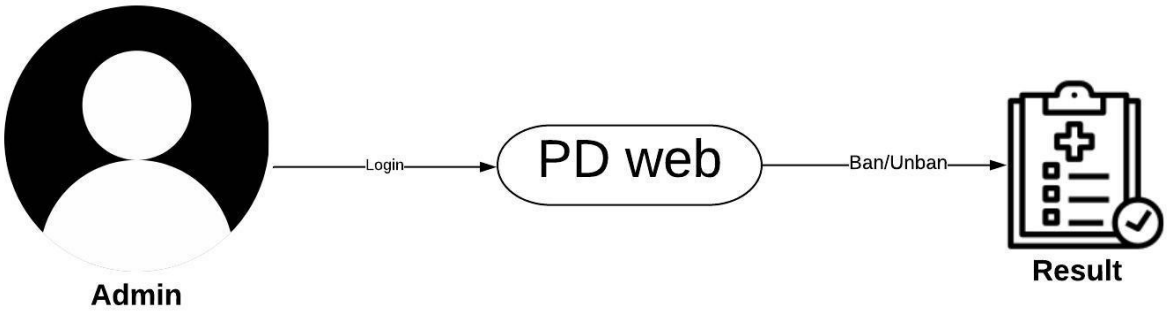


Figure 5 - Process of ban/unban on PD

- **Dating process**

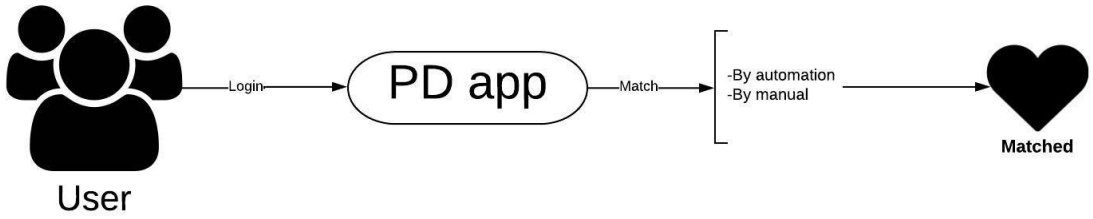


Figure 6 - Process of Dating on PD

❖ **User management process**

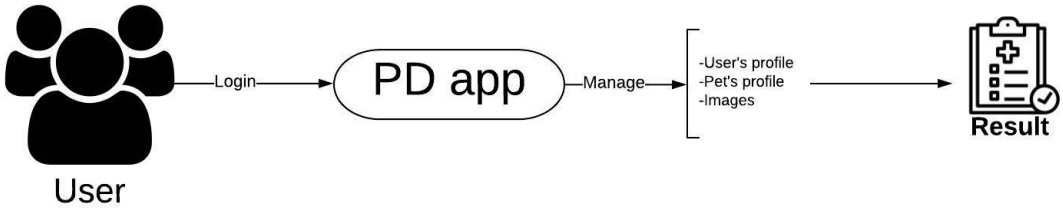


Figure 7 – Process of user management

CHAPTER 2: SOFTWRE PROJECT MANAGEMENT PLAN (SPMP)

2.1. Purpose

Project plan describes the software process model, team organization and management plan of the project. All team members must follow this section as a guideline to complete assigned tasks and deadline.

2.2. Software Process Model

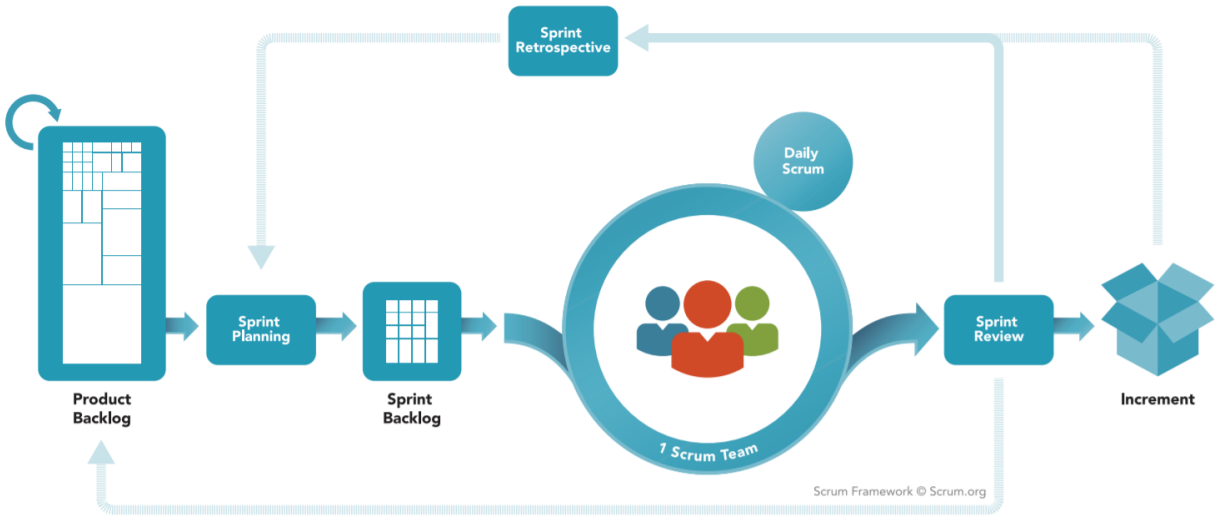


Figure 8 - Agile Model – SCRUM

2.2.1. About the SCRUM

SCRUM is Agile software development life cycle (SDLC) models. Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. And it is a simple framework for effective team collaboration on complex products.

- **Product Backlog**: is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product.
- **Sprint Planning**: The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum Team.
- **Sprint Backlog**: is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a “Done” Increment.
- **Sprint**: The heart of Scrum is a Sprint, a time-box of one month or less during which a “Done”, useable, and potentially releasable product Increment is created. Sprints

have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

Reference: "The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game" – Developed and sustained by Scrum creator: Ken Schwaber and Jeff Sutherland – November 2017.

2.2.2. Advantages and disadvantages of SCRUM

Advantages	Disadvantages
• Works well for fast-moving development projects. • The team gets clear visibility through scrum meetings. • Removing mistakes or rectifying them is considerably easy. • It is iterative in nature and needs continuous feedback from the user for the betterment of the process.	• Scrum often leads to scope creep, due to the lack of a definite end-date. • The chances of project failure are high if individuals aren't very committed or cooperative. • If any team member leaves in the middle of a project, it can have a huge negative impact on the project. • Daily meetings sometimes frustrate team members

2.3. Project Organization

2.3.1. Organization structure

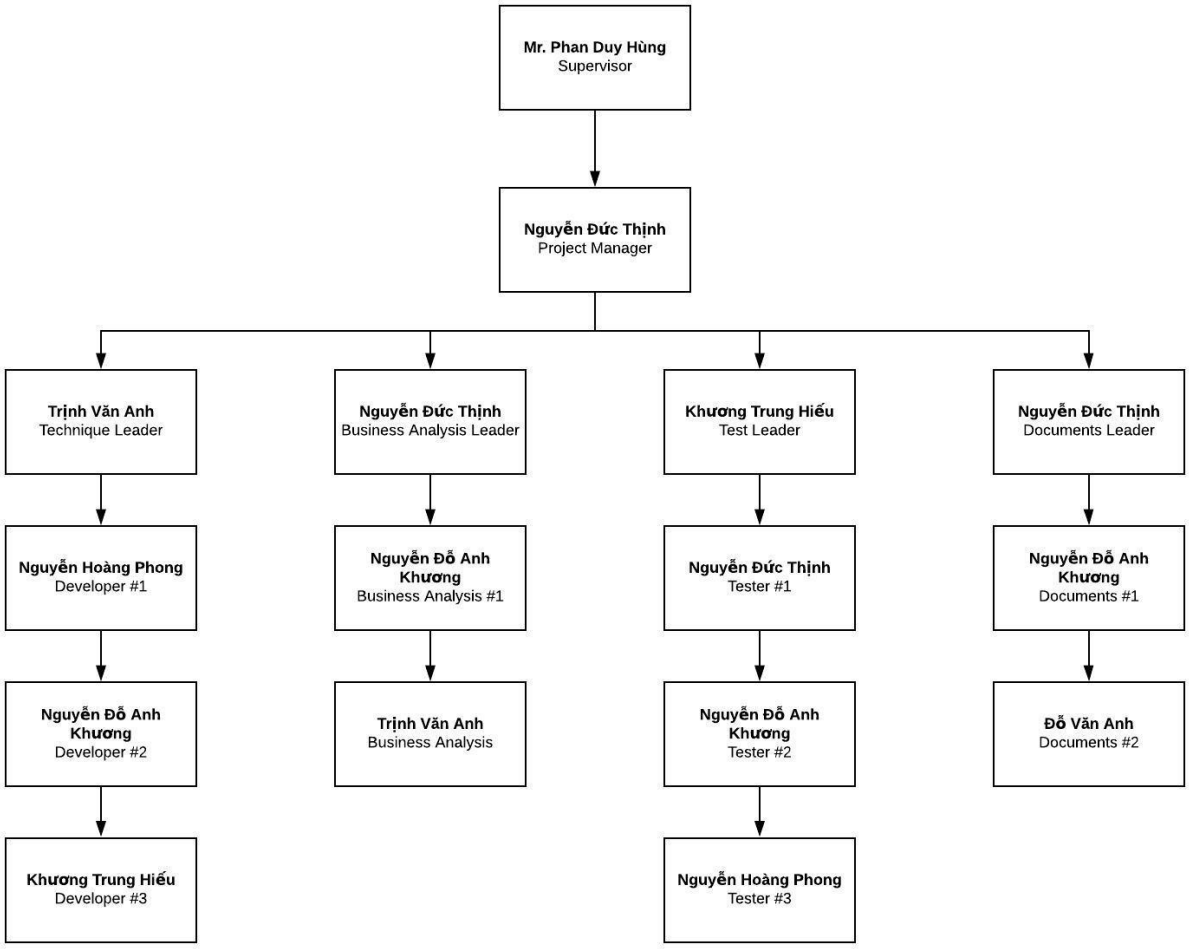


Figure 3 - Organization structure

2.3.2. Roles and responsibilities

Role	Responsibilities	Full Name
Project Manager		
Project Manager	• Guide team toward the goal of successfully passing the final capstone project. • Develop schedule and assigning task with responsibilities for each member. • Communicate with all teams to keep them focusing on the final goal.	Nguyễn Đức Thịnh

	• Propose ideas or issues of the team to supervisor, university.	
Business analyst team		
BA leader	• Elicit and analyze requirement. • Define scope and create SRS template.	Nguyễn Đức Thịnh
BA #1	• Design entity relationship diagram. • Define business process flow and object state.	Nguyễn Đỗ Anh Khương
BA #2	• Capture and specific describe use case.	Trịnh Văn Anh
Developer		
Technical leader	• Define high level architecture base on SRS. • Implement configuration and server. • Design and code dating function. • Develop client functions in console system. • Design and review database. • Design mockup for application.	Trinh Văn Anh
Dev #1	• Lead chatting function. • Design a part of database. • Design and code chatting function. • Code others screen. • Design mockup for application.	Nguyễn Hoàng Phong
Dev #2	• Design and code dating function. • Code others screen. • Design mockup for application.	Nguyễn Đỗ Anh Khương
Dev #3	• Design and code chatting function. • Code others screen. • Design mockup for application.	Khương Trung Hiếu
Tester		

Test leader	• Create template testing documents. • Define test strategy, create test plan and defect log temple.	Khương Trung Hiếu
Test #1	• Implement test case and log defect.	Nguyễn Đức Thịnh
Test #2	• Implement test case and log defect.	Nguyễn Đỗ Anh Khương
Test #3	• Implement test case and log defect.	Nguyễn Hoàng Phong
Documentation		
Docs leader	• Prepare all the documentation relating to the project. • Give format in documentation.	Nguyễn Đức Thịnh
Docs #1	• Check spelling errors and grammar errors.	Nguyễn Đỗ Anh Khương
Docs #2	• Check spelling errors and grammar errors.	Đỗ Văn Anh

2.4. Tools and Techniques

Tool and techniques	Version	Description
Visual Studio Code	Lastest	Text editor for coding, testing
Github	Lastest	Git GUI for controlling source code and version easily
NodeJS	10.16.3 LTS	Environment for application
React-native-cli	2.0.1	Environment for developing React Native app
Google Drive	Online	Documents, resources storage
Lucidchart	Online	Diagram, UML, chart creation tool
Draw.io	Online	Diagram, UML, chart creation tool
Microsoft Project	2019	Plan management and schedule
Microsoft Office	2019	Documentation tool includes: Words, Excel, PP, …

Trello	Online	Control task, backlog
Facebook	Online	Communication tool
Skype	Lastest	Communication tool
Google Meet	Online	Video conferencing, presentation
Heroku Host	Online	Online hosting service
Heroku Storage	Online	Cloud storage for Firebase
Google Firebase Realtime Database	Online	Database for performing realtime function

2.5. Project Management Plan

2.5.1. Project schedule

The tasks list detail is described in file: **"Plan.mpp"**

2.5.2. Meeting minutes

Project Name/Code	PetDating	**Date of meeting**	
Conductor	Nguyễn Đức Thịnh	**Location**	
Topic of meeting			
Attendees:			
Name	**Role**	**Attendance**	
Phan Duy Hùng	Supervisor	Present	
Nguyễn Đức Thịnh	PM	Present	
Nguyễn Hoàng Phong	Team member	Present	
Nguyễn Đỗ Anh Khương	Team member	Present	
Trinh Văn Anh	Team member	Present	
Meeting objective:			
➢			

2.5.3. Risk management plan

***Risk identification**

No	Risk description	Contingency/Mitigation plan	Probability	Impact
1	Scope of project was defined poorly that cause ambiguous for team members	Meeting with BA expert	Medium	High
2	Requirements changes during project time	Meeting and redefine objective with each team member.	Medium	High
3	Failure in estimating sprint time, inadequate change	Doing overnight work to keep process continuing	High	Medium
4	Confliction among team members	- Transfer problems to whole team to resolve it. - Create happy and friendly environment among team members.	Low	Medium
5	Library or third-party features used in project is no longer supported	Choose alternative library or coding from scratch	Low	High
6	Illness or absence of team members	Ensure that the absence of a member will not affect the plan and schedule. Always have plans to deal with this problem	Low	Low

***Probability – Impact matrix**

Probability	High		R3	
	Medium			R1, R2
	Low	R6	R4	R5
		Low	Medium	High
		Impact		

2.5.4. Communication plan

Project report and meeting:

No	Activity	Stakeholder	Time	Description
1	Weekly report	- Team member - Supervisor	Monday	- Report in meeting minutes - Review status based on weekly report - Plan for the next week
2	Daily report	Project manager	Daily	- Report issue and planning - Support to solve issues
3	Ad-hoc report	Team members	Event-based	- Report and talking face to face

Project communication:

No	Style	Description
1	Weekly meeting schedule	Spend one day in a week for meeting to review and update solution for problems.
2	Unscheduled meeting	If any member has problems that he wants to solved immediately, we will have a chat box.

3	Communication channel	Our main communication is physical meeting, chat box, Facebook messenger.

2.5.5. Coding convention

The following coding conventions are used to code the project to help improve readabilities, maintenance abilities, and code more easily.

#	Style	Implementation component
1	NodeJS coding convention	Service API
2	React coding convention	React native convention
3	RESTful API	REST convention

CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATION

3.1. Purpose

This chapter outlines functional and non-functional requirements of our system. It also provides some format constraints in common requirements and project success criteria. The content of this chapter is used as the basis for the work in the subsequent chapters.

3.2. Functional Requirement

3.2.1. Use case diagram

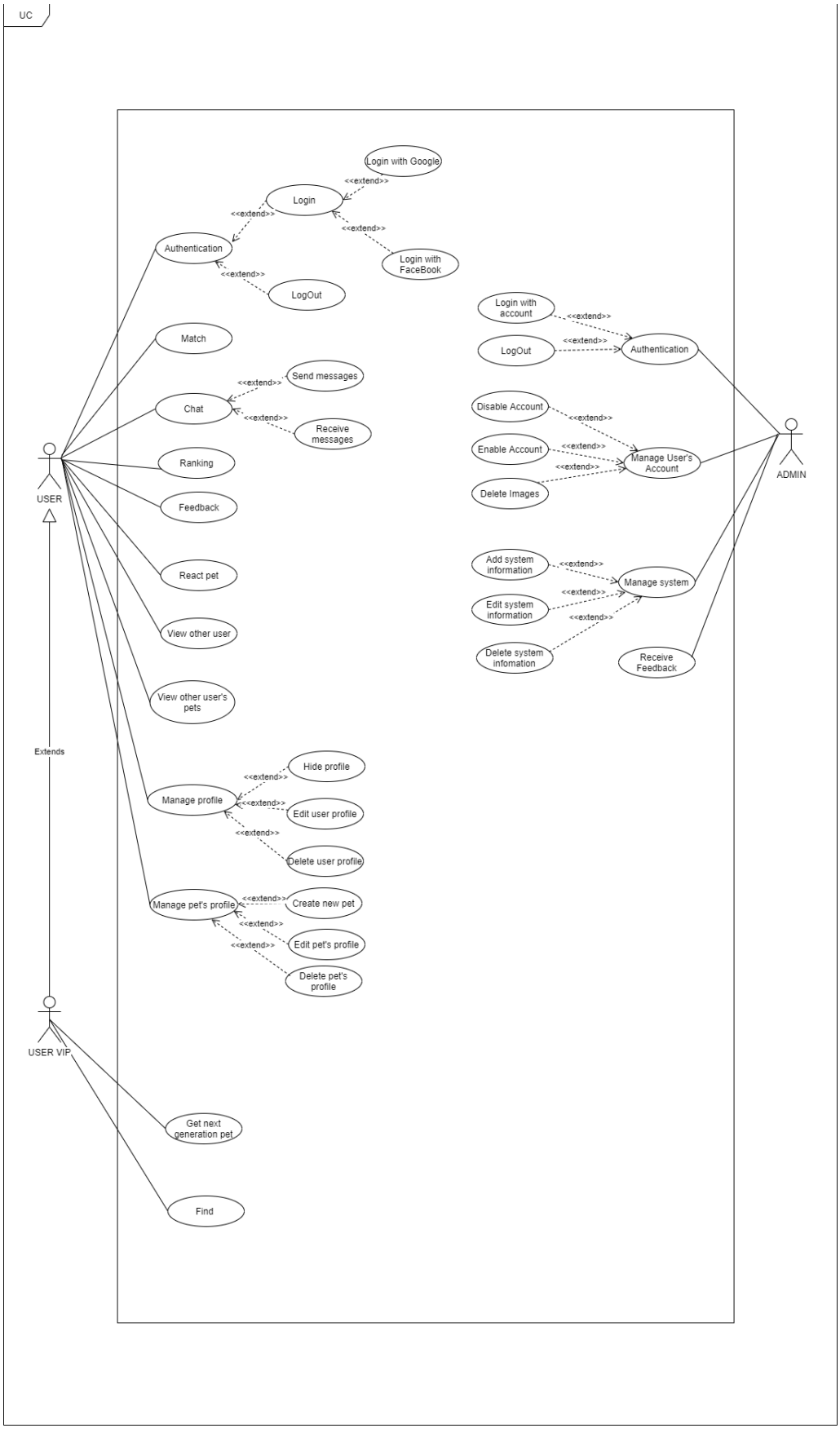

UC
Login with Google
<<extend>>
Login
<<extend>>
<<extend>>
Authentication
Login with FaceBook
<<extend>>
LogOut
Match
Send messages
<<extend>>
Chat
<<extend>>
Receive messages
Ranking
Feedback
React pet
View other user
View other user's pets
Extends
Hide profile
<<extend>>
Manage profile
<<extend>>
Edit user profile
<<extend>>
Delete user profile
Manage pet's profile
<<extend>>
Create new pet
<<extend>>
Edit pet's profile
<<extend>>
Delete pet's profile
USER
USER VIP
Get next generation pet
Find
Login with account
<<extend>>
LogOut
<<extend>>
Authentication
Disable Account
<<extend>>
Enable Account
<<extend>>
<<extend>>
Manage User's Account
Delete Images
Add system information
<<extend>>
Manage system
<<extend>>
Edit system information
<<extend>>
Delete system infomation
Receive Feedback
ADMIN

Figure 10 – Use case diagram

.2.2. Business rules

ID	Description
1	Google+ account or Facebook account must be validated.
2	Admin's password must not be stored as plain text. Instead it must be hashed using a secure hash algorithm.
3	The field must be filled by characters consist of alphabet and numbers.
4	The field must be filled by characters consist of alphabet.
5	The field must be not empty.
6	The characters of field are greater than 10.
7	The characters of field are greater than 3.
8	The characters of field are greater than 5.
9	The maximum characters of field are 30.
10	The maximum characters of field are 20.
11	Image file type must be image type.

3.2.3. Use case list

Actors	Description
User	Everyone who has an account on the PetDating system and used it.
Admin	People who has responsibilities about system

ID	Actor	Name
AD-UC-1.0	Admin	Login with account
AD-UC-2.0		Logout
AD-UC-3.0		Disable Account
AD-UC-4.0		Enable Account
AD-UC-5.0		Delete Images
AD-UC-6.0		Manage system

AD-UC-7.0		Receive Feedback
US-UC-1.0	User	Login with Goole/Facebook/(Phone number)
US-UC-2.0		Logout
US-UC-3.0		Match
US-UC-4.0		Send messages
US-UC-5.0		Receive message
US-UC-6.0		Ranking
US-UC-7.0		Feedback
US-UC-8.0		React pet
US-UC-9.0		View other user
US-UC-10.0		View other user's pets
US-UC-11.0		Hide profile
US-UC-12.0		Edit user's profile
US-UC-13.0		Delete user's profile
US-UC-14.0		Create new pet
US-UC-15.0		Edit pet's profile
US-UC-16.0		Delete pet's profile
US-UC-17.0		Find
US-UC-18.0		Get next generation pet

3.2.4. Use case specification

3.2.4.1. Admin

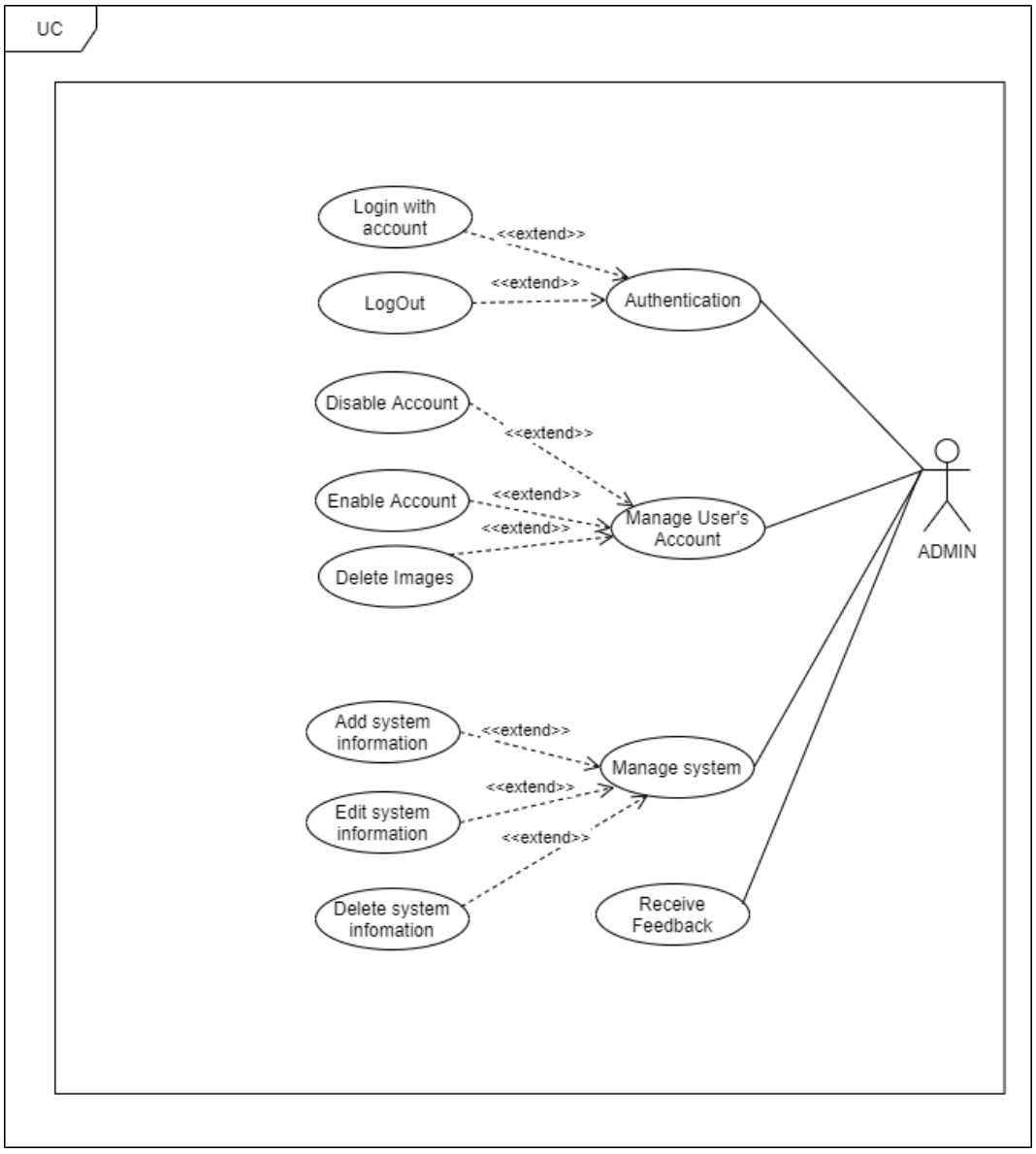


Figure 11 - Use case diagram of Admin actor

3.2.4.1.1. Authentication

***Login with account**

USECASE AD-UC-1.0 SPECIFICATION			
Use case ID	AD-UC-1.0	**Use case version**	V1.0
Use case name	Login with account		

Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Allows admin login to PD system in website.		
Pre-condition	Admin has an account and the account is still working.		
Post-condition	When the normal flow completes successfully, the login status is set to true, and the access token is saved on the website.		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Admin goes to login page	
2.		Website displays the login form
3.	Admin fills account and password then clicks “Login” button.	
4.		System generates cookies and save them.
5.		Website navigates to homepage.

Alternative Flows: Admin logged and not yet log out.

No.	Actor events	System respond
1.	Admin goes to homepage	
2.		Website checks cookies then navigates to login page if not exist cookies.

Extension Flows: N/A

Exception:

AD1.0-E1 – Cannot connect with Account API

System displays error message.

AD1.0-E2 – Wrong account or password

System displays error message and requests login again.

Business Rules: B3, B4

*Logout

USECASE AD-UC-2.0 SPECIFICATION			
Use case ID	AD-UC-2.0	**Use case version**	V1.0
Use case name	LogOut		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Log out the PD System.		
Pre-condition	Has logged into PD system.		
Post-condition	When the normal flow completes successfully, the login status is set to false, and the access token is cleared.		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Clicks icon “Account” on top right of website.	
2.		Website displays “Account Management” pop-up
3.	Admin/Mod clicks “Logout” on top right of website.	

4.		Website navigates to Login page.

Alternative Flows: N/A

Extension Flows:

Exception: N/A

Business Rules: N/A

*__Disable Account__

USECASE AD-UC-3.0 SPECIFICATION			
Use case ID	AD-UC-3.0	**Use case version**	V1.0
Use case name	Disable Account		
Author	Nguyen Do Anh Khuong		
Date	16/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Allows admin to disable account of user (ban)		
Pre-condition	Admin logged in to system.		
Post-condition	User's Account was banned		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Clicks icon "Manage Profile" on top of website.	
2.		Website displays "Manage Profile " screen

3.	Choose account wanna ban and click on disable button	
4.		Website show confirm dialog on screen
5.	Fills all require field and presses the "Confirm" button	
6.		Shows sucess message

Alternative Flows: N/A

Extension Flows: N/A

Exception: N/A

Business Rules: N/A

*__Enable Account__

USECASE AD-UC-4.0 SPECIFICATION			
Use case ID	AD-UC-4.0	**Use case version**	V1.0
Use case name	Enable Account		
Author	Nguyen Do Anh Khuong		
Date	16/07/2020	**Priority**	Medium
Primary actor	Admin	**Secondary actor**	N/A
Description	Administrator Enable user's Account.		
Pre-condition	Admin has logged into system by admin account. Account have been banned		
Post-condition	User's Account was Enable (unbanned)		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Admin clicks “Manage Profile” on left-menu	
2.		Website navigates to “Manage Profile” page
3.	Choose account wanna Enable and click on “disable/enable” button	
4.		Website show “confirm” dialog on screen
5.	Fills all require field and presses the “Confirm” button	
6.		Shows sucess message

Alternative Flows: N/A

Extension Flows: N/A

Exception:

AD6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

*__Delete Image__

USECASE AD-UC-5.0 SPECIFICATION			
Use case ID	AD-UC-5.0	**Use case version**	V1.0
Use case name	Delete Image		
Author	Nguyen Do Anh Khuong		

Date	16/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Administrator can View and detele image of User's Account.		
Pre-condition	Admin has logged into system by admin account.		
Post-condition	Image have been deleted		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Admin clicks "Account Manager" on left-menu	
2.		Website navigates to "Account manager" page
3.	Admin can view , and search account was feedbacked	
4.		Website show information of user
5.	Admin change information and Admin clicks "detele" beside image	
6.		Website reloaded and show new information

Alternative Flows: N/A

Extension Flows: N/A

Exception:

AD6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

*Manage system

USECASE AD-UC-6.0 SPECIFICATION			
Use case ID	AD-UC-6.0	**Use case version**	V1.0
Use case name	Manager system		
Author	Nguyen Do Anh Khuong		
Date	16/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Administrator can manage information of system		
Pre-condition	Admin has logged into system by admin account.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	Admin clicks “Manager” on left-menu	
2.		Website navigates to “Manager” page
3.	Admin choose “system manager”	
4.		Website shows “System Manager” screen
5.	Admin can view , change information (add , edit ,delete)	
6.		Website reloaded and show new information

Alternative Flows: N/A

Extension Flows: N/A

Exception:

AD6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

*__Receive Feedback__

USECASE AD-UC-7.0 SPECIFICATION			
Use case ID	AD-UC-7.0	**Use case version**	V1.0
Use case name	Manager system		
Author	Nguyen Do Anh Khuong		
Date	16/07/2020	**Priority**	High
Primary actor	Admin	**Secondary actor**	N/A
Description	Administrator see all of feed back from user		
Pre-condition	Admin has logged into system by admin account.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	Admin clicks “Feedback” on left-menu	
2.		Website navigates to “FeedBack” page
3.		Website show list all of feedback

Alternative Flows: N/A

Extension Flows: N/A

Exception:

AD6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2. User

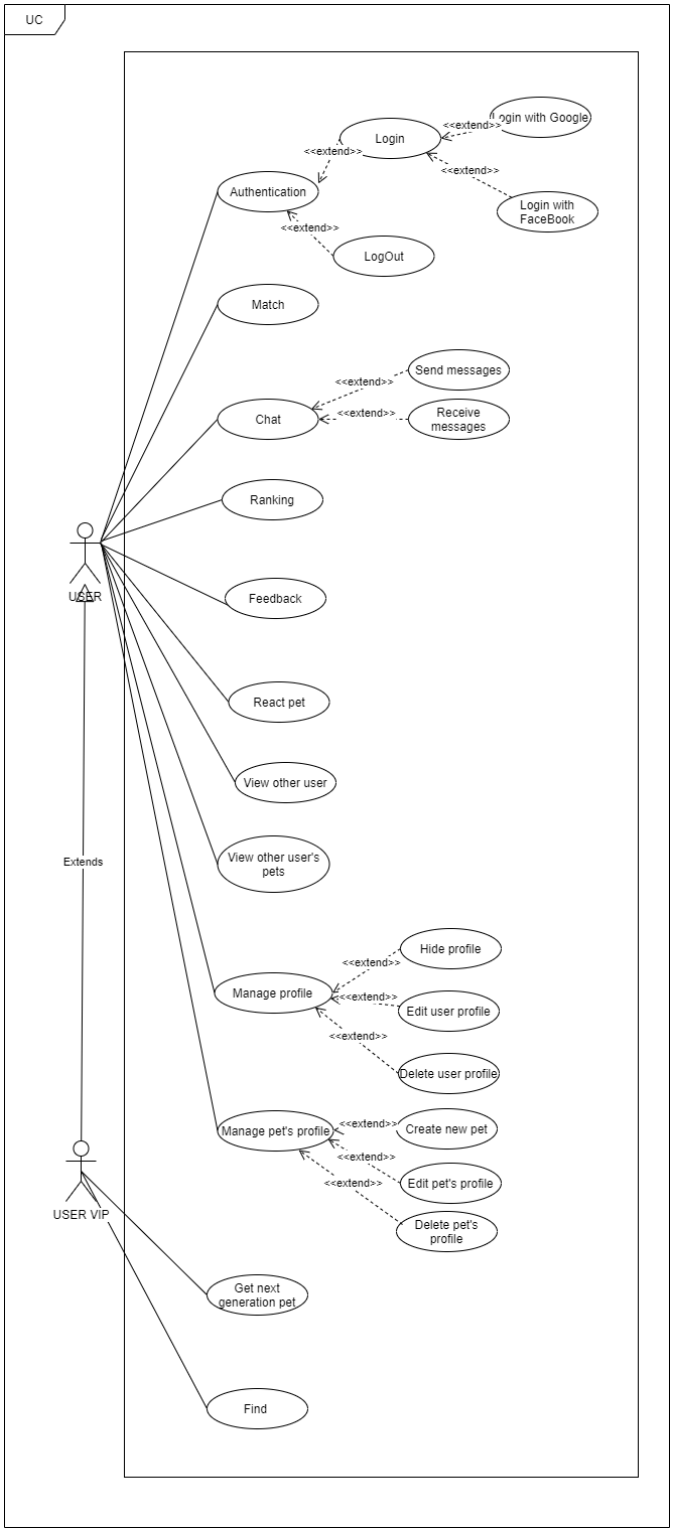


Figure 12 - Use case diagram of User actor

3.2.4.2.1. Authentication

***Login with Google or Facebook**

USECASE US-UC-1.0 SPECIFICATION			
Use case ID	US-UC-1.0	**Use case version**	V1.0
Use case name	Login with Google+ or Facebook		
Author	Nguyen Đỗ Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	Allows User login to PD system in mobile application.		
Pre-condition	People has a Google/Facebook account and the account is still working.		
Post-condition	**When the normal flow completes successfully, the login status is set to true, and the access token is saved on the application.**		
Trigger			

Main flow:

No.	Actor events	System respond
1.	User opens PD application	
2.		App displays title “Đăng nhập vào PET-DATING”, “Đăng nhập với Google+” button and “Đăng nhập với Facebook” button.
3.	User clicks “Đăng nhập với Google+” or “Đăng nhập với Facebook”.	

4.		App shows a pop-up login.
5.	User fills email and password or choose account (if signed in before)	
6.		**System generates access token, user profile and save it on application.**
7.		**App navigates to inside app.**

Alternative Flows: User logged and not yet log out.

No.	Actor events	System respond
1.	User opens PD application	
2.		**Application checks data then navigates to inside app.**

Extension Flows: N/A

Exception:

US1.0-E1 – Cannot connect with Google API

System displays error message.

US1.0-E2 – Cannot connect with Facebook API

System displays error message.

Business Rules: B1

*Logout

USECASE US-UC-2.0 SPECIFICATION

Use case ID	US-UC-2.0	**Use case version**	V1.0
Use case name	LogOut		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	Log out the app		
Pre-condition	Has logged into on application		
Post-condition	When the normal flow completes successfully, the login status is set to false, and the access token is cleared.		
Trigger			

Main flow:

No.	Actor events	System respond
1.	Clicks icon “Account” on top right of app	
2.		Application displays “Account “ screen
3.	User clicks “Logout” on tab	
4.		Website navigates to Login screen

Alternative Flows: N/A

Extension Flows:

Exception: N/A

Business Rules: N/A

3.2.4.2.2. Match

USECASE US-UC-3.0 SPECIFICATION			
Use case ID	US-UC-3.0	**Use case version**	V1.0
Use case name	Match		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User selects pet for dating.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User choose Matching screen.	
2.		Application displays matching Screen
3.	User choose pet active on list pets	
4.		Screen display pet was choosed
5	User choose “X” button. Or drag pet card left to skip	
6.	User choose “V” button. Or drag pet card right to match	
7.		Application next to other pet display list of pets
8.	User Vip can use “Rollback “ to back pet was passed	

9.		If user match with other pet , screen display message “Matched !“

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US10.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.3. Chat

3.2.4.2.3.1. Send message

USECASE US-UC-4.0 SPECIFICATION			
Use case ID	US-UC-4.0	**Use case version**	V1.0
Use case name	Send message		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User sends messages to others user.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	Do US-UC-2.0 then view receiver profile.	

2.		Application displays receiver profile.
3.	User clicks “Chat” button in receiver profile.	
4.		Application displays “Chat” screen.
5.	User enters message then clicks icon send.	
		Application sends message to receiver.

Alternative Flows:

No.	Actor events	System respond
1.	User click message icon on top tab bar	
2.		Application displays “ConversationList” screen.
3.	User clicks specific conversation	
4.		Application displays “Chat” screen.
5.	User fills receiver name and enter message then clicks icon send.	
		Application sends message to receiver.

Extension Flows: N/A

Exception:

US5.0-E1 – Cannot communicate with API server

System displays error message.

US5.0-E2 – User does not exist or empty

System cannot send message.

Business Rules:N/A

3.2.4.2.3.2. Receive message

USECASE US-UC-5.0 SPECIFICATION			
Use case ID	US-UC-5.0	**Use case version**	V1.0
Use case name	Receive message		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User receives messages from others user.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.		Application display notification
2.	User click message icon on top tab bar	
3.		Application displays “ConversationList” screen.
4.	User clicks sender name to read message.	
5.		Application displays chatting screen with sender.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.4. Ranking

USECASE US-UC6.0 SPECIFICATION			
Use case ID	US-UC-6.0	**Use case version**	V1.0
Use case name	Ranking		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User see top of 10 pet have most reaction		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User choose sidebar	
2.		Application display sidebar with menu (feedback , ranking, privacy ,..)
3.	User choose ranking	

4.		Application display top of pet, that have most of like

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US6.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.5. Feedback

USECASE US-UC-7.0 SPECIFICATION			
Use case ID	US-UC-7.0	**Use case version**	V1.0
Use case name	Feedback		
Author	Nguyen Do Anh Khuong		
Date	09/07/2020	**Priority**	Medium
Primary actor	User	**Secondary actor**	N/A
Description	User can feed back about bug, issue … to admin		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User choose sidebar	

2.		Application display sidebar with menu (feedback , ranking, privacy ,..)
3.	User choose “Feedback”	
4.		Application display Text area to feedback
5.	User Enter his feedback on text area And choose “submit”	
6.		Show message “ your feedback was send to admin, thank you for your feedback !”

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US4.0-E1 – Cannot communicate with API server

System displays error message.

US4.0-E2 – User does not exist and not match

System displays nothing.

Business Rules: N/A

3.2.4.2.6. React pet

USECASE US-UC-8.0 SPECIFICATION			
Use case ID	US-UC-8.0	**Use case version**	V1.0
Use case name	React pet		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A

Description	User react a pet to rank pet on system
Pre-condition	User has logged into application.
Post-condition	Privacy of user is public
Trigger	

Main flow:

No.	Actor events	System respond
1.	User choose react on UC-3.0	
2.		Application displays target pet's profile.
3.	User clicks react icon in target pet profile.	
4.		Application displays success message.

Alternative Flows: N/A

Extension Flows:

No.	Actor events	System respond
1.	Click on Pet profile	
2.		Application displays target profile.
3.	User clicks like icon in target profile.	
4.		Application displays success message.

Exception:

US8.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.7. View other user

USECASE US-UC-9.0 SPECIFICATION			
Use case ID	US-UC-9.0	**Use case version**	V1.0
Use case name	View other user		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	Medium
Primary actor	User	**Secondary actor**	N/A
Description	User views others user's profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User use "Match" screen and click "info" on top-right card and choose "View owner"	
2.		Application displays profile screen, which includes all information of that user : Name Date of birth Gender Email Phone Description

3.	User roll down to choose user's profile, which user want to view	

Alternative Flows: N/A

No.	Actor events	System respond
1.	User clicks top left icon on top tab bar	
2.		Application displays profile screen, which includes: Name Date of birth Gender Email Phone Description
3.	User roll down to choose user's profile, which user want to view	

Extension Flows: N/A

Exception:

US7.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.8. Hide profile

USECASE US-UC-11.0 SPECIFICATION

Use case ID	US-UC-11.0	Use case version	V1.0
Use case name	Hide profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	Medium
Primary actor	User	**Secondary actor**	N/A
Description	User hide profile (other user can't see his/her profile and all of her/his pets)		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User choose sidebar	
2.		Application display sidebar with menu (feedback , ranking, privacy ,..)
3.	User choose "Hide profile"	
4.		Show confirm message "You are sure to hide your profile "
5.	User choose "Yes"	
		Show message "your profile was hide !"

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US7.0-E1 – Cannot communicate with API server

System displays error message. **Business Rules: N/A**

3.2.4.2.9. Manage profile

3.2.4.2.9.1. Modify user's profile

USECASE US-UC-12.0 SPECIFICATION			
Use case ID	US-UC-12.0	**Use case version**	V1.0
Use case name	Modify user's profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User modify own profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User clicks "Account" icon on top tab bar	
2.		Application displays "Account Management" screen
3.		Application navigates to "Own Profile" screen, which include: avata profile list image

4.	User clicks “Modify profile”.	
5.		Application navigates to “Modify Profile” screen.
6.	User fills and select all required information then clicks “Edit” button.	
		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US13.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: B5, B8, B9

3.2.4.2.9.2. Delete user profile

USECASE US-UC-13.0 SPECIFICATION			
Use case ID	US-UC-13.0	**Use case version**	V1.0
Use case name	Delete user’s profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User delete his/her profile		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User clicks “Account” icon on side tab bar	
2.		Application displays “Account Management” screen
3.	Choose “Edit profile”	
4.		Application navigates to “Profile” screen, which include: avata profile list image
5.	User clicks “Delete profile”.	
6.		Show confirm message “ Do you want to delete your profile . Warn : your profile will loss all of information
7.	User choose “yes”.	
8.		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US13.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.10. Find user

USECASE US-UC-17.0 SPECIFICATION			
Use case ID	US-UC-17.0	**Use case version**	V1.0
Use case name	Find		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User Find other user/pets around		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	Choose “Find” icon on sidebar	
2.		Application navigates to “Find” screen
3.	User setting distance and gender of pet/user .	
4.		Show list pet/user appropriate with condition
5.	Choose pet/user you want to see more information	
6.		Show detail information of that pet/user

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US17.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.11. React Pet

USECASE US-UC-8.0 SPECIFICATION			
Use case ID	US-UC-8.0	**Use case version**	V1.0
Use case name	React pet		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User react a pet to rank pet on system		
Pre-condition	User has logged into application.		
Post-condition	Privacy of user is public		
Trigger			

Main flow:

No.	Actor events	System respond
1.	User choose react on UC-3.0	
2.		Application displays target pet's profile.
3.	User clicks react icon in target pet profile.	
4.		Application displays success message.

Alternative Flows: N/A

Extension Flows:

No.	Actor events	System respond
1.	Click on post	
2.		Application displays target profile.
3.	User clicks follow icon in target profile.	
4.		Application displays success message.

Exception:

US8.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.12. View another user

USECASE US-UC-9.0 SPECIFICATION			
Use case ID	US-UC-9.0	**Use case version**	V1.0
Use case name	View other user		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	Medium
Primary actor	User	**Secondary actor**	N/A
Description	User views others user's profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User clicks top left icon on top tab bar	
2.		Application displays profile screen, which includes: List user
3.	User roll down to choose user's profile, which user want to view	

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US7.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.13. View other user's pets

USECASE US-UC-10.0 SPECIFICATION			
Use case ID	US-UC-10.0	**Use case version**	V1.0
Use case name	View other user		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	Medium

Primary actor	User	**Secondary actor**	N/A
Description	User views others user's profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	When user view other user's profile (US-UC-9.0) him can choose view pet's profile	
2.		Application displays profile screen, which includes: List pet
3.	User roll down to choose pet's profile, which user want to view	

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US7.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.14. Manage profile

USECASE US-UC-11.0 SPECIFICATION			
Use case ID	US-UC-11.0	**Use case version**	V1.0
Use case name	Modify user's profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User modify own profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User clicks "Account" icon on top tab bar	
2.		Application displays "Account Management" screen
3.		Application navigates to "Own Profile" screen, which include: avatar profile list image
4.	User clicks "Modify profile".	
5.		Application navigates to "Modify Profile" screen.
6.	User fills and select all required information then clicks "Edit" button.	
		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US13.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: B5, B6, B8, B9

3.2.4.2.15. Manage pet's profile

3.2.4.2.15.1. Create pet's profile

USECASE US-UC-14.0 SPECIFICATION			
Use case ID	US-UC-14.0	**Use case version**	V1.0
Use case name	Create pet's profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User adds own pet's profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User clicks pet icon on top tab bar	
2.		Application displays "Pet Management" screen
3.	User click new icon.	

4.		Application navigates to “Create Pet” screen.
5.	User fills and select all required information then clicks “Add” button.	
6.		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US15.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: B5, B10

3.2.4.2.15.2. View pet’s profile

USECASE US-UC-10.0 SPECIFICATION			
Use case ID	US-UC-10.0	**Use case version**	V1.0
Use case name	View pet’s profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User views other user’s pet profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User use “Match” screen and click “info” on top-right card	
2.		Application displays profile screen, which includes all information of that pet : Name Date of birth Gender Weight Breed Description
3.	User roll down to choose pet profile, which user want to view	

Alternative Flows: N/A

No.	Actor events	System respond
1.	User clicks pet’s icon on “user profile “ screen	
2.		Application displays profile screen, which includes all information of that pet : Name Date of birth Gender Weight Breed
3.	User roll down to choose pet profile, which user want to view	

Extension Flows: N/A

Exception:

US7.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.15.3. Edit pet's profile

USECASE US-UC-15.0 SPECIFICATION			
Use case ID	US-UC-15.0	**Use case version**	V1.0
Use case name	Edit pet's profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User edit own pet's profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	Do US-UC-16.0 then click "Edit profile"	
2.		Application navigates to "Edit Pet Profile" screen

3.	User fills and select all required information then clicks “Edit” button.	
		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US17.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: B7, B8,B9

3.2.4.2.15.4. Delete pet’s profile

USECASE US-UC-16.0 SPECIFICATION			
Use case ID	US-UC-16.0	**Use case version**	V1.0
Use case name	Delete pet’s profile		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User delete own pet’s profile.		
Pre-condition	User has logged into application.		
Post-condition			
Trigger			

Main flow:

No.	Actor events	System respond
1.	User open “ pet profile “ screen	

2.		Show " Pet profile " screen
3.	Scroll down and choose "Delete Pet"	
4.		Application navigates to "delete Pet Profile" popup confirm
5.	User see information then clicks "yes" button.	
6.		Application displays success message.

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US17.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.2.4.2.16. Get next generation

USECASE US-UC-18.0 SPECIFICATION			
Use case ID	US-UC-18.0	**Use case version**	V1.0
Use case name	Get next generation		
Author	Nguyen Do Anh Khuong		
Date	10/07/2020	**Priority**	High
Primary actor	User	**Secondary actor**	N/A
Description	User get predict for next generation		
Pre-condition	User has logged into application.		
Post-condition			

Trigger	

Main flow:

No.	Actor events	System respond
1.	Open “Match “ screen and click icon “next generation” on slide card	
2.		Application navigates to “view next generation” screen
3.	User see information then clicks “confirm” button.	

Alternative Flows: N/A

Extension Flows: N/A

Exception:

US17.0-E1 – Cannot communicate with API server

System displays error message.

Business Rules: N/A

3.3. Non-functional Requirement

3.3.1. Security

- ✓ User must use Google authentication (login with Google) or Facebook authentication (login with Facebook) in order to join the system. So, this is an absolute guarantee of account information security.
- ✓ Admin does not access to database of user.
- ✓ Your information can hide in private mode.

3.3.2. Availability

- ✓ The system active 24/24.

3.3.3. Usability

- ✓ User interface should be friendly and easy to use.
- ✓ Application supports Android.
- ✓ The guideline is clearly and easy to use.
- ✓ The system is easy to deploy.

CHAPTER 4: SPOFTWARE DESIGN

4.1. Purpose

This chapter is to give the developer team an overview of what the system's architecture is, and how they should be implemented. This chapter consists of:

- ✓ Overview of system architecture
- ✓ Database Design
- ✓ Detailed Design

4.2. Overview of System Architecture

4.2.1. Diagram

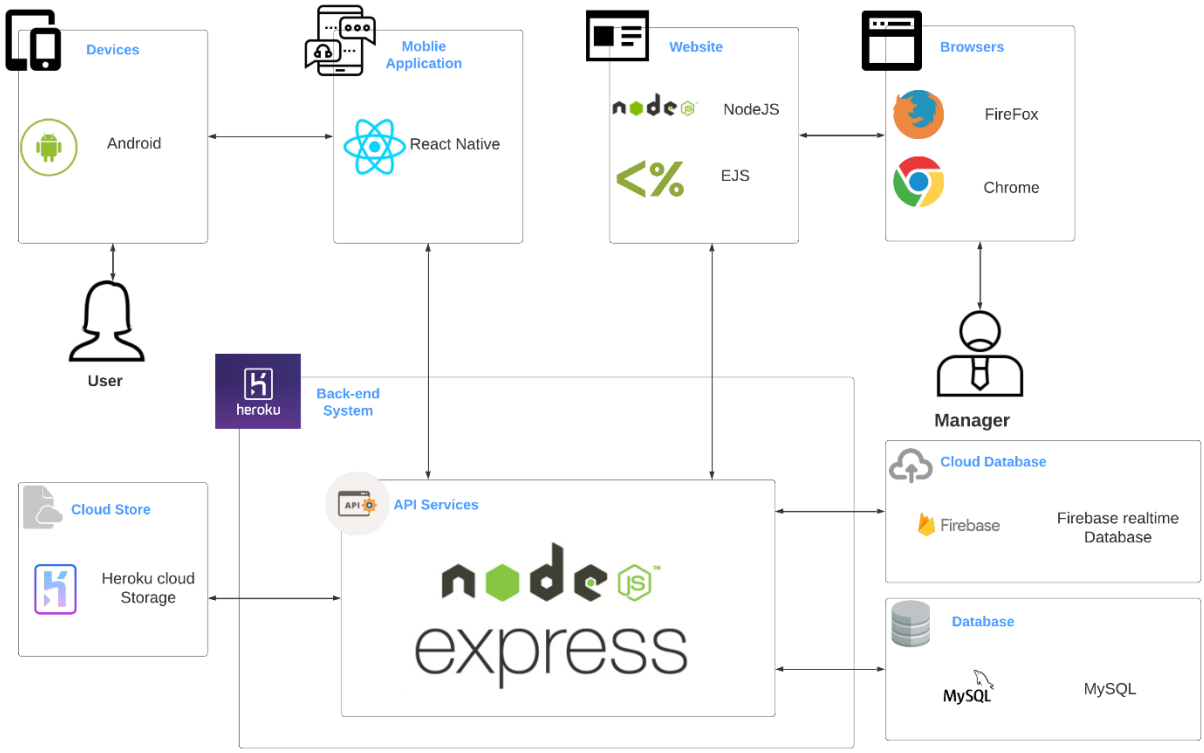


Figure 13 - PetDating system architecture

4.2.2. Component Explanation

This section will explain the function and mechanism of each unit in the system architecture design.

4.2.2.1. Deploy and Hosting

4.2.2.1.1. Heroku

Heroku is a service provider like as hosting, VPS, Cloud, … We use Heroku to deploy API services, hosting for website.

4.2.2.1.2. Heroku Cloud Storage

Heroku Cloud Storage is an online file storage web service for storing and accessing data. The service combines the performance and scalability of Google's cloud with advanced security and sharing capabilities. **We use Heroku Cloud Storage for storing images that uploaded from users.**

4.2.2.2. Mobile Application components

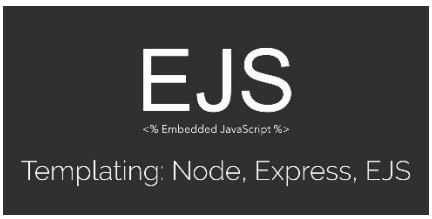
React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces. Many platforms, one React. Create platform-specific versions of components so a single codebase can share code across platforms**.**

4.2.2.3. Website components

4.2.2.3.1. NodeJS

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code server-side. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. **We use NodeJS for font-end website.**

4.2.2.3.2. EJS



EJS is a simple templating language that lets you generate HTML markup with JavaScript. No Religiousness about how to organize things. No reinvention of iteration and control-flow. **We use EJS for operate across devices, operating systems and web browsers.**

4.2.2.4. API Services components

4.2.2.4.1. NodeJS

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code server-side. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. **We use NodeJS to develop API services.**

4.2.2.4.2. ExpressJS

ExpressJS is a NodeJS framework. It's fast, unopinionated, minimalist framework for NodeJS.

We use ExpressJS and many of its modules to construct our API services.

4.2.2.5. Database components

4.2.2.5.1. Firebase Realtime Database

The **Firebase Realtime Database** is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. **We use it to work with chatting function.**

4.2.2.5.2. MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. **We use MongoDB to manage PD data.**

4.3. Application Custom MVC Design

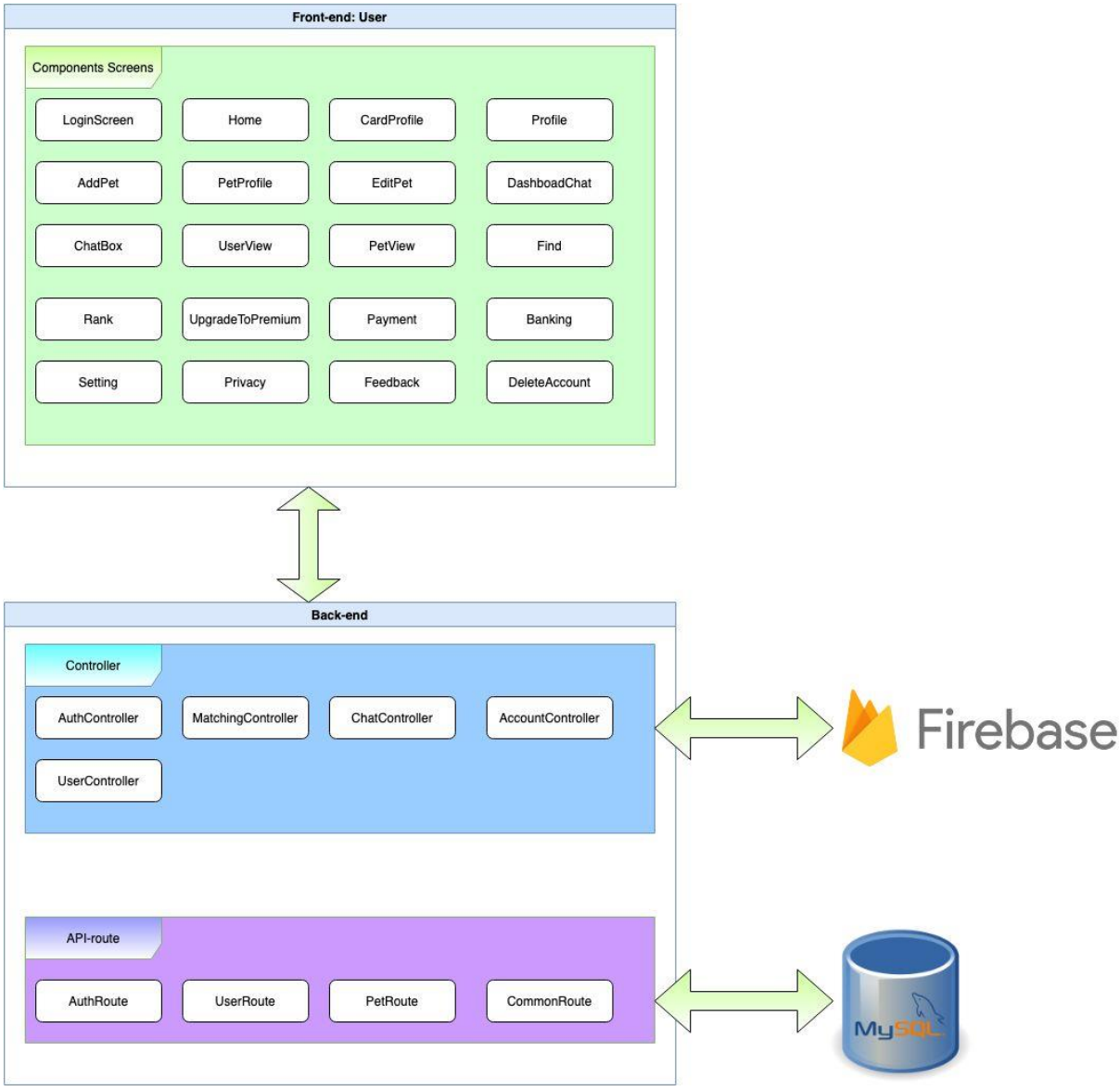


Figure 14 – MVC design

4.4. Database Design

4.4.1. Entity relationship diagram

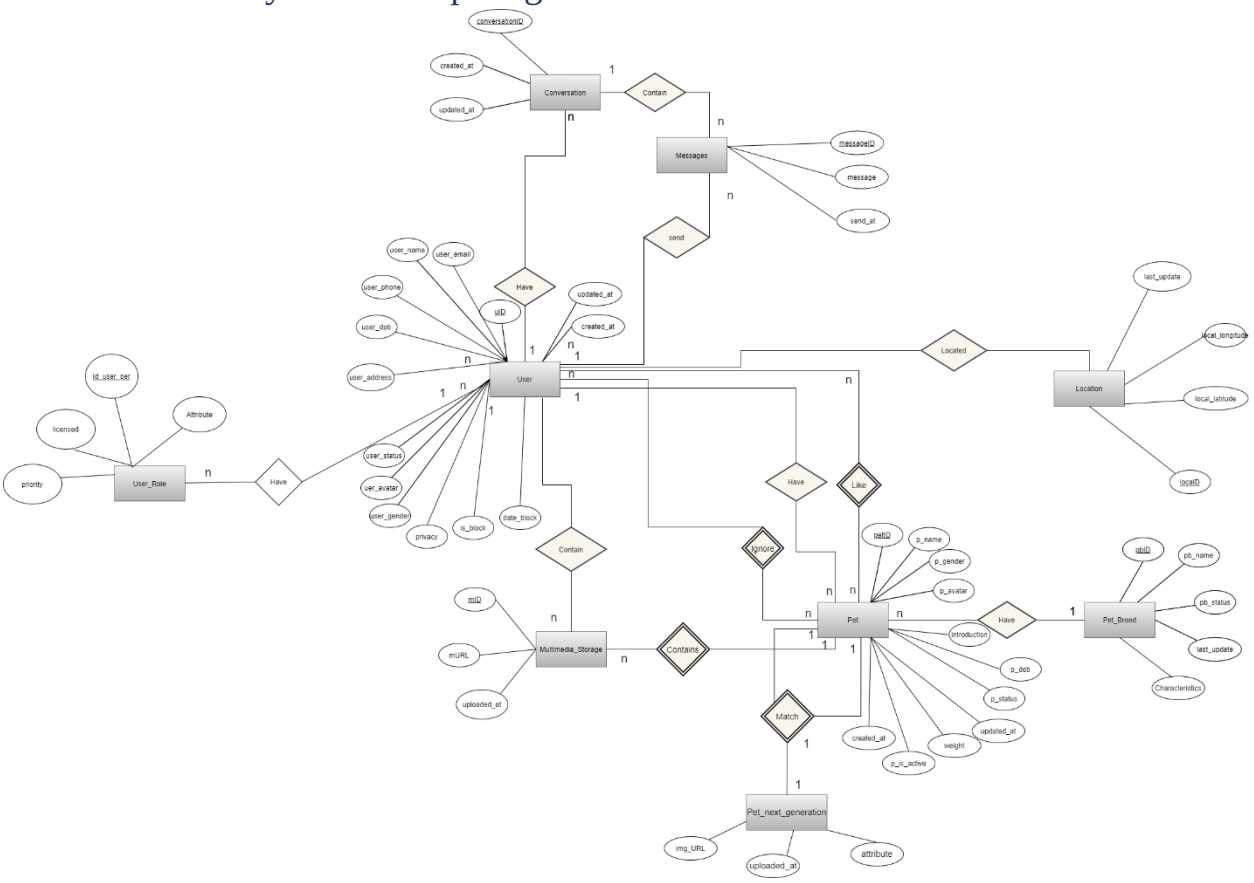


Figure 15 - Entity relationship diagram

Entity	Attributes	Description
User	**uID**	id of user
	user_email	email of user
	user_name	name of user
	user_avatar	avatar of user
	user_gender	gender of user
	user_phone	phone of user
	user_dob	birthdate of user

	user_address	address of user
	user_status	status of user
	privacy	privacy of user
	is_block	user is blocked or not
	date_block	date expire blocked
	local	Real-time location
	created_at	datetime that the user was created
	updated_at	datetime that the user was updated
Pet	**petID**	id of pet
	p_name	name of pet
	p_dob	birthdate of pet
	p_gender	gender of pet
	p_avatar	link to the avatar of pet
	p_status	status of pet
	introduction	introduction of the pet
	p_is_active	this pet is choosen
	weight	weight of pet
	created_at	datetime that the pet was created
	updated_at	datetime that the pet was updated
Location	**loca_id**	id of pet location
	local_latitude	latitude of location
	loca_longitude	longitude of location
	updated_at	datetime that the specie was updated
Multimedia_Storage	**mID**	id of multimedia
	mURL	link to multimedia
	uploaded_at	datetime that the multimedia was uploaded
Conversation	**conversationID**	id of conversation
	user_one	user on conversation
	user_two	user on conversation
	created_at	datetime that the conversation was created

	updated_at	datetime that the last message in conversation was updated
User_Role	**Id_user_per**	id of role
	licensed	licensed of user role
	priority	priority of user role
	attribute	attribute of user role
Pet_Next_Generation	**img_URL**	url of the image
	atribute	attribute of pet next generation
	uploaded_at	datetime that the image was uploaded
Message	messageID	id of message
	message	content of message
	send_at	time message be send

4.4.2. Database diagram

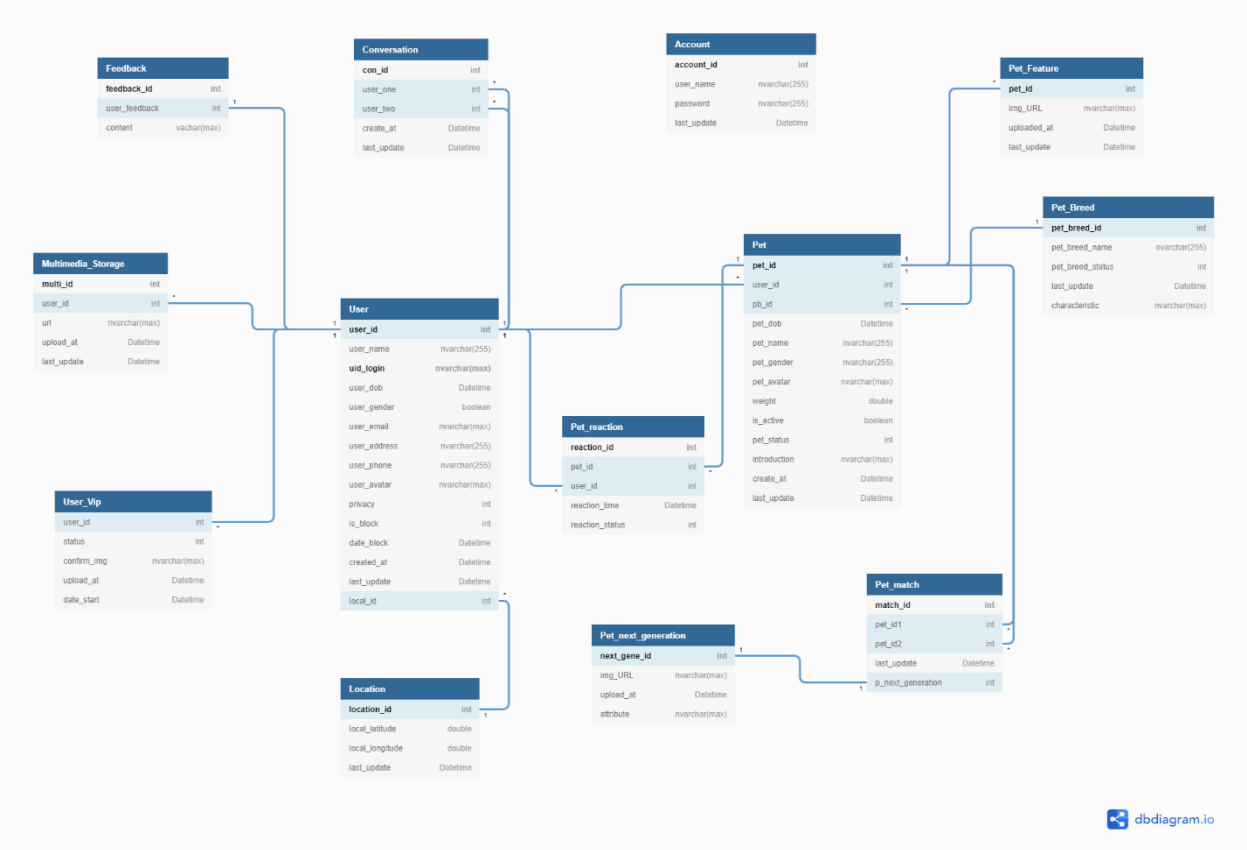


Figure 16 - Database diagram

Database Explanation:

***User:**

No	Field Name	Type	Allow Null	Descerption
1	**uID**	int	no	id of user **PRIMARY KEY**
2	user_email	varchar(255)	No	email of user **UNIQUE**
	uid_login	varchar(max)	No	Id login by google or facebook **PRIMARY KEY**
3	user_name	varchar(255)	No	name of user
4	user_avatar	varchar(max)	No	avatar of user
5	user_gender	boolean	No	gender of user (true : male , false :female)
6	user_phone	varchar(20)	Yes	phone of user
7	user_dob	date	No	birthdate of user
8	user_address	varchar(max)	Yes	address of user
9	user_status	boolean	No	status of user 1: Activated - Default 0: inActivated
10	privacy	boolean	No	privacy of user true: unhide false : hide
11	is_block	int	No	User is blocked or not 1 : block 0 : unblock
	date_block	date	No	Expire block date
12	created_at	timestamp	No	datetime that the user was created Default: CURRENT_TIMESTAMP
13	updated_at	timestamp	No	datetime that the pet was updated

				Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***Pet:**

No	Field Name	Type	Allow Null	Descerption
1	**petID**	int	No	id of pet **Primary key**
2	*user_id*	int	No	id of user who have pets Foreign key references to uID in "User" table
3	*pb_id*	int	No	breed of pet Foreign key references to pbID in "PetBreed" table
5	p_name	varchar(255)	No	name of pet
6	p_dob	date	No	birthdate of pet
7	p_gender	varchar(255)	No	gender of pet (Đực; Cái)
8	p_avatar	varchar(max)	No	avatar of pet
9	weight	double	Yes	weight of pet
10	is_active	boolean	No	pet is choosen 1: Activated 0: inActivated
11	p_status	int	No	status of pet
12	introduction	text	Yes	introduction about pet
13	created_at	timestamp	No	datetime that the pet was created Default: CURRENT_TIMESTAMP
14	updated_id	timestamp	No	datetime that the pet was updated Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***Pet_Breed:**

No	Field Name	Type	Allow Null	Descerption
1	**pbID**	int	No	id of pet's breed **Primary key**
2	*ps_id*	int	No	id of pet's species Foreign key references to psID in "PetSpecies" table
3	pb_name	varchar (30)	No	name of breed
4	pb_status	boolean	No	status of breed 1: Activated - Default 0: inActivated
5	updated_at	timestamp	No	datetime that the breed was updated. Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***Location:**

No	Field Name	Type	Allow Null	Descerption
1	**locaID**	int	No	id of location **Primary key**
2	local_latitude	double	No	local_latitude of location
3	local_longitude	double	No	local_longitude of location
4	updated_at	timestamp	No	time that the location was updated.

***Conversation:**

No	Field Name	Type	Allow Null	Descerption
1	**conID**	int	No	id of conversation

				Primary key
2	*userOne*	int	No	id of user one who join in a conversation Foreign key references to uID in "User" table
3	*userTwo*	int	No	id of user two who join in a conversation Foreign key references to uID in "User" table
4	created_at	timestamp	No	datetime that the conversation was created Default: CURRENT_TIMESTAMP
5	updated_at	timestamp	No	datetime that the last message in conversation was updated Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***Mutimedia_Storage:**

No	Field Name	Type	Allow Null	Descerption
1	**mID**	int	No	id of multimedia **Primary key**
2	*user_id*	int	No	user_id of user who has multimedia Foreign key refernces to uID in "User" table
3	url	varchar(max)	No	url contain content
4	uploaded_at	timestamp	No	datetime that the multimedia was uploaded Default: CURRENT_TIMESTAMP

***Account**

No	Field Name	Type	Allow Null	Descerption
1	**account_id**	int	No	id of account

2	user_name	varchar(255)	No	username of account user
3	password	varchar(255)	No	password of account user
4	last_update	timestamp	No	Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***User_Vip:**

No	Field Name	Type	Allow Null	Descerption
1	**user_id**	int	No	id of user have vip
2	status	int	No	status of user vip 1: is Vip 0: non-Vip
3	confirm_img	varchar(max)	No	image user screen shot transfer money to accounts of admin
4	upload_at	timestamp	No	Default: CURRENT_TIMESTAMP
5	date_start	Datetime	No	date admin set user is Vip

***Feedback:**

No	Field Name	Type	Allow Null	Descerption
1	**feedback_id**	int	No	id of feedback
2	user_feedback	int	No	id of user feedback reference to User table
3	content	vachar(max)	No	content of feedback

***Pet_Feature:**

No	Field Name	Type	Allow Null	Descerption
1	**pet_id**	int	No	id of pet reference to PetId in Pet table

2	img_URL	varchar(max)	No	img url content
3	uploaded_at	timestamp	No	Default: CURRENT_TIMESTAMP
4	last_update	timestamp	No	Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

***Pet_Match:**

No	Field Name	Type	Allow Null	Descerription
1	**match_id**	int	No	id of match between two pets
2	*pet_id1*	int	No	id of pet match reference to PetId in Pet table
3	*pet_id2*	int	No	id of pet is matched reference to PetId in Pet table
4	last_update	timestamp	No	Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
5	p_next_generation	int	No	id of pet next generation reference to next_gene_id of Pet_Next_Generation table

***Pet_Reaction:**

No	Field Name	Type	Allow Null	Descerription
1	**reaction_id**	int	No	id of reaction
2	*pet_id*	int	No	id of pet , that is reacted reference to PetId of Pet table
3	*user_id*	int	No	id of user reacted, reference to UserId of User table
4	reaction_time	timestamp	No	Default: CURRENT_TIMESTAMP

5	reaction_status	int	No	type of react: 1: like 2: dislike

***Pet_Next_Generation:**

No	Field Name	Type	Allow Null	Describeription
1	**next_gene_id**	int	No	Id of next generation
2	*img_URL*	varchar(max)	No	url of image, which is content
3	upload_at	timestamp	No	Default: CURRENT_TIMESTAMP
4	attribute	nvarchar(max)	No	attribute of next generation

4.5. Detailed Design

4.5.1. Login

***Login class diagram**

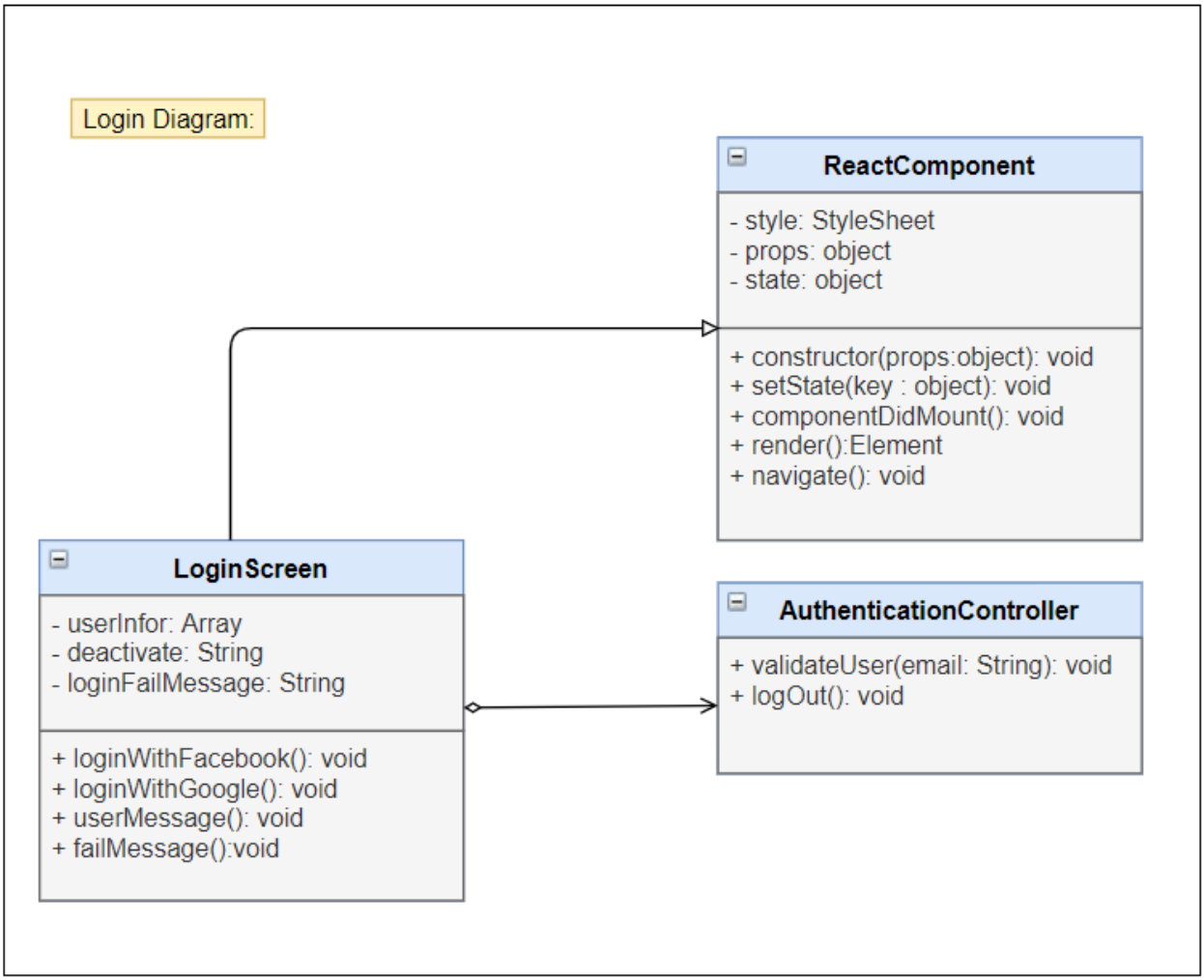


Figure 17 - Login class diagram

Name	AuthController	**Type**	Class
Description	Controller class to control authentication.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
validateUser(email)	void	public	Validate the email in database.

Name	LoginScreen	**Type**	Class
Description	A class that displays login screen.		
Attributes			
Name	**Type**	**Visibility**	**Description**
loading	boolean	private	state of loading modal
Operations			
Name	**Type**	**Visibility**	**Description**
loginWithGoogle()	void	public	Handle login with google
loginWithFacebook()	void	public	Handle login with facebook
saveUserInfo()	void	public	Save user information in AsyncStorage
userMessage()	void	public	Display deactivated message
createNewUser()	void	public	Navigate to CreateUser Screen

***Login with Google sequence diagram**

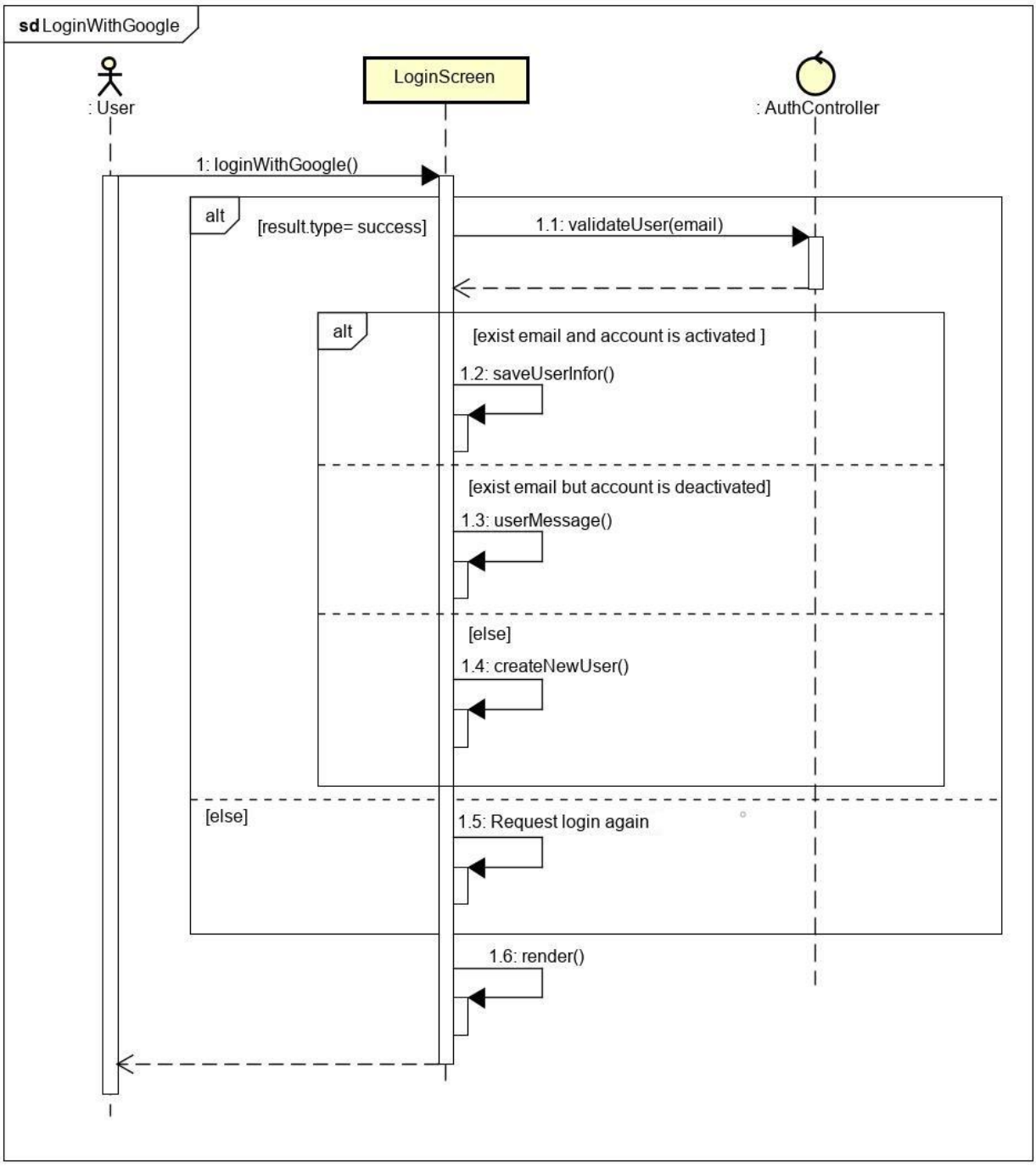


Figure 18 - Login with Google sequence diagram

***Login with Facebook sequence diagram**

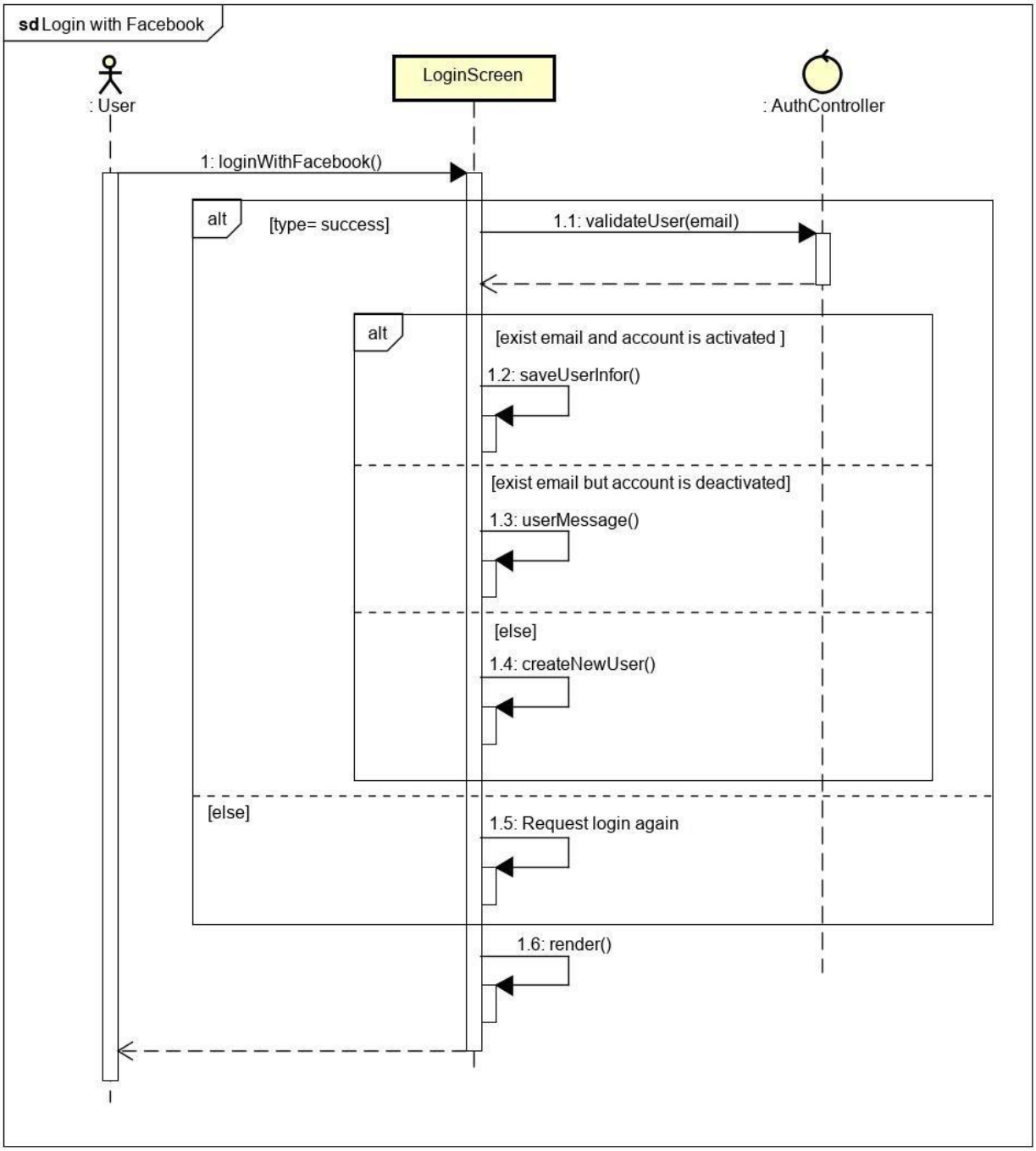


Figure 19 - Login with Facebook class diagram

s4.5.2. Logout

***Logout class diagram**

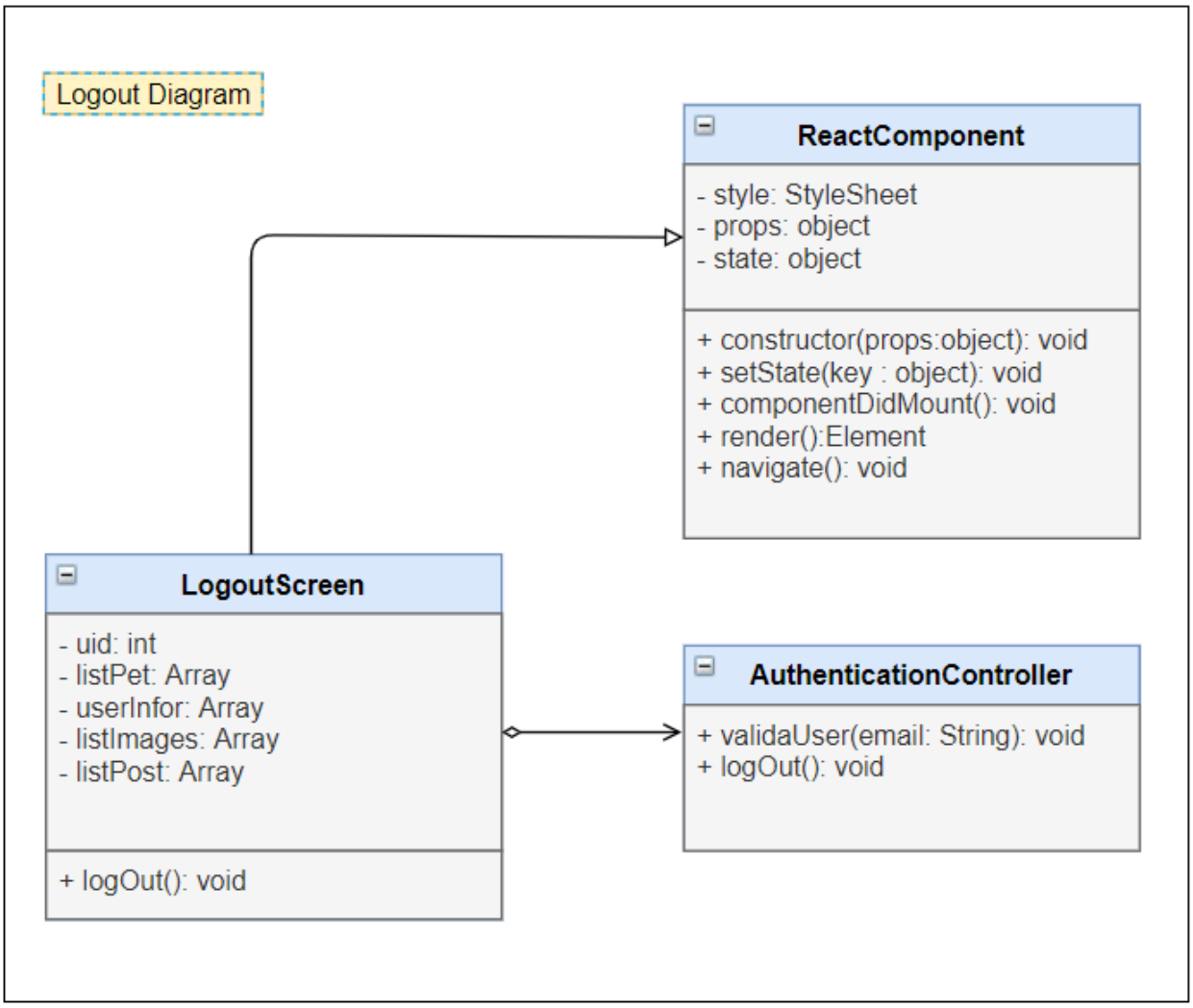


Figure 20 - Logout class diagram

Name	AuthController	**Type**	Class
Description	Controller class to control authentication.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
logOutSystem()	void	public	Clear AnsyncStorage and logout user from application.

Name	NavScreen	Type	Class
Description	A class that displays navigator screen.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
logOut()	void	public	Handle logout from application.

***Logout sequence diagram**

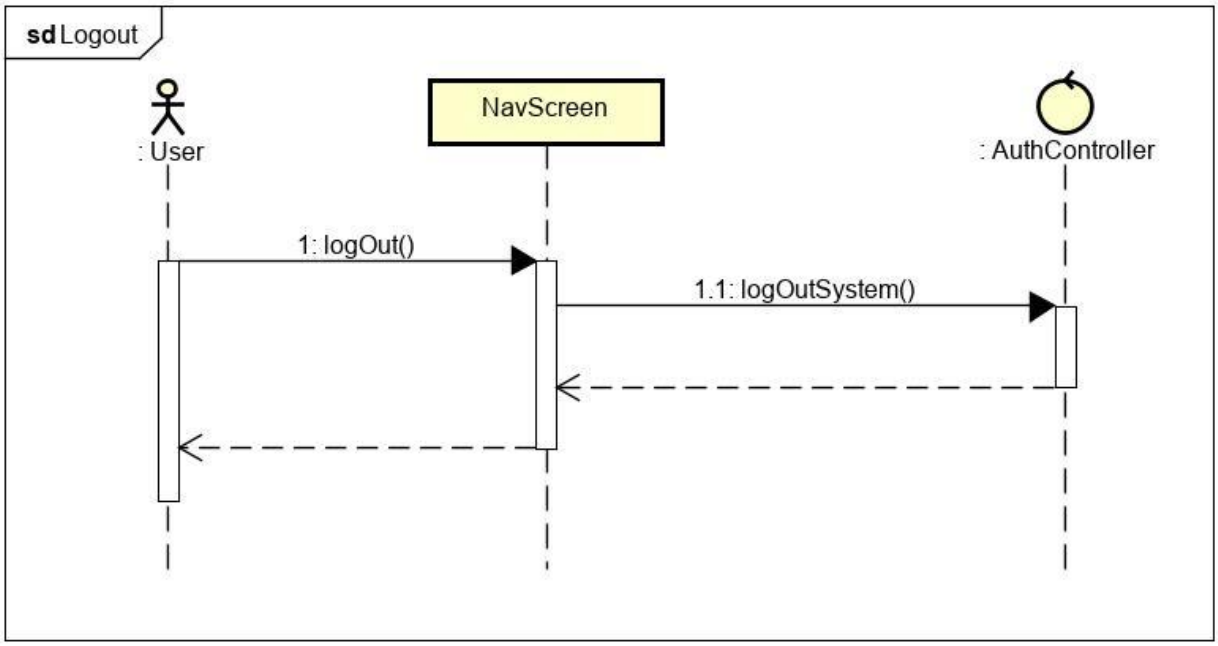


Figure 21 - Logout sequence diagram

4.5.3. Matching

*** Matching Class Diagram**

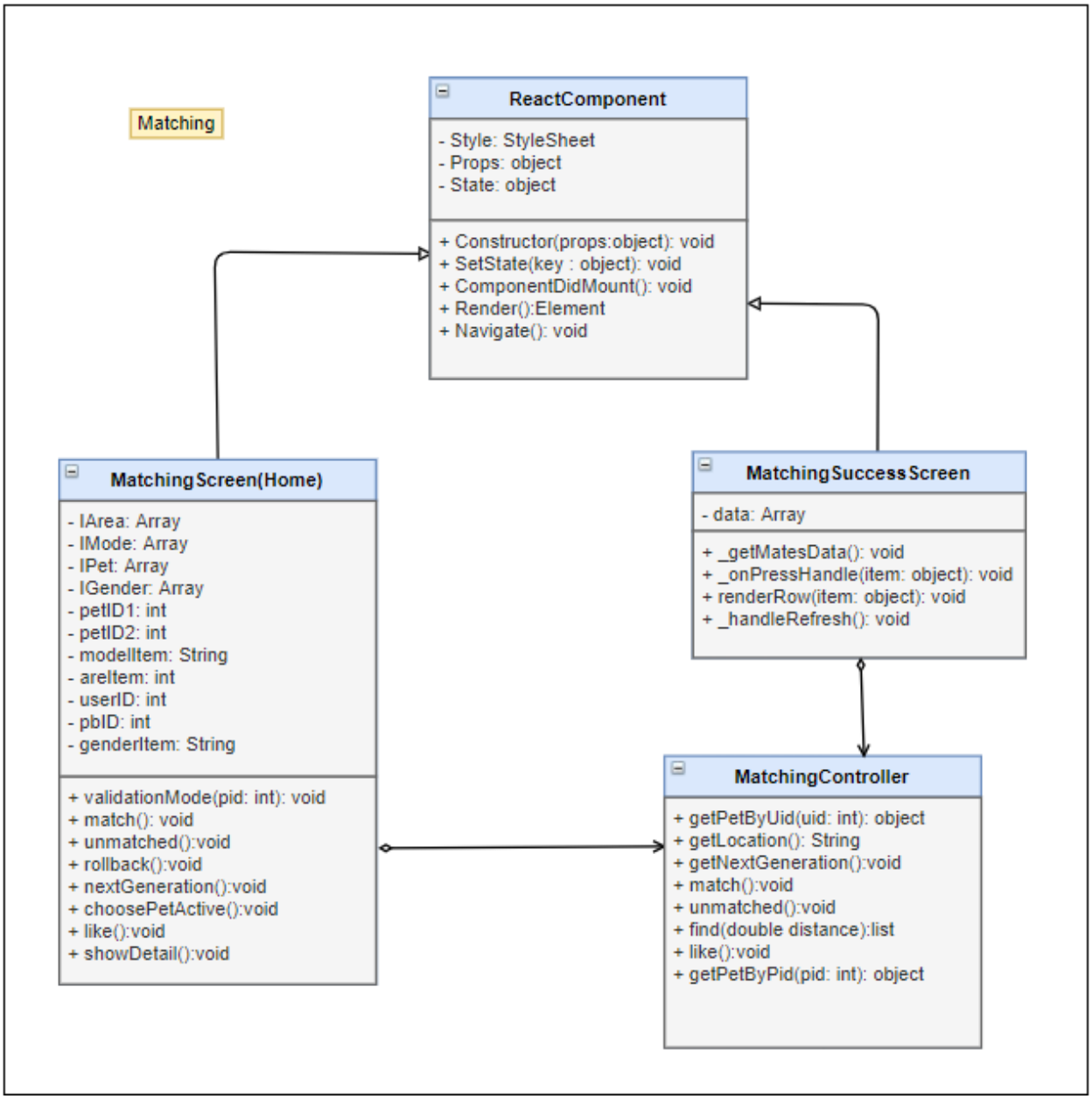


Figure 22 - Matching class diagram

Name	MatchingController	**Type**	Class
Description	Controller class to control User and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**

Operations			
Name	**Type**	**Visibility**	**Description**
getPetByUid (id)	object	public	Get Information of list pet with UserId in database
getLocation ()	void	public	Get real location of user
getNextGeneration()	void	public	Get new baby pet for 2 parent pets
match()	void	public	Match random for active pet Or unmatch them
unmatched()	void	public	Unmatch 2 pets (they are matched)
like()	void	public	Reaction a pet to rankking or show

Name	Matching screen	**Type**	Class
Description	Main (Home) Screen Screen allow user match, like , get nextgeneration , choose pet active		
Attributes			
Name	**Type**	**Visibility**	**Description**
IArea	Array	private	List location of pets
IMode	Array	private	List Mode view of pets
IPet	Array	private	List pet show
IGender	Array	private	List gender of pets
petID1	int	private	Id of pet actived
petID2	int	private	Id of pet matched
modelItem	String	private	Mode of view pets
areItem	int	private	Pet was showed or not
userID	int	private	userId of user , who own pet be matched

pbID	int	private	Pet_breed of pet active
genderItem	String	private	Gender of pet is showing
Operations			
Name	**Type**	**Visibility**	**Description**
validationMode (pid: int)	void	public	Decision pet is showed or not
match	void	public	Add match record between 2 pets to database or delete both of 2 records
unmatched	void	public	Unmatch 2 pet (they are matched)
rollback	void	public	Rollback pet card user passed
nextGeneration	void	public	Get image of Next generation between 2 pets (parent)s
choosePetActive	void	public	Choose pet to match
like	void	public	Reaction pet on screen
showDetail	void	public	Show all information of that pet

Name	MatchingSuccess screen	**Type**	Class
Description	Screen Matching Success Result		
Attributes			
Name	**Type**	**Visibility**	**Description**
data	Array	private	Data tranfer of matching screen
Operations			
Name	**Type**	**Visibility**	**Description**
_onPressHandle	void	public	Handle event press matching button
_getMatesData	void	public	Get list information of 2 pets are matched
renderRow	void	public	Show list pets are matched
_handleRefresh	void	public	Handle event refresh screen

***Matching Sequence Diagram**

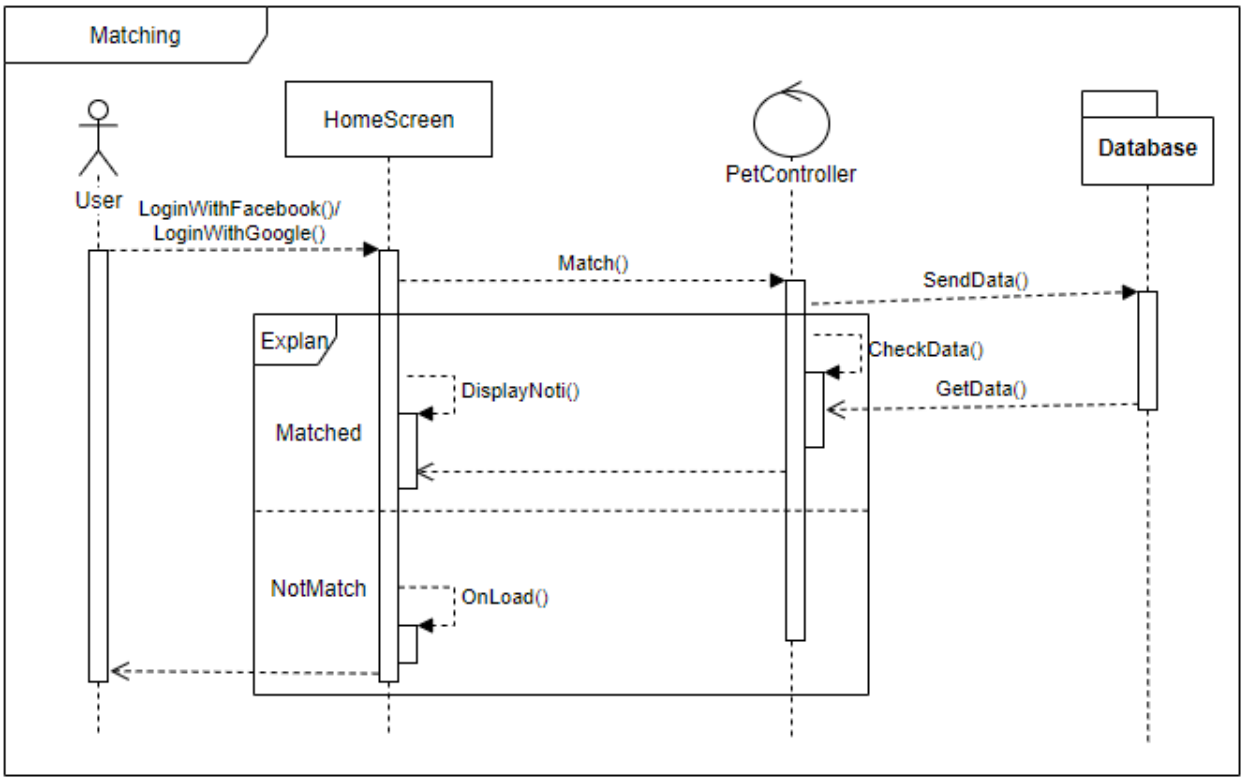


Figure 23 - Matching Sequence diagram

4.5.4. Chat

*** View ConversationList Class Diagram**

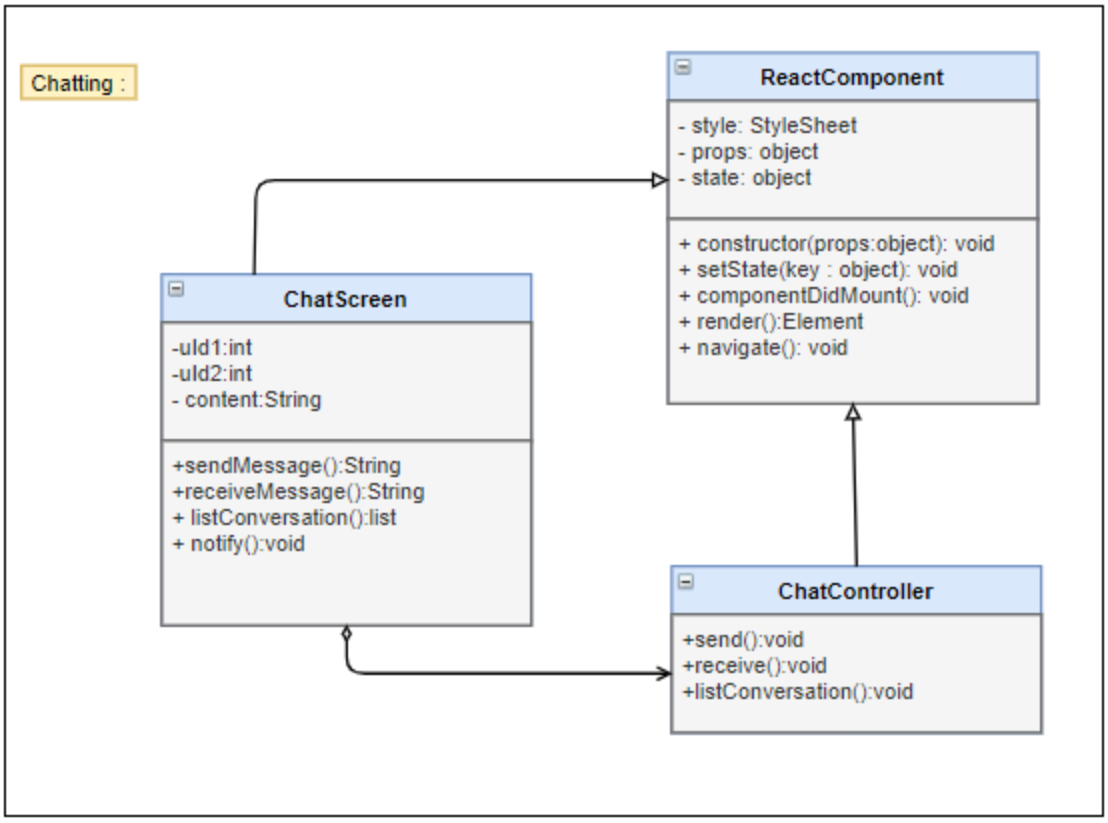


Figure 24 - View ConversationList Class Diagram

Name	ChatController	**Type**	Class
Description	Controller control chat messages and conversations		

Attributes			
Name	**Type**	**Visibility**	**Description**

Operations			
Name	**Type**	**Visibility**	**Description**
send()	void	public	Send message (real time)
receive()	void	public	Recive messages (real time)
listConversation	list	public	Show list all of conversation

Name	Chatting Screen	**Type**	Class

Description	Client Screen show list conversation and chat messages		
Attributes			
Name	**Type**	**Visibility**	**Description**
conversationList	Array	private	List all of conversation
Operations			
Name	**Type**	**Visibility**	**Description**
chatNote	void	public	Load List all of conversation new message
notify	void	public	Notify when receive messages
send	void	public	Send message
receive	void	public	Receive message

Name	ChatController	**Type**	Class
Description	The screen controls chatting action		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getCVList(uid)	Promise	public	Get list of conversations in database

Name	ConversationList	**Type**	Class
Description	The screen displays content of conversation		
Attributes			
Name	**Type**	**Visibility**	**Description**
loggedUserID	int	private	Id of user
dataSource	Array	private	List of conversations
isLoading	boolean	private	Status of loading

Operations			
Name	**Type**	**Visibility**	**Description**
_getConversationList()	void	public	Handle get list of conversations

***View ConversationList Sequence Diagram**

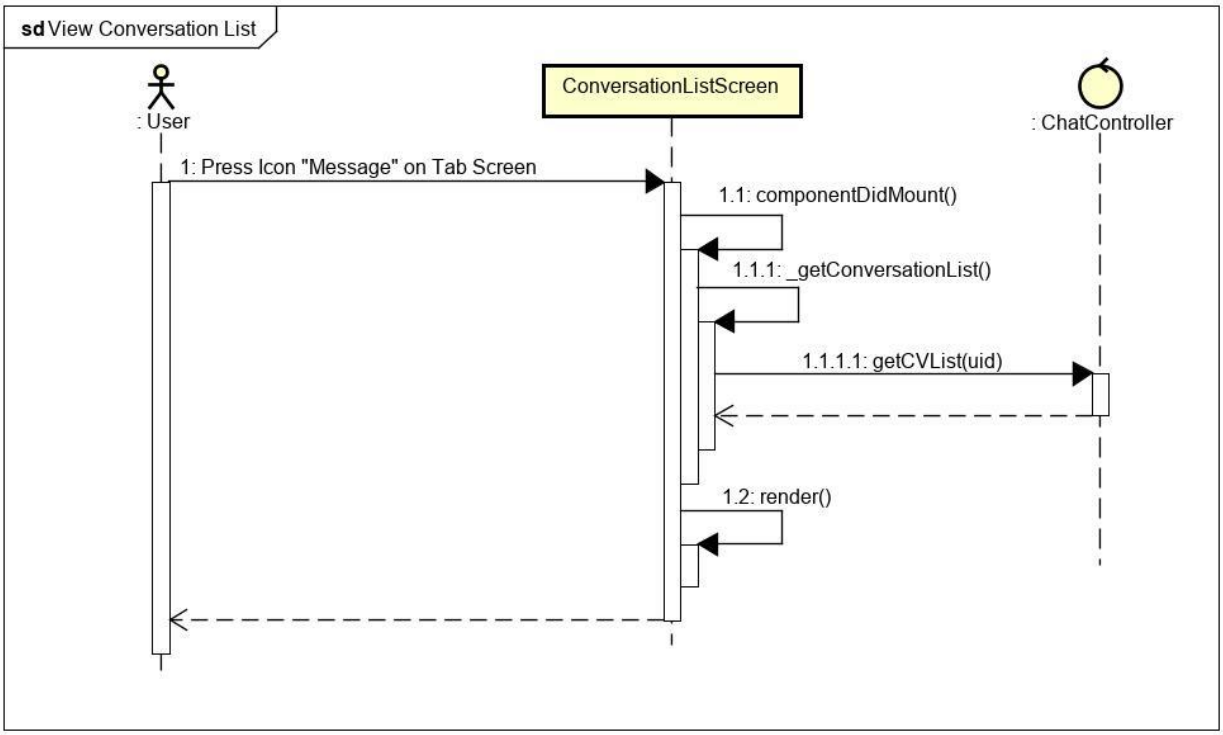


Figure 25 - Conversation list sequence diagram

***Send Messages class diagram**

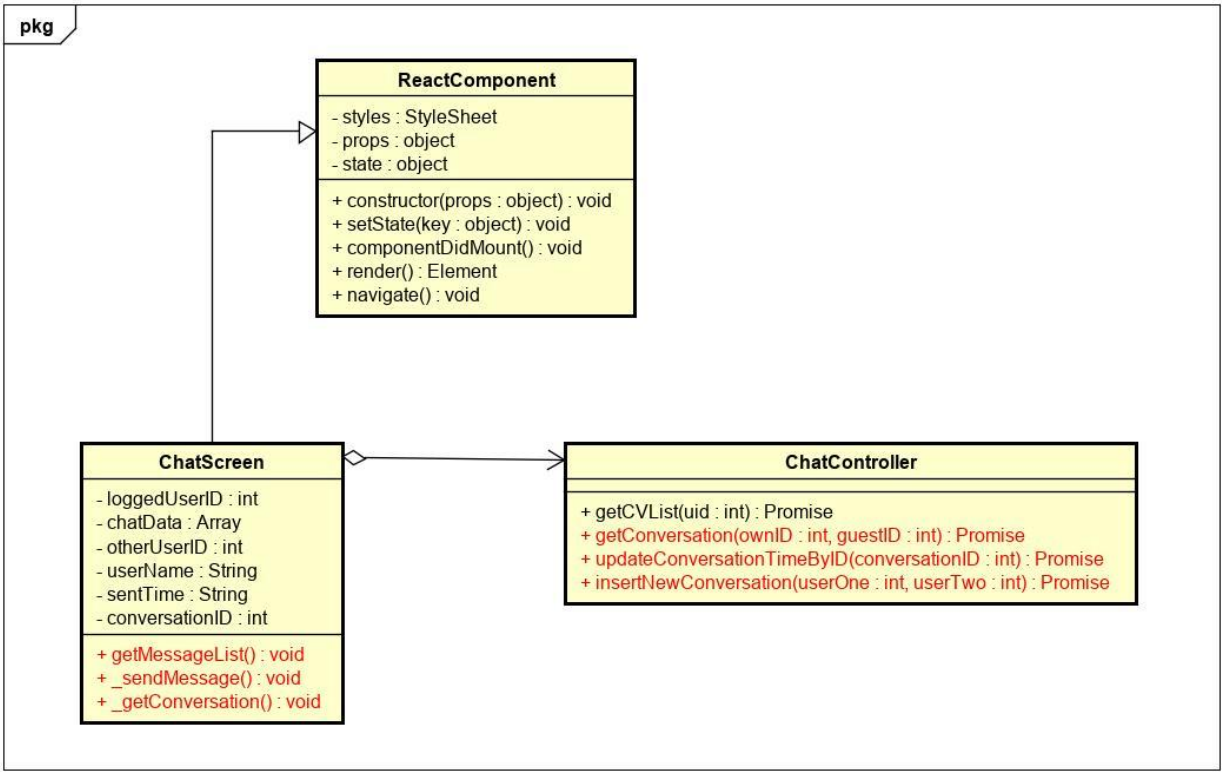


Figure 26 - Send message class diagram

Name	ChatController	**Type**	Class
Description	The screen controls chatting action		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getConversation()	Promise	public	Get detail of the conversations in database
updateConversationTimeByID()	Promise	public	Update conversation time
insertNewConversation()	Promise	public	Insert new conversation to database

Name	ChatScreen	**Type**	Class

Description	The screen displays content of chat		
Attributes			
Name	**Type**	**Visibility**	**Description**
loggedUserID	int	private	Id of own user
chatData	Array	private	List messages
otherUserID	int	private	Status of loading
userName	String	private	Name of user
sentTime	String	private	Current time to send message
conversationID	int	private	Id of this conversation
Operations			
Name	**Type**	**Visibility**	**Description**
_getConversation()	void	public	Handle get detail of conversations
_sendMessage()	void	public	Handle send message
getMessageList()	void	public	Get list of messages

***Send Messages sequence diagram**

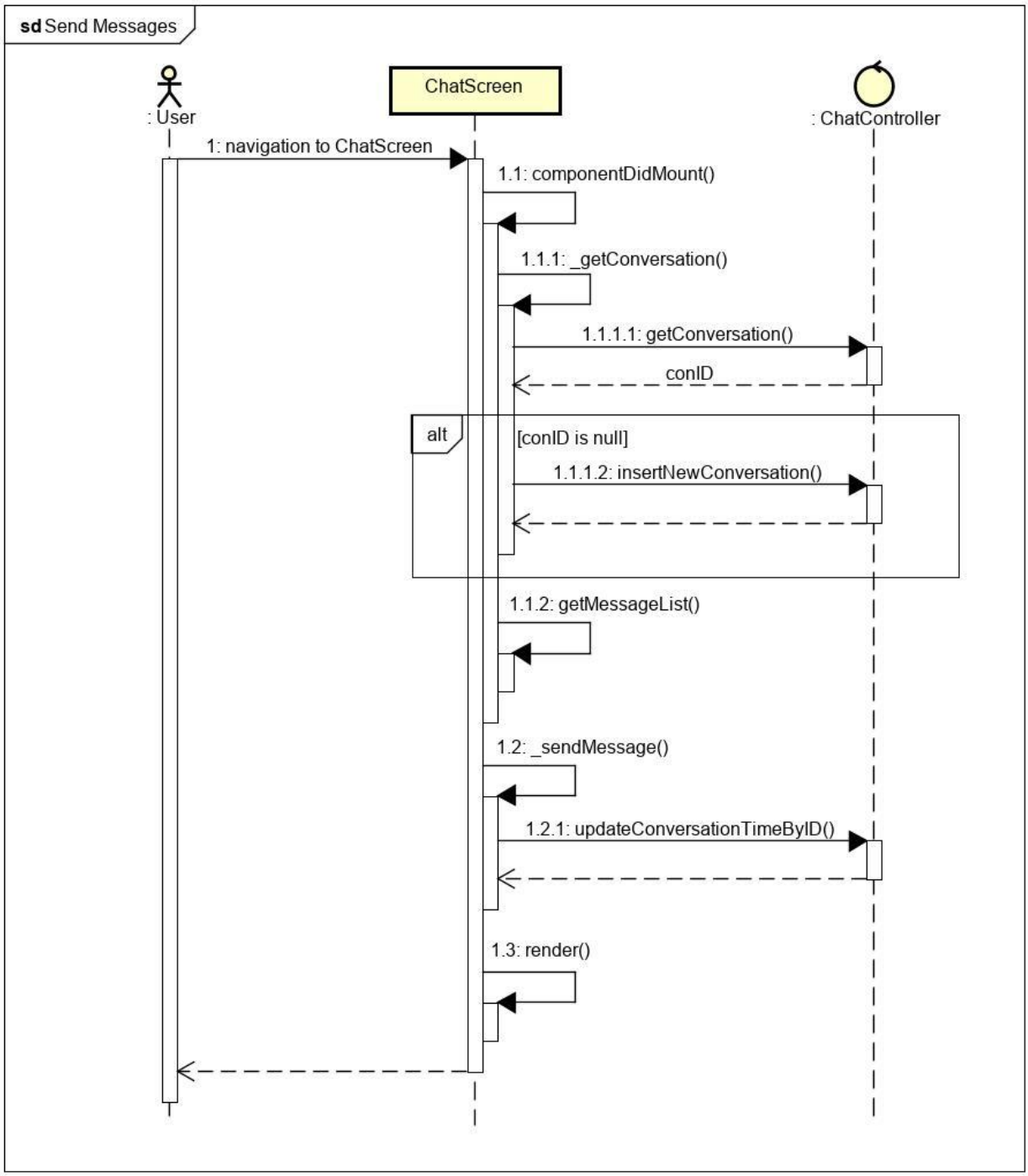


Figure 27 - Send message sequence diagram

***Receive Messages class diagram**

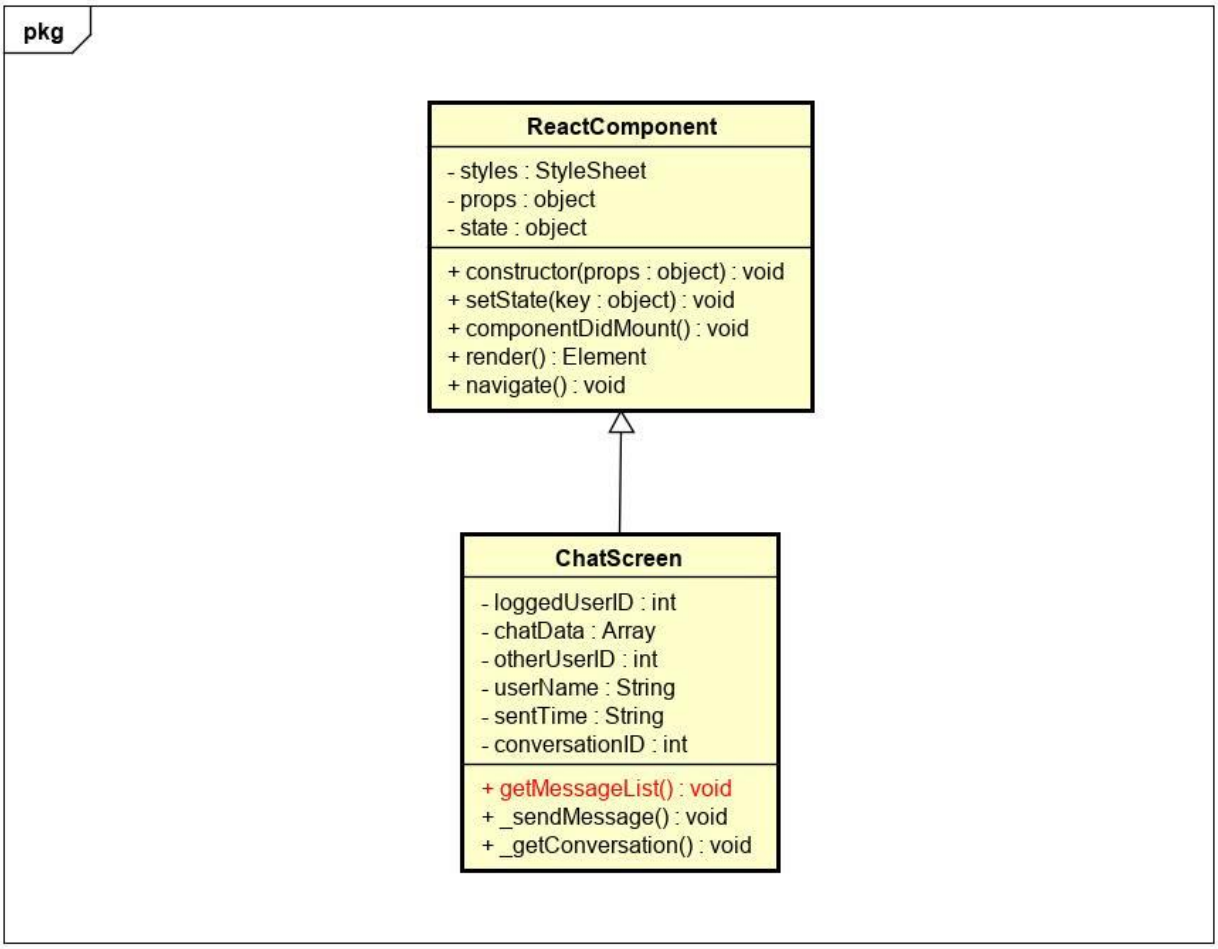


Figure 28 - Receive message class diagram

Name	ChatScreen	**Type**	Class
Description	The screen displays content of chat		
Attributes			
Name	**Type**	**Visibility**	**Description**
loggedUserID	int	private	Id of own user
chatData	Array	private	List messages
otherUserID	int	private	Status of loading
userName	String	private	Name of user
sentTime	String	private	Current time to send message
conversationID	int	private	Id of this conversation
Operations			

Name	Type	Visibility	Description
getMessageList()	void	public	Get list of messages

*** Receive Messages sequence diagram**

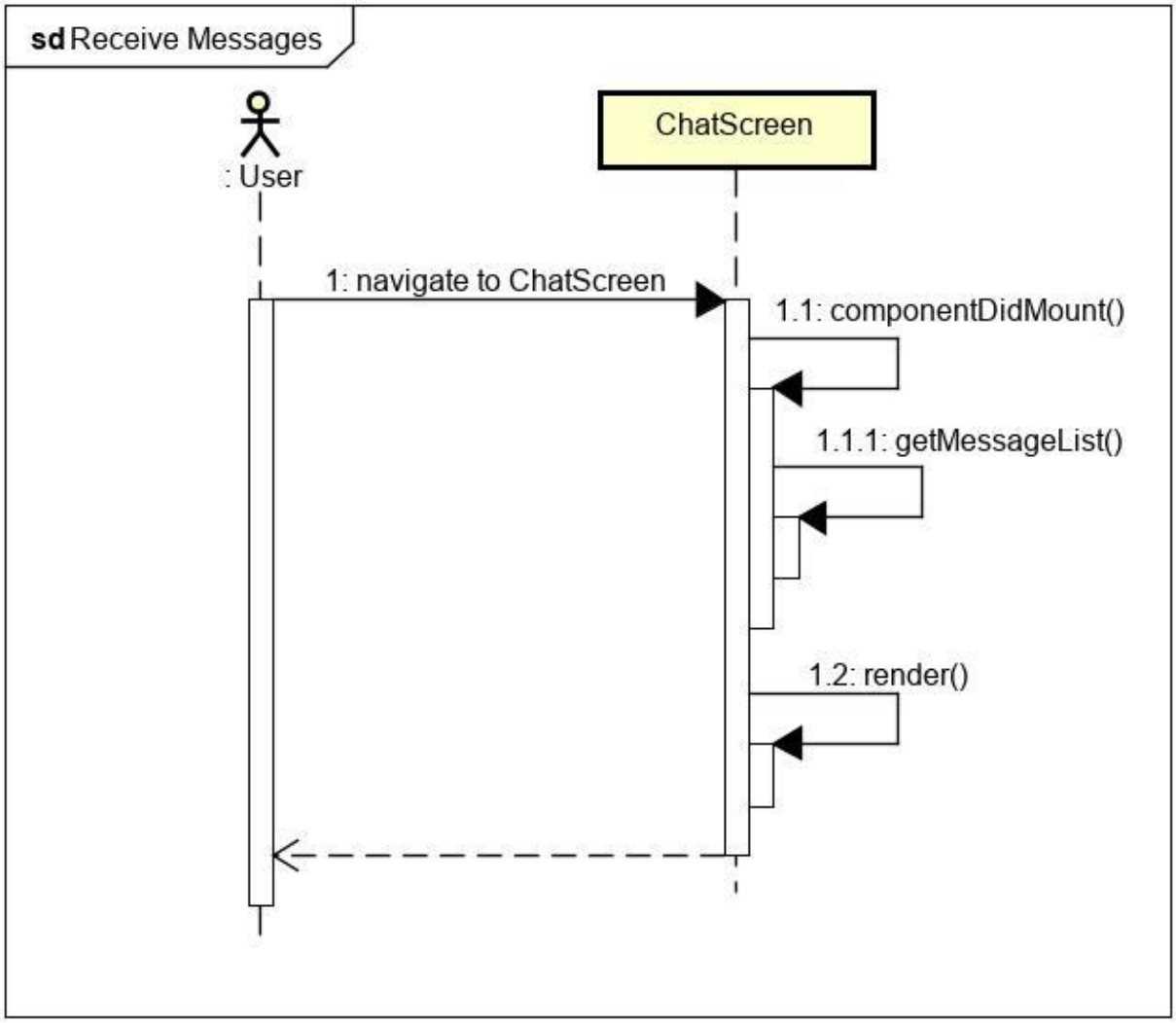


Figure 29 - Receive message sequence diagram

4.5.5. Ranking

*** Ranking class diagram**

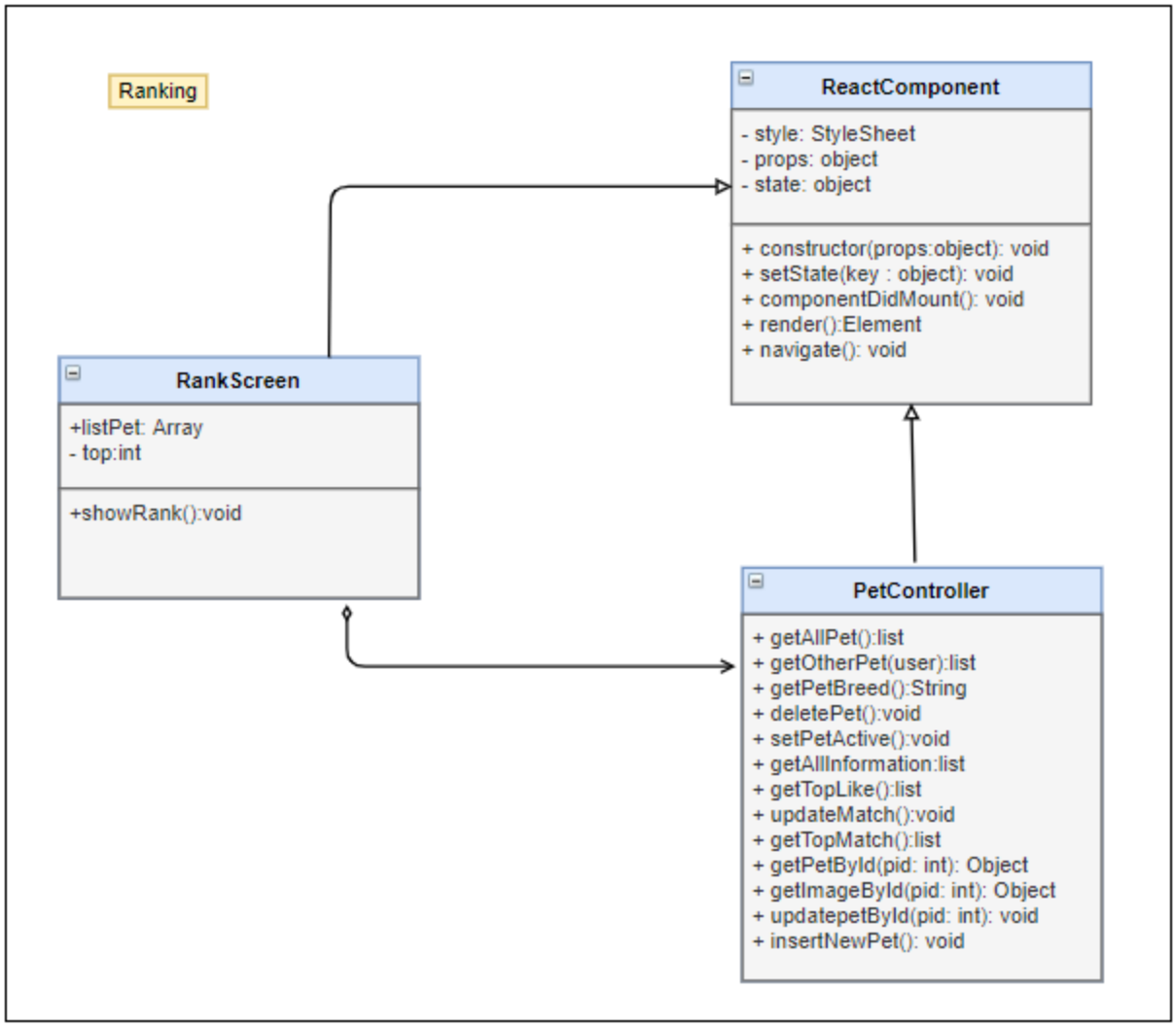


Figure 30 – Ranking class diagram

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getAllPet	list	public	Get list all of pet belong to this user

deletePet	void	public	Delete a pet in list pets
setPetActive	void	public	Set a pet is active to match
getAllInformation	list	public	Get all information of pet on list pet
getTopLike	list	public	Get top pet have most liked
updateMatch	void	public	Update match record in database
getTopMatch	list	public	Get top pets are most matched
getPetById(pid: int)	Object	public	Get Pet by PetID
getImageById(pid: int)	Object	public	Get all image by petID

Name	RankingScreen	**Type**	Class
Description	Screen show top of pet with most liked , match		
Attributes			
Name	**Type**	**Visibility**	**Description**
listPet	Array	private	List pet have most match , like
top	Int	private	Amount pet in rank
Operations			
Name	**Type**	**Visibility**	**Description**
showRank	void	public	Show list pet with most matched , like

*** Ranking sequence diagram**

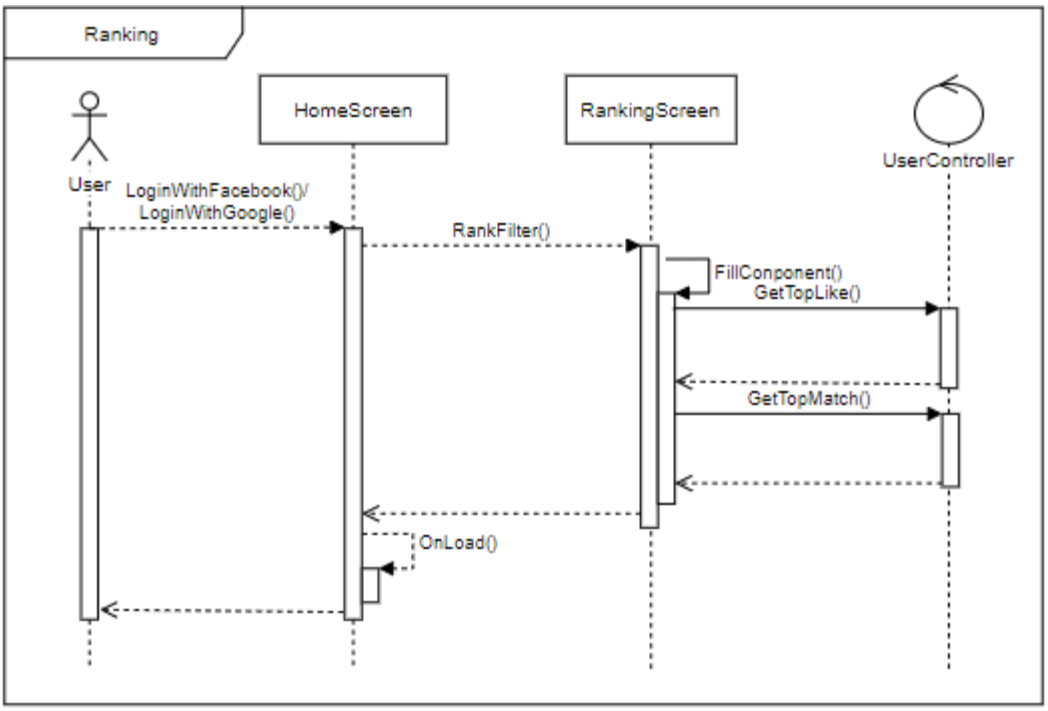


Figure 31 – Ranking sequence diagram

4.5.6. Feedback

*** Feedback class diagram**

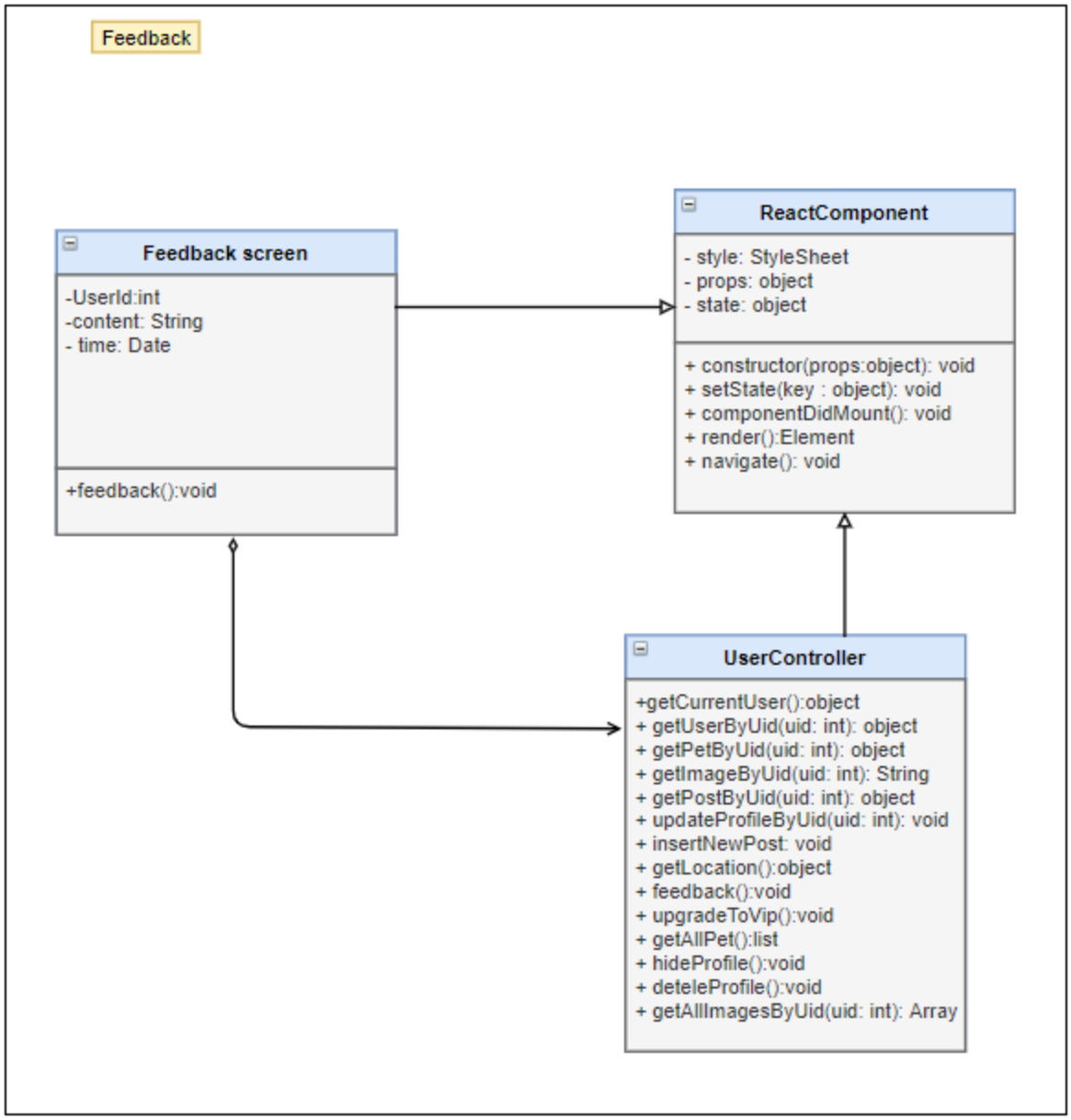


Figure 32 – Feedback class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**

Operations			
Name	**Type**	**Visibility**	**Description**
getCurrentUser()	object	public	Get all information of user, which did login
getUserByUid(uID)	object	public	Get all information of user with Uid
getImageByUid(())	String	public	Select image form database by Uid
getLocation()	object	public	Get location(latitude and longitude) of user
Feedback()	void	public	Send feedback to admin
getAllImagesByUid()	Array	public	Get list of image User has

Name	FeedbackScreen	**Type**	Class
Description	Screen to send feedback		

Attributes			
Name	**Type**	**Visibility**	**Description**
UserId	int	private	Id of currence user have logined
content	String	private	Content of message feedback
time	Date	private	Time feedback

Operations			
Name	**Type**	**Visibility**	**Description**
feedback	void	public	Send feedback to server

*** Feedback sequence diagram**

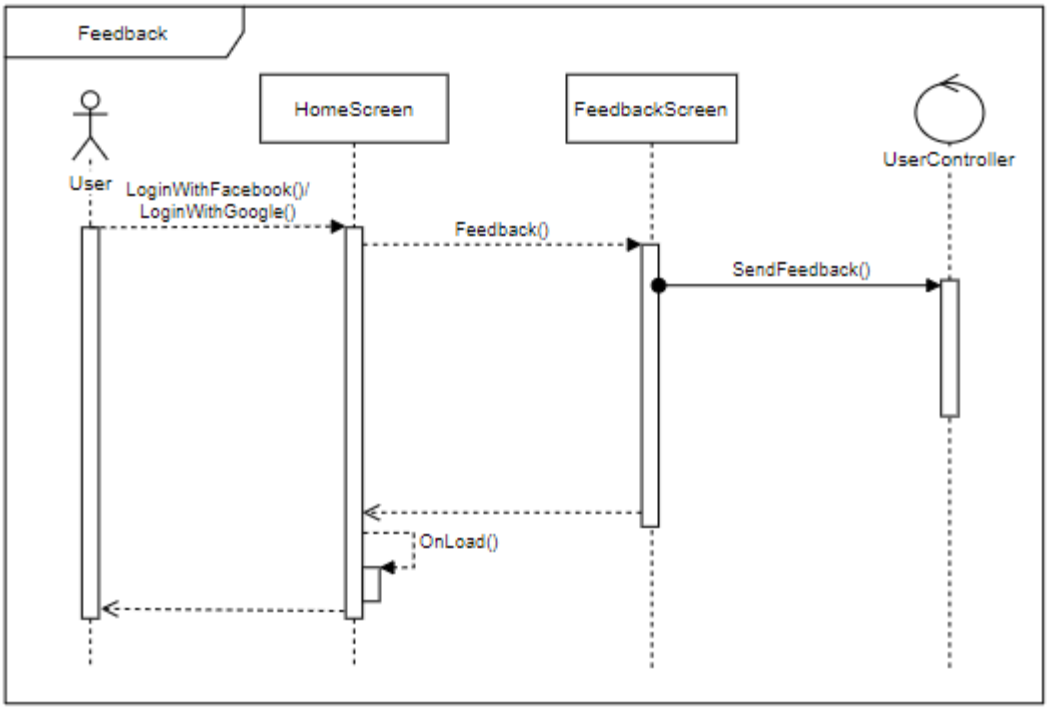


Figure 33 – Feedback sequence diagram

4.5.7. React pet

*** React pet class diagram**

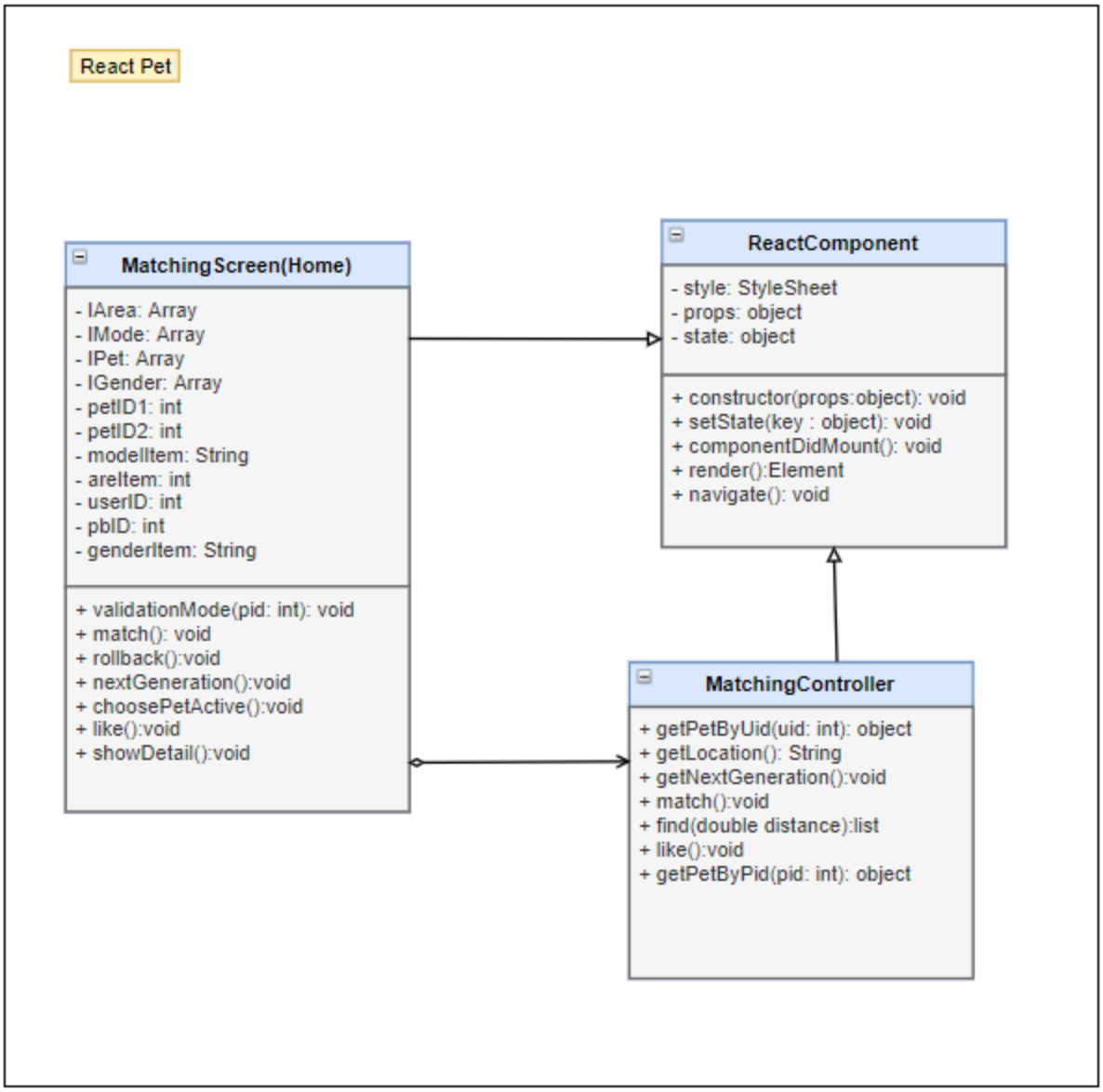


Figure 34 – React pet class diagram

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			

Name	Type	Visibility	Description
getAllPet	list	public	Get list all of pet belong to this user
getOtherPet(user)	list	public	Get list all of Pet by userID
getPetBreed	String	public	Get pet breed for pet
setPetActive	void	public	Set a pet is active to match
getAllInformation	list	public	Get all information of pet on list pet
getTopLike	list	public	Get top pet have most liked
updateMatch	void	public	Update match record in database
getTopMatch	list	public	Get top pets are most matched
getPetById(pid: int)	Object	public	Get Pet by PetID
getImageById(pid: int)	Object	public	Get all image by petID

Name	MatchingScreen	**Type**	Class
Description	Reaction pet		
Attributes			
Name	**Type**	**Visibility**	**Description**
IArea	Array	private	List location of pets
IMode	Array	private	List Mode view of pets
IPet	Array	private	List pet show
IGender	Array	private	List gender of pets
petID1	int	private	Id of pet actived
petID2	int	private	Id of pet matched
modelItem	String	private	Mode of view pets
areItem	int	private	Pet was showed or not
userID	int	private	userId of user , who own pet be matched
pbID	int	private	Pet_breed of pet active

genderItem	String	private	Gender of pet is showing
Operations			
Name	**Type**	**Visibility**	**Description**
validationMode (pid: int)	void	public	Decision pet is showed or not
match	void	public	Add match record between 2 pets to database
rollback	void	public	Rollback pet card user passed
nextGeneration	void	public	Get image of Next generation between 2 pets (parent)s
choosePetActive	void	public	Choose pet to match
like	void	public	Reaction pet on screen
showDetail	void	public	Show all information of that pet

*** React pet sequence diagram**

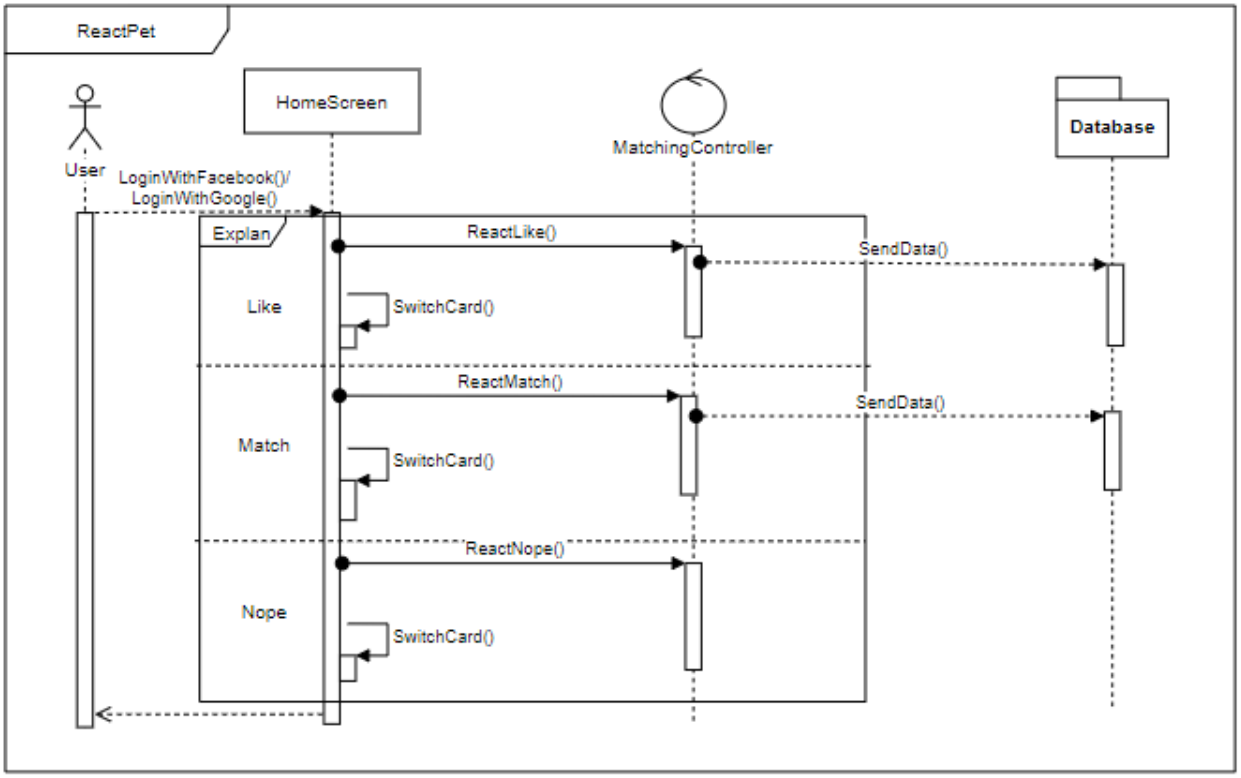


Figure 35 – React pet sequence diagram

4.5.8. View other user

*** View other user class diagram**

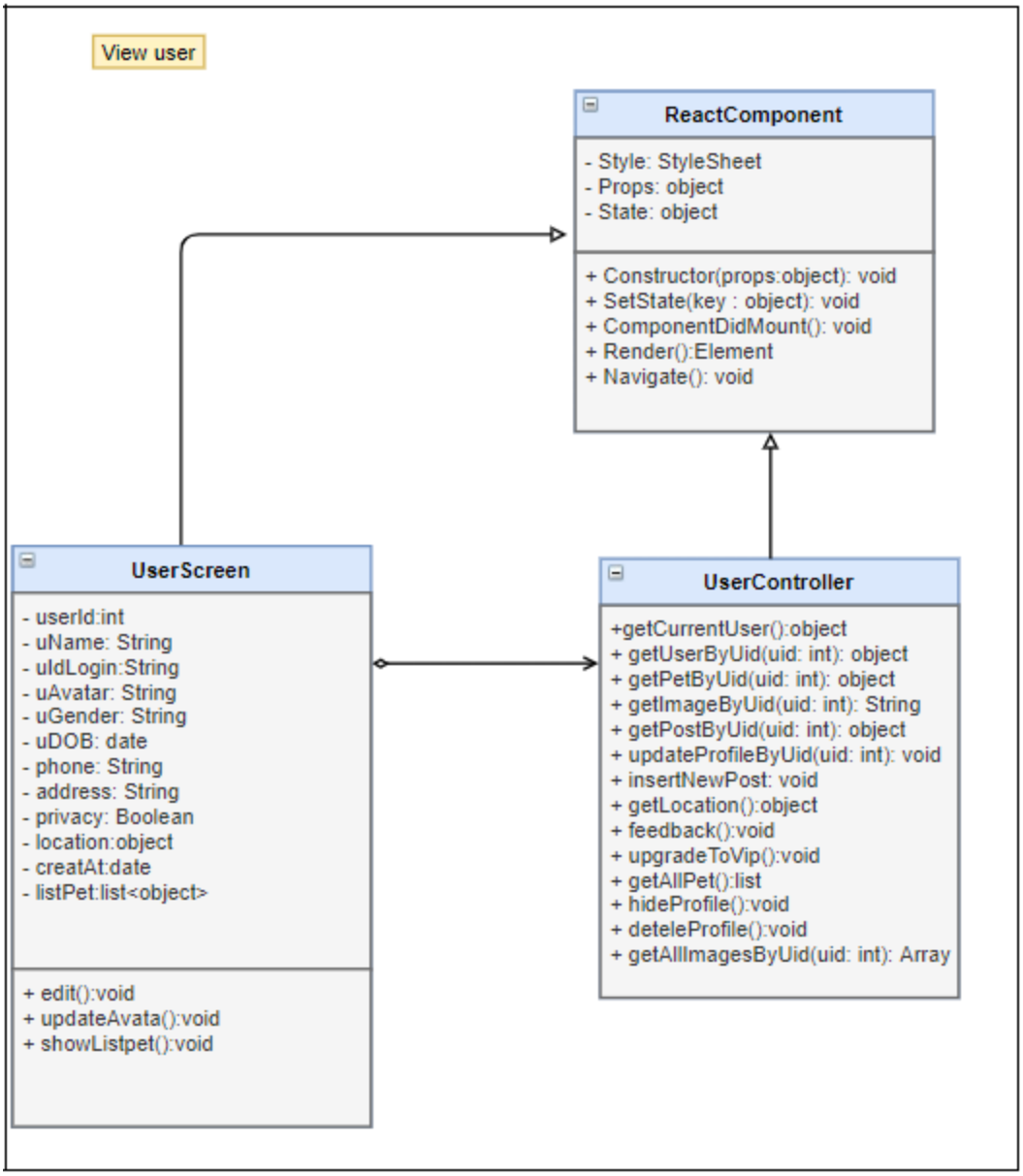


Figure 36 – View other user class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**

Operations			
Name	**Type**	**Visibility**	**Description**
getCurrentUser()	object	public	Get all information of user, which did login
getUserByUid(uID)	object	public	Get all information of user with Uid
getPetByPid(pID)	object	public	Get information of Pet with PetID
getImageByUid(())	String	public	Select image form database by Uid
updateProfileByUid()	void	public	Update detail of user profile
getLocation()	object	public	Get location(latitude and longitude) of user
getAllPet()	list	public	Get all information of list pet Pet For this user
getAllImagesByUid()	Array	public	Get list of image User has

Name	UserScreen	**Type**	Class
Description	Screen show all information of user		
Attributes			
Name	**Type**	**Visibility**	**Description**
userId	int	private	UserID of user , who you want to view
uName	String	private	User name
uIdLogin	String	private	User id login (facebook , Google)
uAvatar	String	private	User’s Avatar
uGender	String	private	User’s gender
uDOB	date	private	User’s date of birth
phone	String	private	User’s phone numbers
address	String	private	User’s address

privacy	Boolean	private	User's privacy
location	object	private	Local of user (latitude , longitude)
creatAt	date	private	Time user was created
listPet	list	private	List pet of that user
Operations			
Name	**Type**	**Visibility**	**Description**
showListPet()	void	public	Refresh and show list pets

*** View other user sequence diagram**

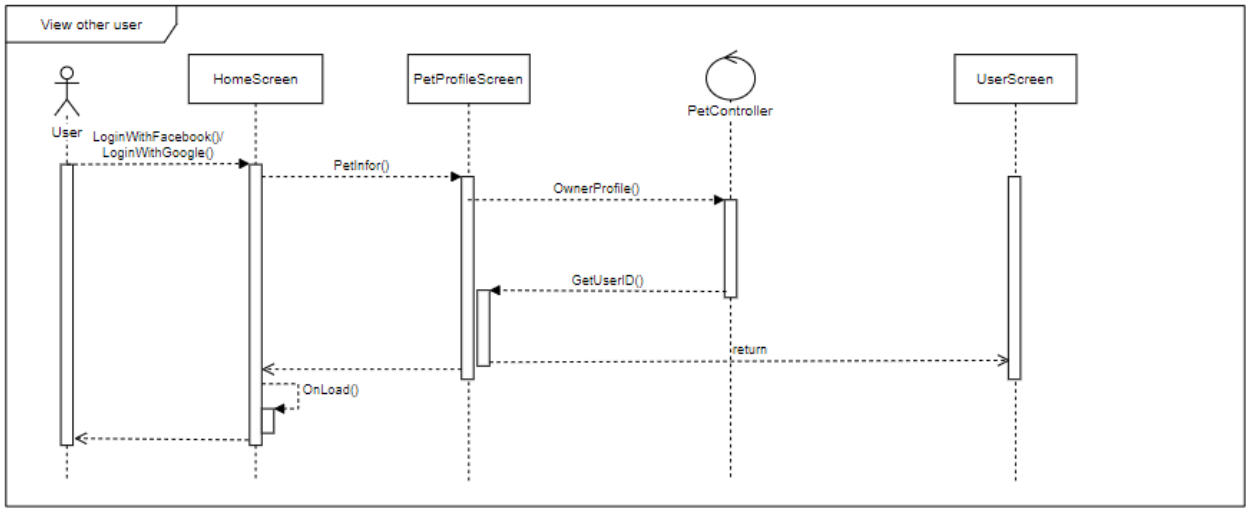


Figure 37 – View other user sequence diagram

4.5.9. View other user's pet

*** View other user's pet class diagram**

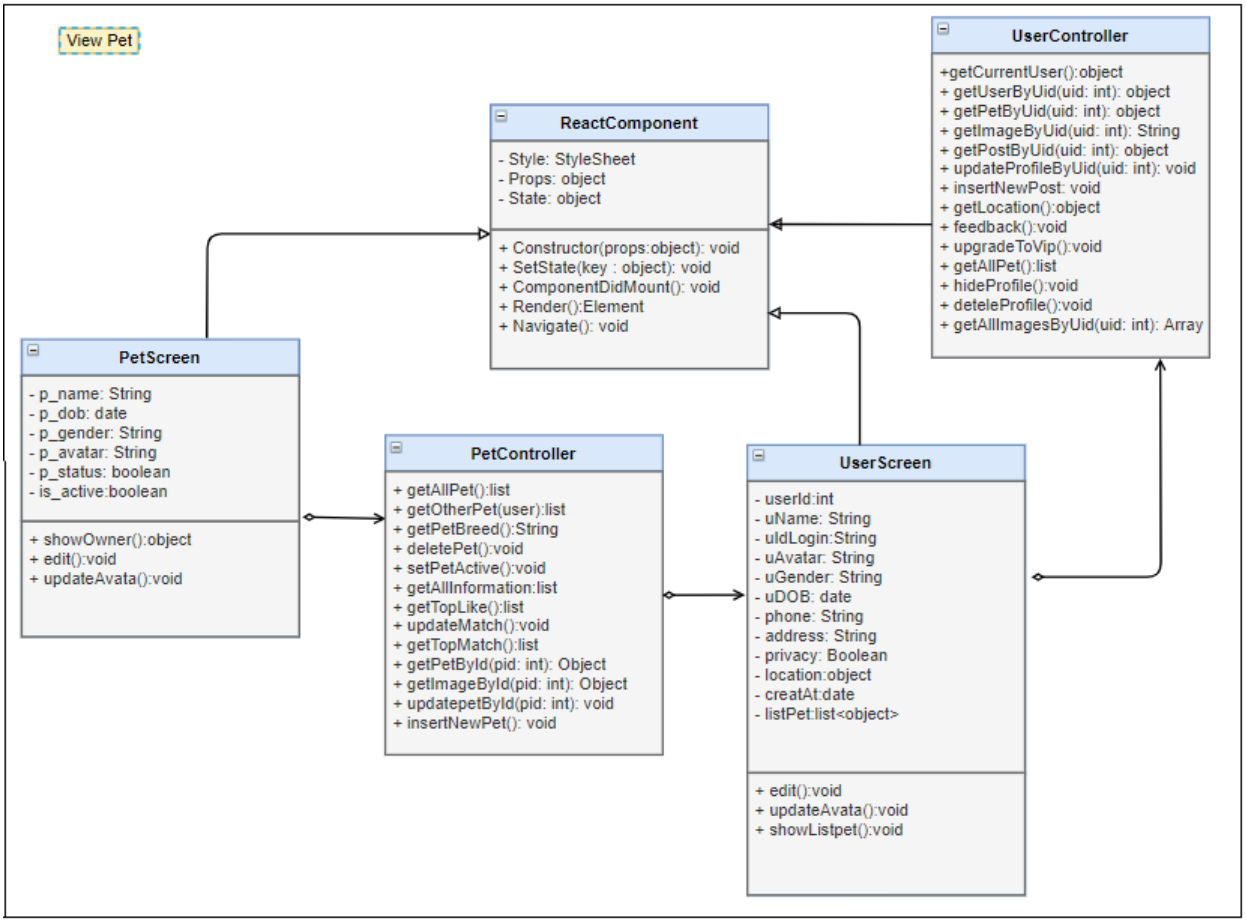


Figure 38 – View other user's pet class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getCurrentUser()	object	public	Get all information of user, which did login

getUserByUid(uID)	object	public	Get all information of user with Uid
getPetByPid(pID)	object	public	Get information of Pet with PetID
getImageByUid(())	String	public	Select image form database by Uid
updateProfileByUid()	void	public	Update detail of user profile
getLocation()	object	public	Get location(latitude and longitude) of user
getAllImagesByUid()	Array	public	Get list of image User has

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getAllPet	list	public	Get list all of pet belong to this user
getOtherPet(user)	list	public	Get list all of Pet by userID
getPetBreed	String	public	Get pet breed for pet
getAllInformation	list	public	Get all information of pet on list pet
getTopLike	list	public	Get top pet have most liked
getTopMatch	list	public	Get top pets are most matched
getPetById(pid: int)	Object	public	Get Pet by PetID
getImageById(pid: int)	Object	public	Get all image by petID
insertNewPet()	void	public	Create a new pet

Name	UserScreen	**Type**	Class
Description	Screen show all information of user		

Attributes			
Name	**Type**	**Visibility**	**Description**
userId	int	private	UserID of user , who you want to view
uName	String	private	User name
uIdLogin	String	private	User id login (facebook , Google)
uAvatar	String	private	User’s Avatar
uGender	String	private	User’s gender
uDOB	date	private	User’s date of birth
phone	String	private	User’s phone numbers
address	String	private	User’s address
privacy	Boolean	private	User’s privacy
location	object	private	Local of user (latitude , longitude)
creatAt	date	private	Time user was created
listPet	list	private	List pet of that user

Operations			
Name	**Type**	**Visibility**	**Description**
showListPet()	void	public	Refresh and show list pets

Name	PetScreen	**Type**	Class
Description	Screen show detail information of pet		

Attributes			
Name	**Type**	**Visibility**	**Description**
p_name	String	private	Name of pet

p_dob	date	private	Pet's date of birth
p_gender	String	private	Pet's gender
p_avatar	String	private	Pet's avata
p_status	boolean	private	Pet's status
is_active	boolean	private	Pet's is active
Operations			
Name	**Type**	**Visibility**	**Description**
showOwner	object	public	Show information of user , who own pet

*** View other user's pet sequence diagram**

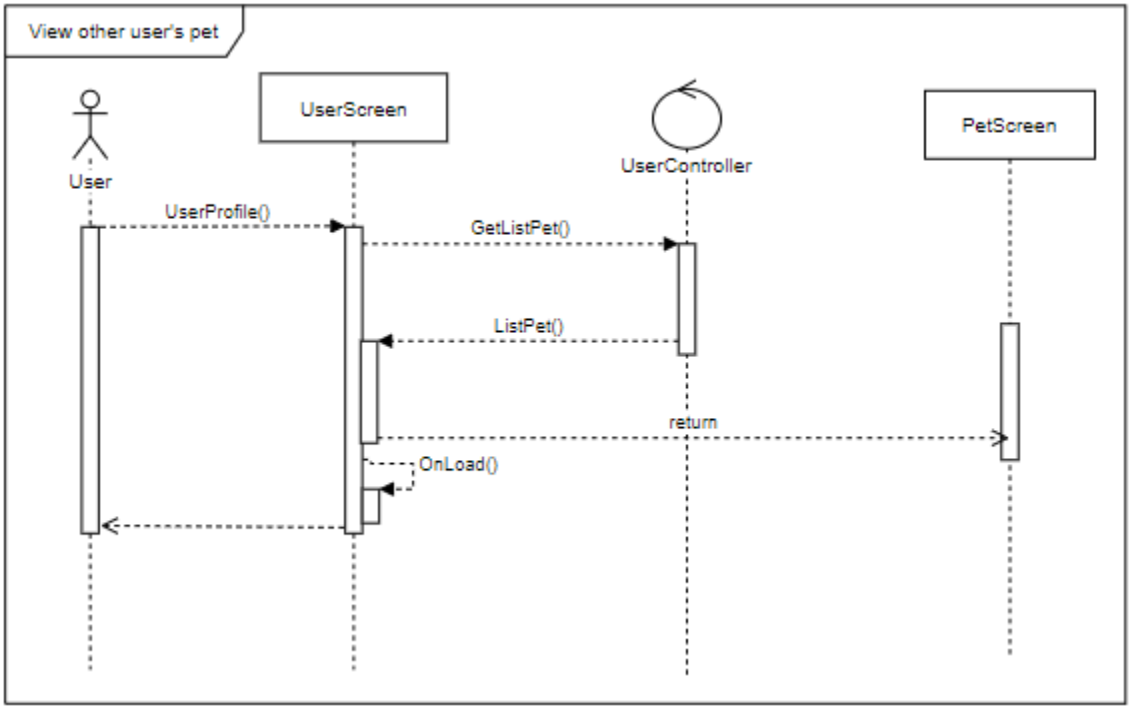


Figure 39 – View other user's pet sequence diagram

4.5.10. Hide profile

*** Hide profile class diagram**

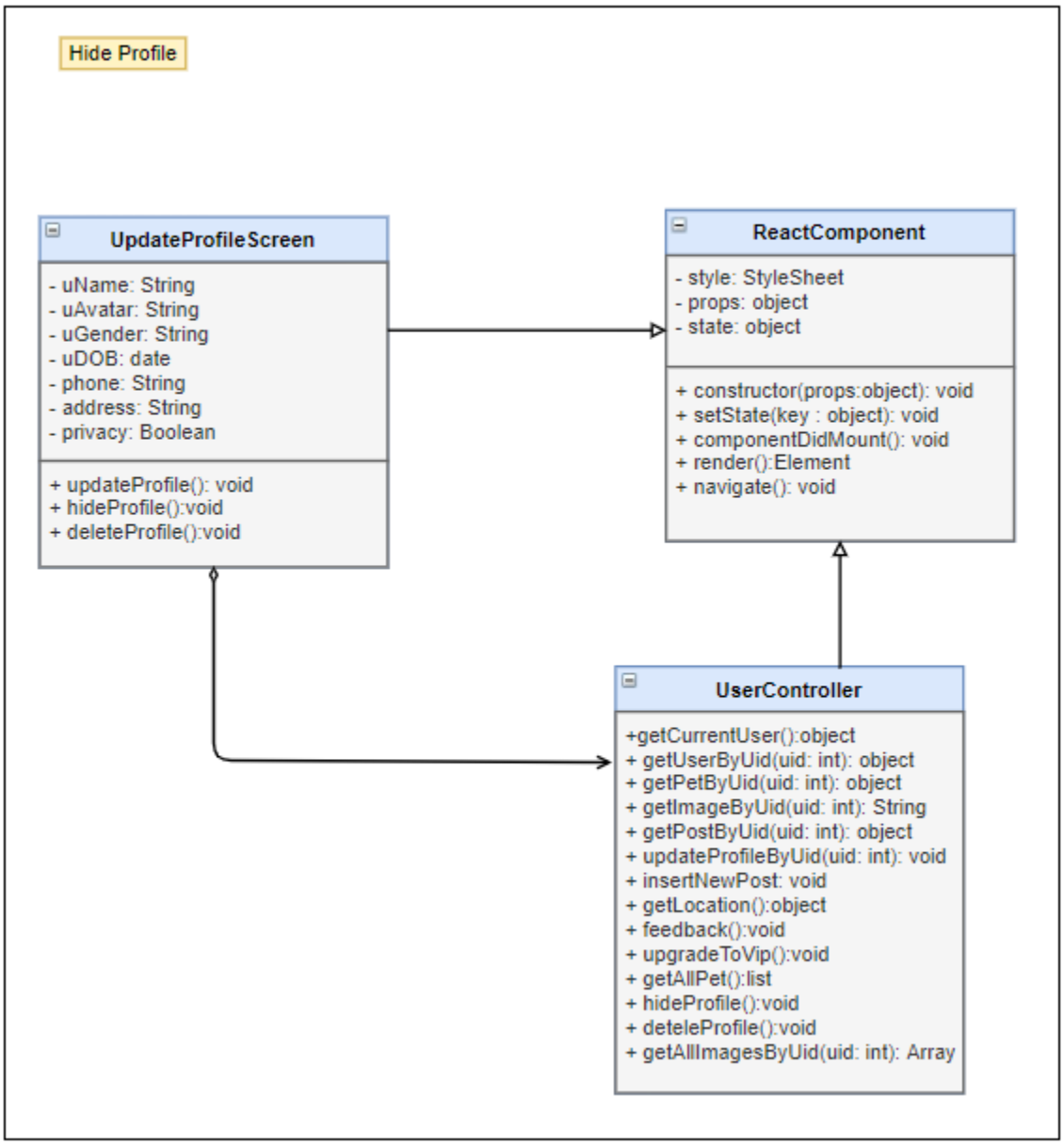


Figure 40 – Hide profile class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			

Name	Type	Visibility	Description
Operations			
Name	**Type**	**Visibility**	**Description**
getCurrentUser()	object	public	Get all information of user, which did login
getUserByUid(uID)	object	public	Get all information of user with Uid
getImageByUid(())	String	public	Select image form database by Uid
updateProfileByUid()	void	public	Update detail of user profile
getLocation()	object	public	Get location(latitude and longitude) of user
hideProfile()	void	public	Hide of unhide user profile
getAllImagesByUid()	Array	public	Get list of image User has

Name	Update profile Screen	Type	Class
Description	Screen update information of user profile		
Attributes			
Name	**Type**	**Visibility**	**Description**
uName	String	Private	User name
uAvatar	String	Private	User's Avatar
uGender	String	Private	User's gender
uDOB	date	Private	User's date of birth
phone	String	Private	User's phone numbers
address	String	Private	User's address
privacy	Boolean	Private	User's privacy
Operations			
Name	**Type**	**Visibility**	**Description**

hideProfile()	void	Public	Hide user profile

*** Hide profile sequence diagram**

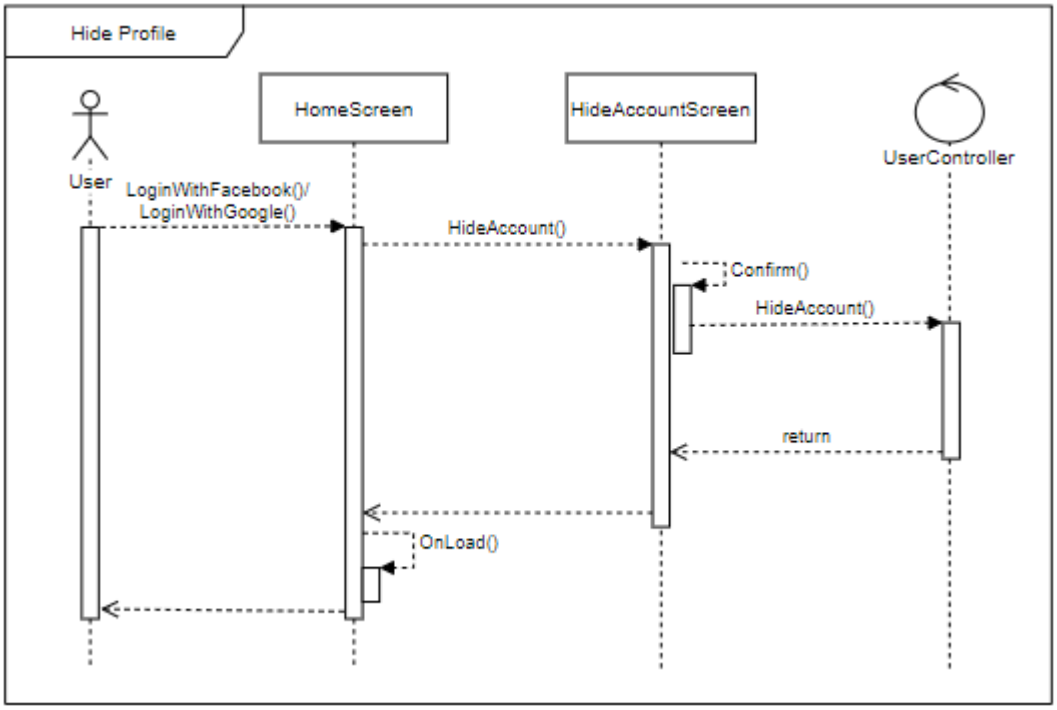


Figure 41 – Hide profile sequence diagram

4.5.11. Edit user's profile

*** Edit pet's profile class diagram**

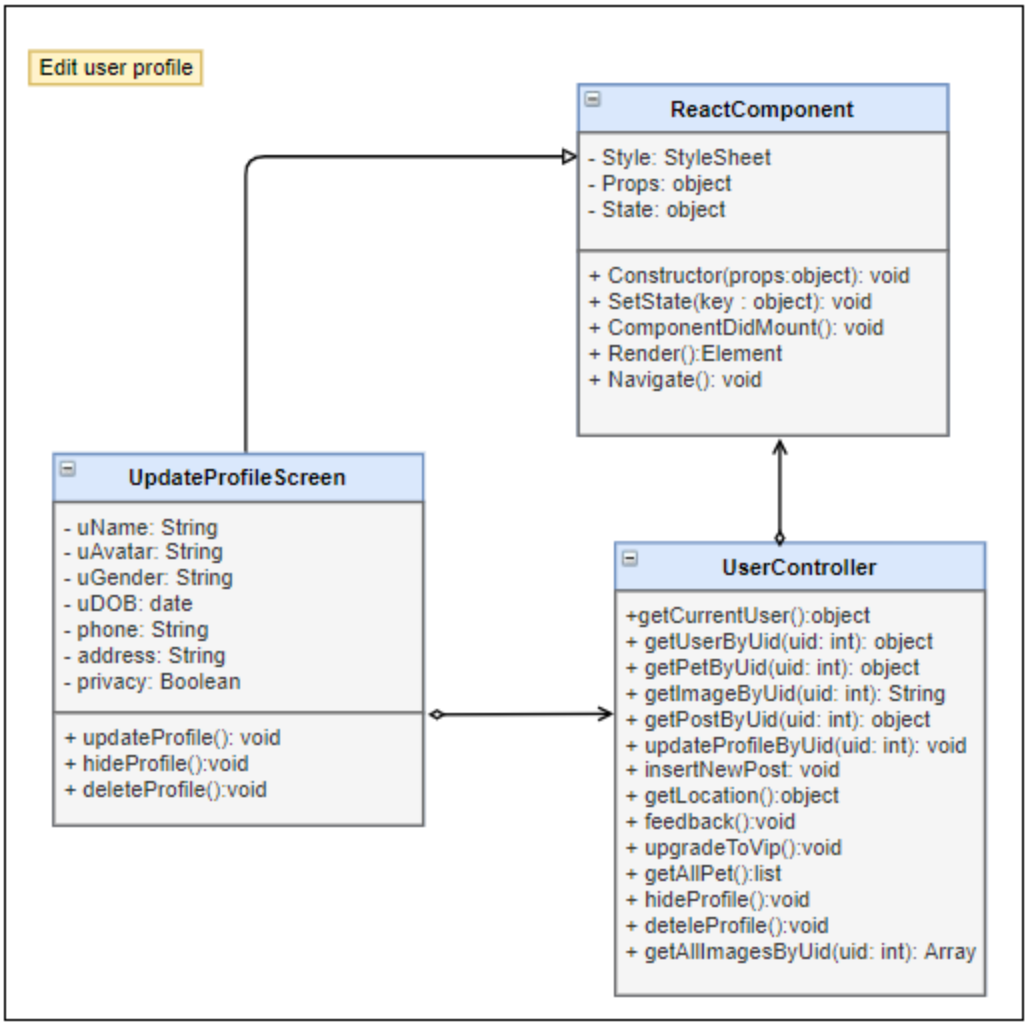


Figure 42 – Edit user's pet class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**

getCurrentUser()	object	public	Get all information of user, which did login
getPetByPid(pID)	object	public	Get information of Pet with PetID
getImageByUid(())	String	public	Select image form database by Uid
updateProfileByUid()	void	public	Update detail of user profile
getLocation()	object	public	Get location(latitude and longitude) of user
upgradeToVip()	void	public	Update user role to Vip user
getAllPet()	list	public	Get all information of list pet Pet For this user
deteleProfile()	void	public	Delete physical all of information for this user profile
getAllImagesByUid()	Array	public	Get list of image User has

Name	Update profile Screen	**Type**	Class
Description	Screen update information of user profile		
Attributes			
Name	**Type**	**Visibility**	**Description**
uName	String	private	User name
uAvatar	String	private	User's Avatar
uGender	String	private	User's gender
uDOB	date	private	User's date of birth
phone	String	private	User's phone numbers
address	String	private	User's address
privacy	Boolean	private	User's privacy
Operations			
Name	**Type**	**Visibility**	**Description**

updateProfile	void	Public	Update user profile

*** Edit user's profile sequence diagram**

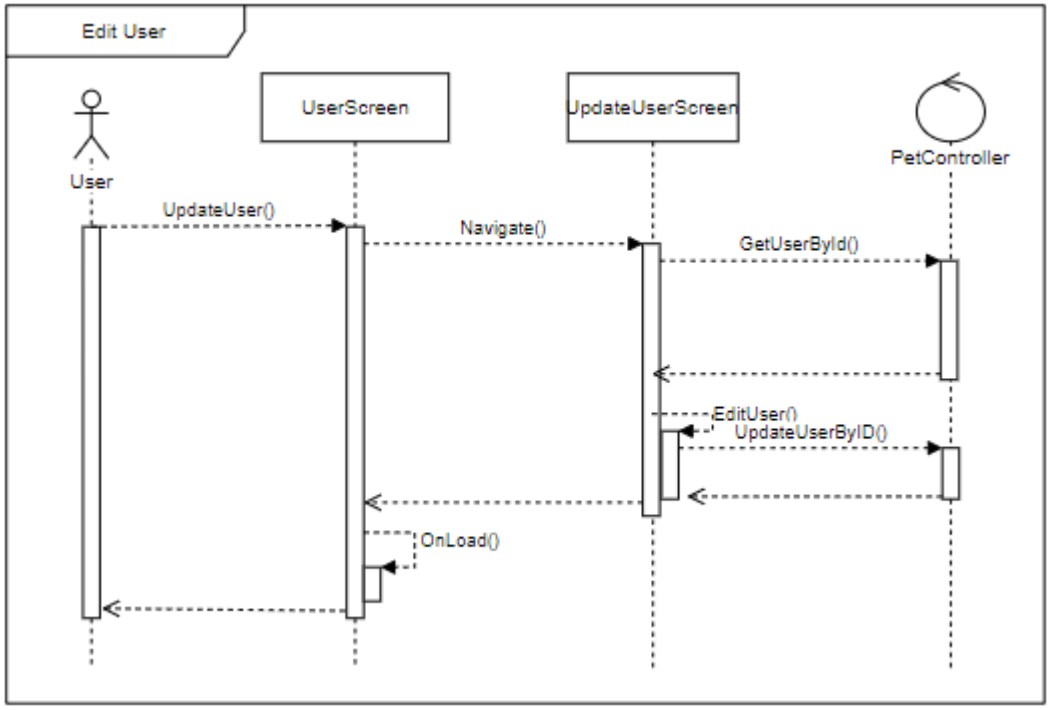


Figure 43 – Edit user's profile sequence diagram

4.5.12. Delete user's profile

*** Delete user's profile class diagram**

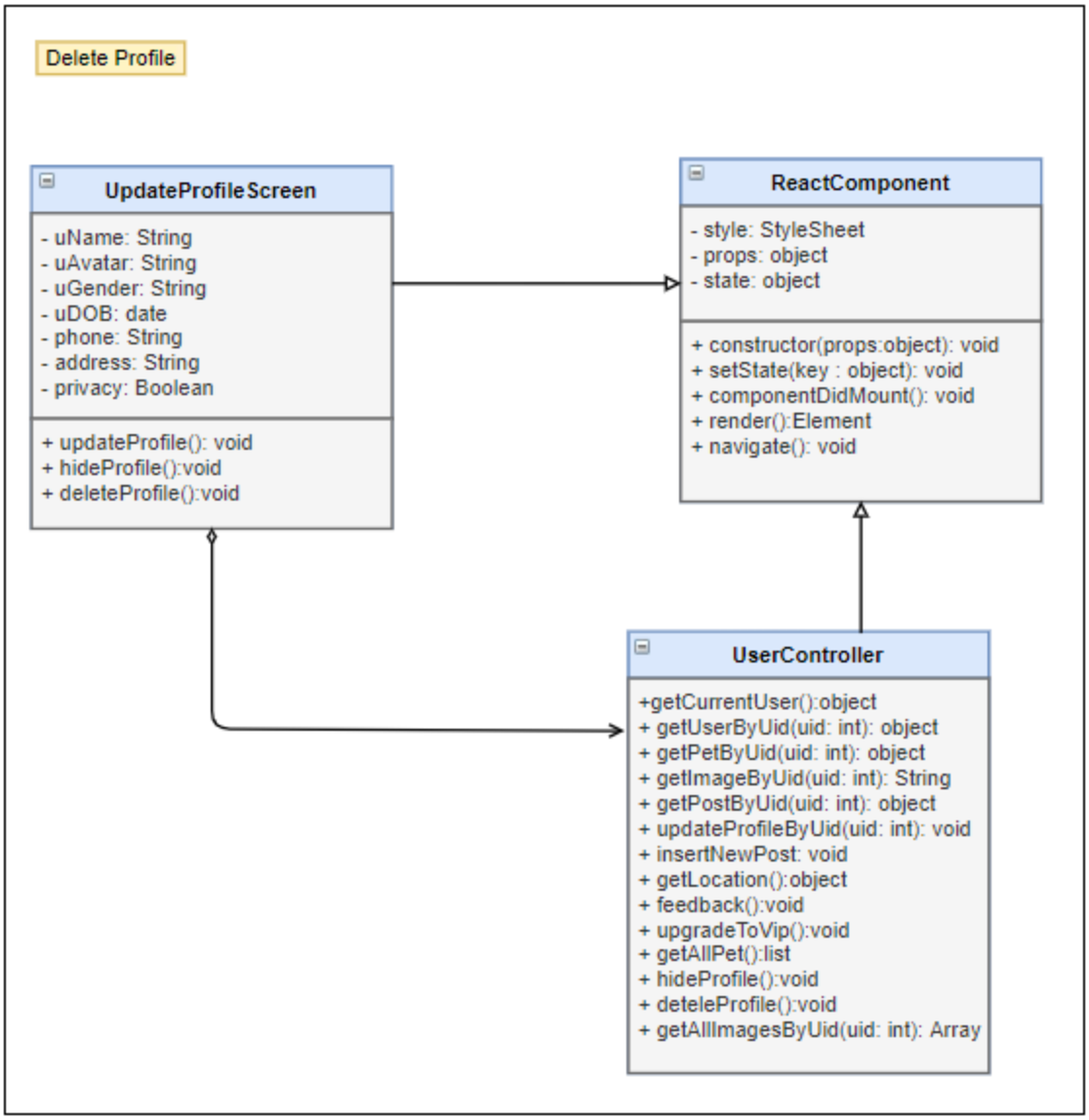


Figure 44 – Delete user's profile class diagram

Name	UserController	**Type**	Class
Description	Controller class to control Matching and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			

Name	Type	Visibility	Description
getCurrentUser()	object	public	Get all information of user, which did login
getImageByUid(())	String	public	Select image form database by Uid
updateProfileByUid()	void	public	Update detail of user profile
getLocation()	object	public	Get location(latitude and longitude) of user
deteleProfile()	void	public	Delete physical all of information for this user profile
getAllImagesByUid()	Array	public	Get list of image User has

Name	Update profile Screen	Type	Class
Description	Screen update information of user profile		

Attributes			
Name	**Type**	**Visibility**	**Description**
uName	String	private	User name
uAvatar	String	private	User's Avatar
uGender	String	private	User's gender
uDOB	date	private	User's date of birth
phone	String	private	User's phone numbers
address	String	private	User's address
privacy	Boolean	private	User's privacy

Operations			
Name	**Type**	**Visibility**	**Description**
deleteProfile()	void	Public	Delete user profile

*** Delete user's profile sequence diagram**

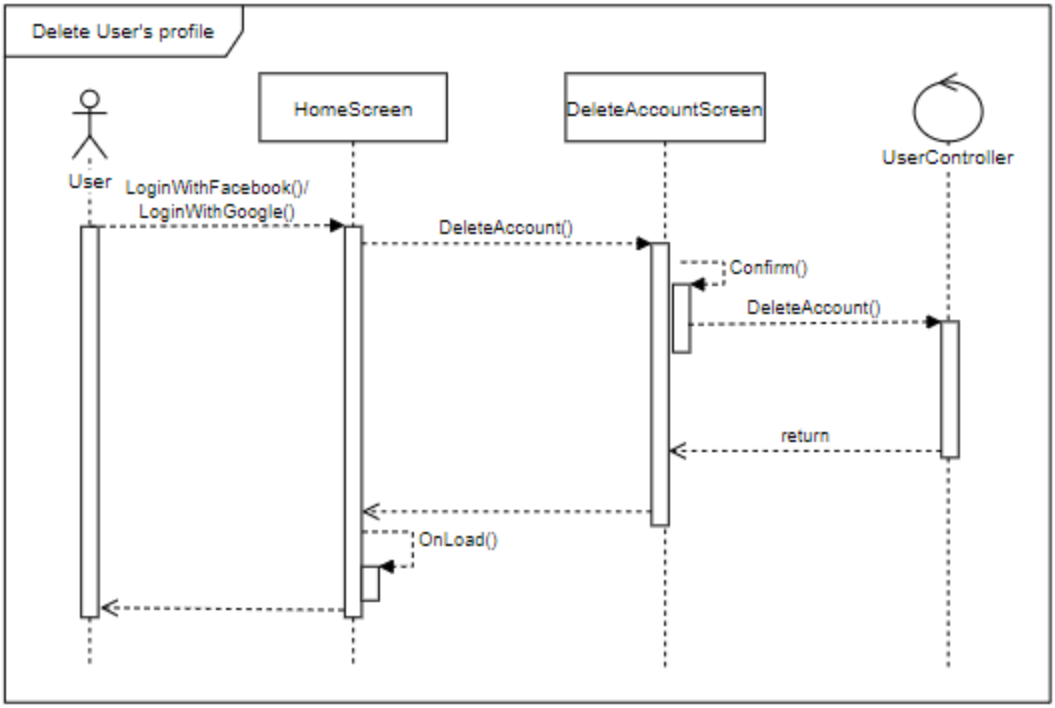


Figure 45 – Delete user's profile sequence diagram

4.5.13. Create new pet

*** Create new pet class diagram**

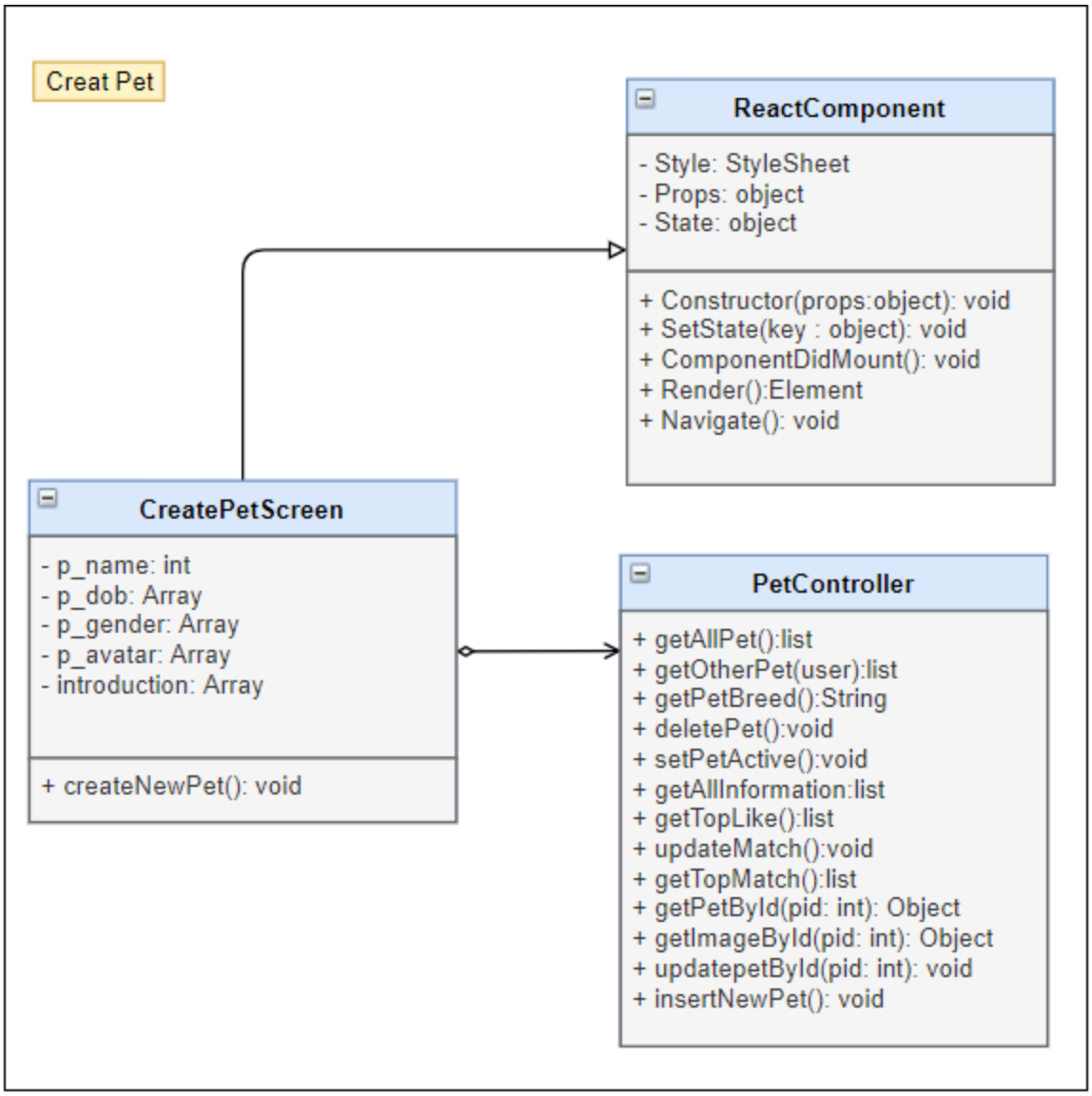


Figure 46 – Create new pet class diagram

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			

Name	Type	Visibility	Description
getAllPet	list	Public	Get list all of pet belong to this user
getPetBreed	String	Public	Get pet breed for pet
setPetActive	void	Public	Set a pet is active to match
getAllInformation	list	Public	Get all information of pet on list pet
getPetById(pid: int)	Object	Public	Get Pet by PetID
getImageById(pid: int)	Object	Public	Get all image by petID
insertNewPet()	void	Public	Create a new pet

Name	CreatePetScreen	Type	Class
Description	Screen create pet		
Attributes			
Name	**Type**	**Visibility**	**Description**
p_name	String	private	Name of pet
p_dob	date	private	Pet's date of birth
p_gender	String	private	Pet's gender
p_avatar	String	private	Pet's avatar
Introduction	String	Private	Pet's introduction
Operations			
Name	**Type**	**Visibility**	**Description**
createNewPet()	void	public	Create new pet of user login

*** Create new pet sequence diagram**

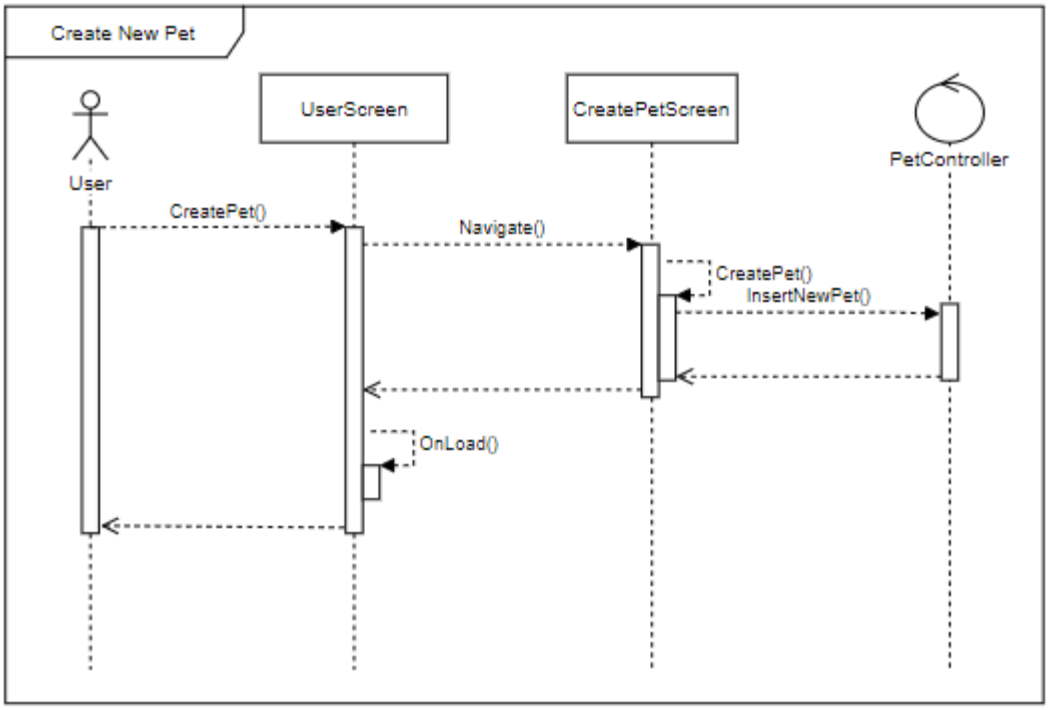


Figure 47 – Create new pet sequence diagram

4.5.14. Edit pet's profile

*** Edit pet's profile class diagram**

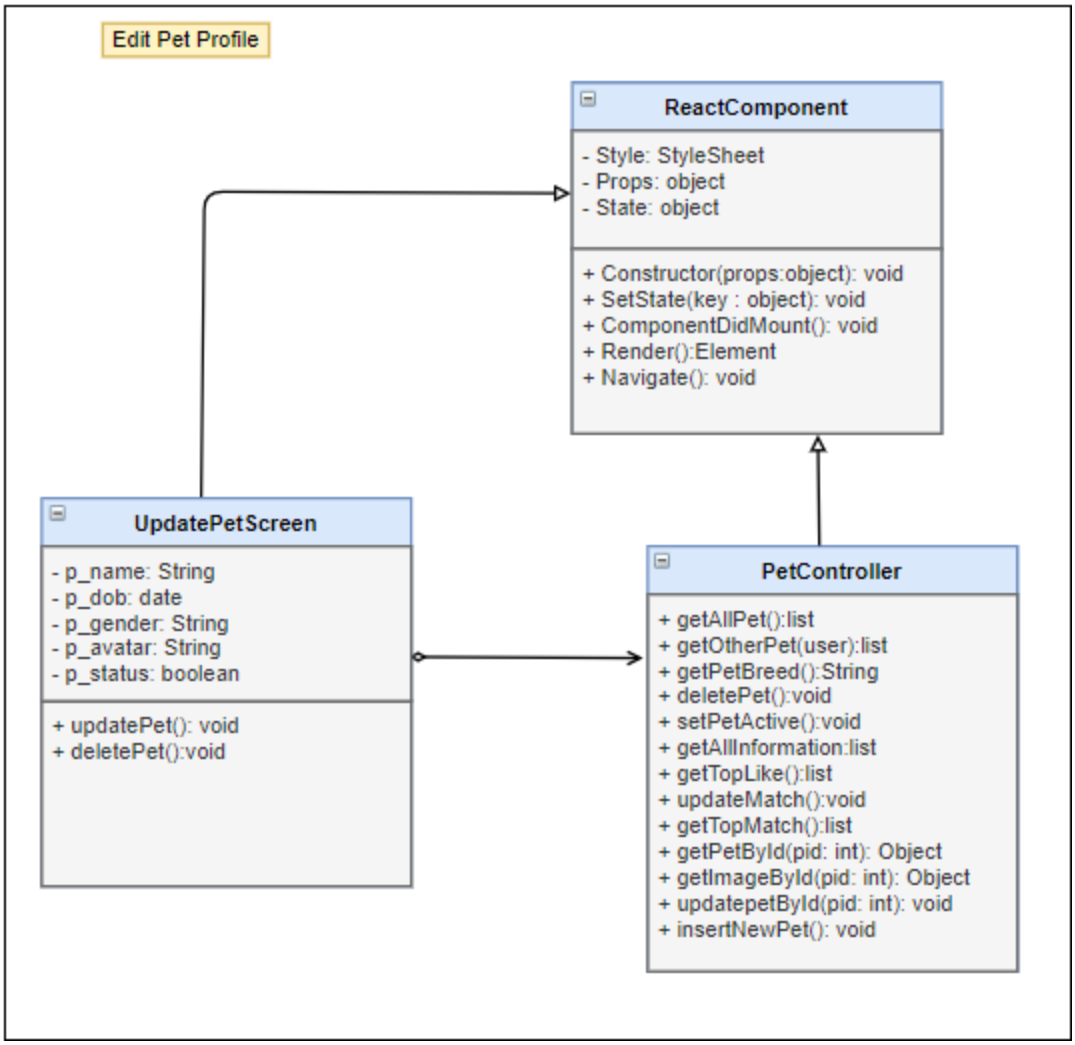


Figure 48 – Edit pet's profile class diagram

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getAllPet	list	public	Get list all of pet belong to this user

getOtherPet(user)	list	public	Get list all of Pet by userID
getPetBreed	String	public	Get pet breed for pet
setPetActive	void	public	Set a pet is active to match
getAllInformation	list	public	Get all information of pet on list pet
getPetById(pid: int)	Object	public	Get Pet by PetID
getImageById(pid: int)	Object	public	Get all image by petID
updatepetById(pid: int)	void	public	Update Pet by petID

Name	UpdatePetScreen	**Type**	Class
Description	Screen update pet's information		
Attributes			
Name	**Type**	**Visibility**	**Description**
p_name	String	private	Name of pet
p_dob	date	private	Pet's date of birth
p_gender	String	private	Pet's gender
p_avatar	String	private	Pet's avata
p_status	boolean	private	Pet's status
Operations			
Name	**Type**	**Visibility**	**Description**
updatePet()	void	public	Update pet function

*** Edit pet's profile sequence diagram**

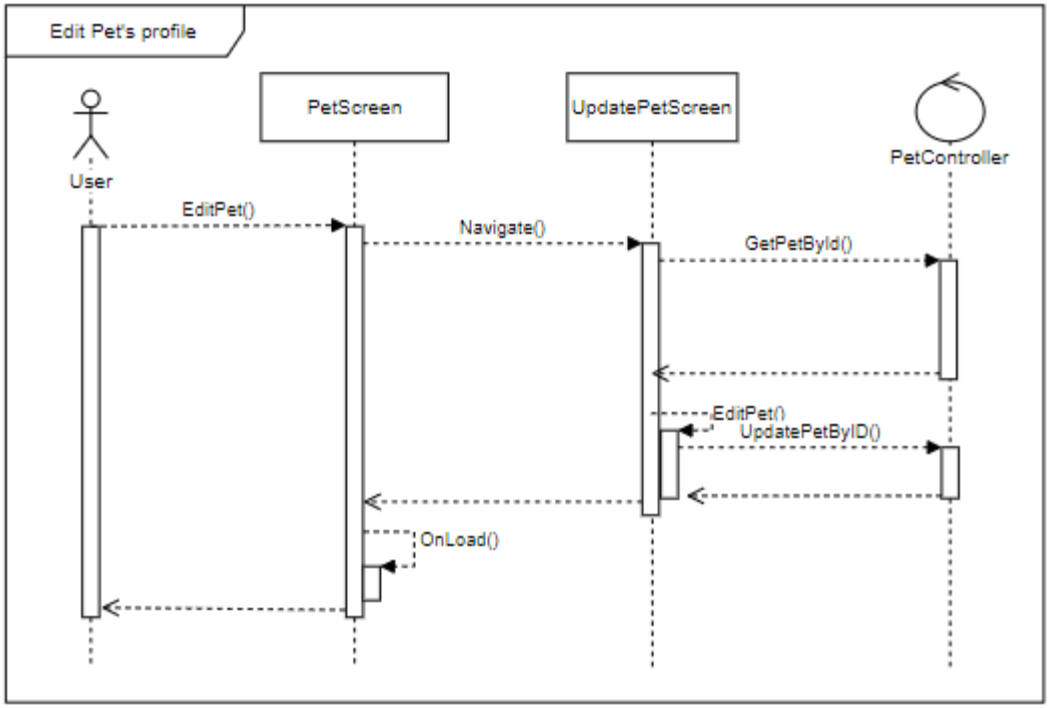


Figure 49 – Edit pet's profile sequence diagram

4.5.15. Delete pet's profile

*** Delete pet's profile class diagram**

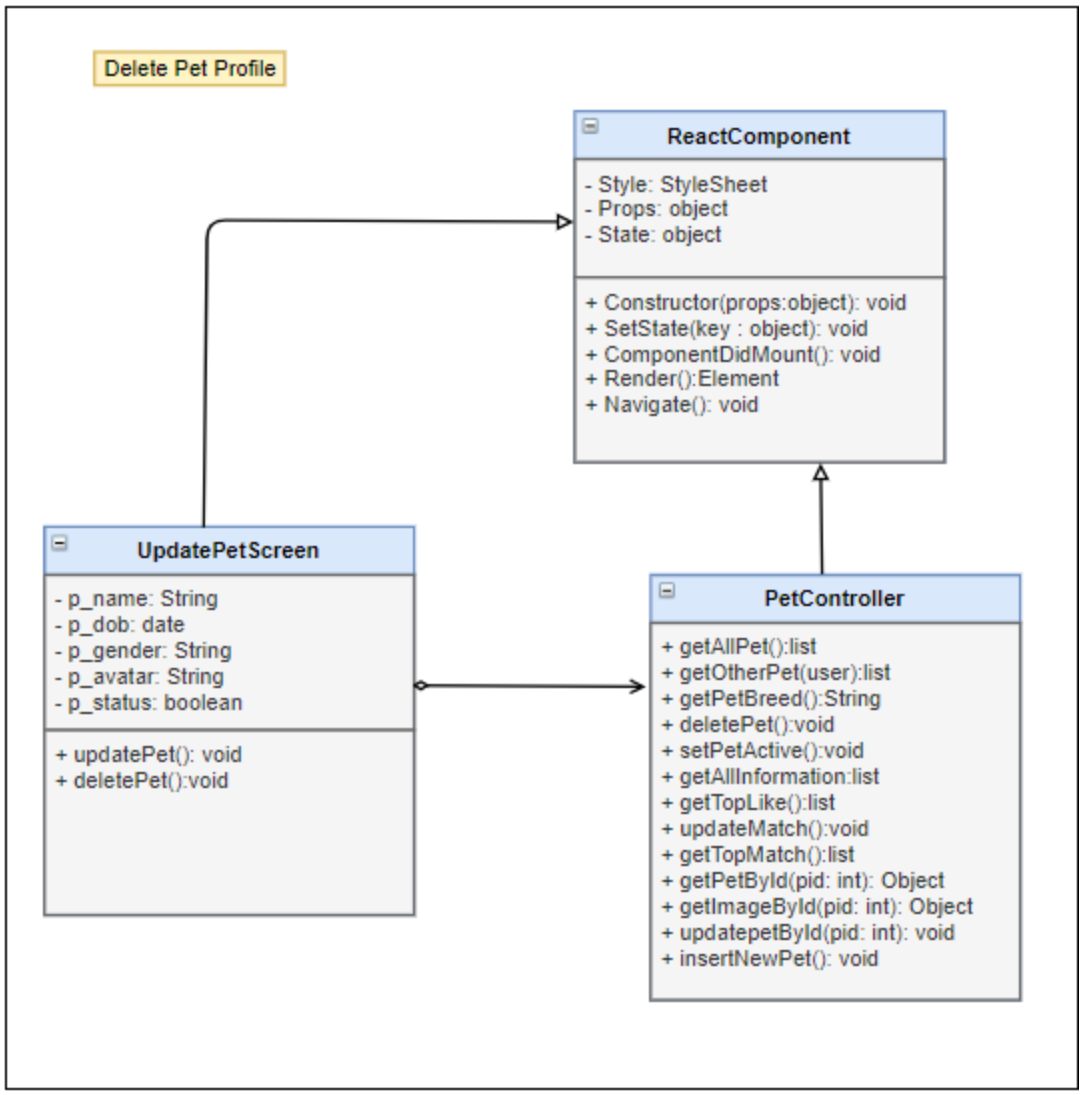


Figure 50 – Delete pet's class diagram

Name	PetController	**Type**	Class
Description	Controller class to control Pet and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**

getAllPet	list	public	Get list all of pet belong to this user
getOtherPet(user)	list	public	Get list all of Pet by userID
getPetBreed	String	public	Get pet breed for pet
deletePet	void	public	Delete a pet in list pets
setPetActive	void	public	Set a pet is active to match
getAllInformation	list	public	Get all information of pet on list pet
getPetById(pid: int)	Object	public	Get Pet by PetID
getImageById(pid: int)	Object	public	Get all image by petID

Name	UpdatePetScreen	**Type**	Class
Description	Screen update pet's information		
Attributes			
Name	**Type**	**Visibility**	**Description**
p_name	String	private	Name of pet
p_dob	date	private	Pet's date of birth
p_gender	String	private	Pet's gender
p_avatar	String	private	Pet's avata
p_status	boolean	private	Pet's status
Operations			
Name	**Type**	**Visibility**	**Description**
deletePet()	void	public	Delete pet function

*** Delete pet's profile sequence diagram**

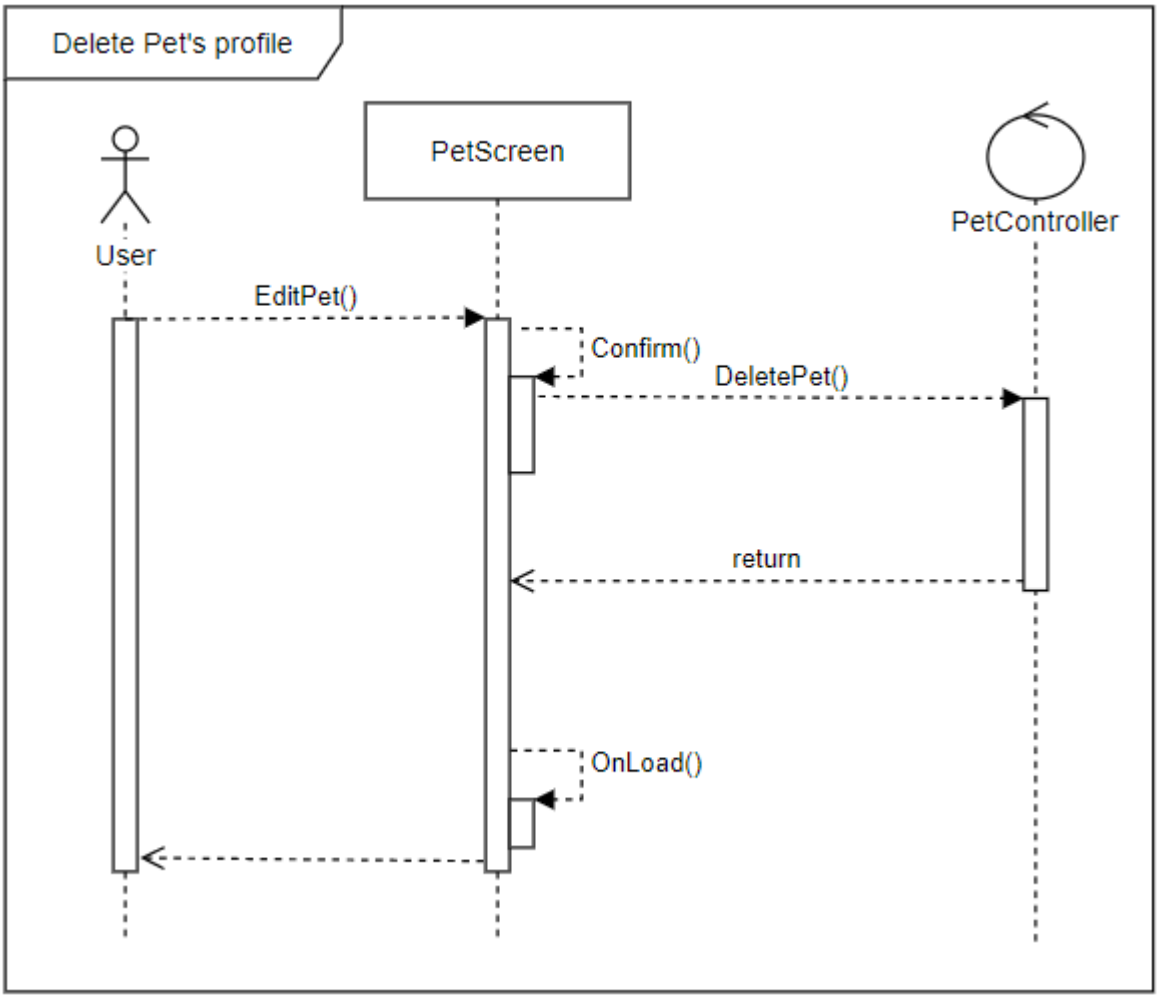


Figure 51 – Delete pet's profile sequence diagram

4.5.16. Find

*** Find class diagram**

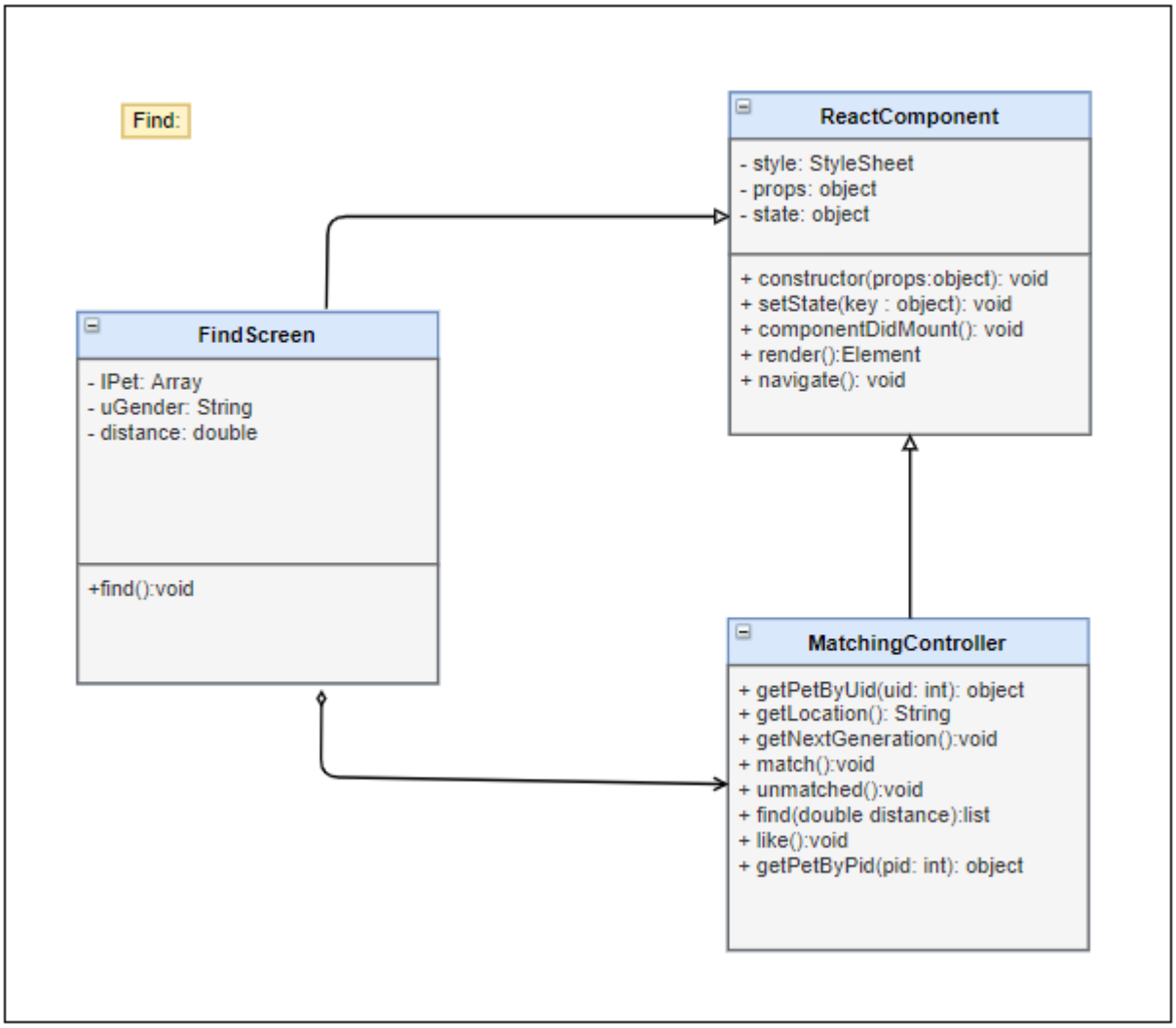


Figure 52 – Find class diagram

Name	MatchingController	**Type**	Class
Description	Controller class to control User and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getPetByUid (id)	object	public	Get Information of list pet with UserId in database

getLocation ()	void	public	Get real location of user
find(double distance)	list	public	find user around with distance
like()	void	public	Reaction a pet to rankking or show
getPetByPid(pid: int)	object	public	Get Information of pet with PetId in database

Name	FindScreen	**Type**	Class
Description	Screen provide Find pet/user feature		
Attributes			
Name	**Type**	**Visibility**	**Description**
lPet	Array	private	List of pet result
uGender	String	private	User gender
distance	double	private	Distance device to other users
Operations			
Name	**Type**	**Visibility**	**Description**
find()	void	public	Find user/pet around device

*** Find sequence diagram**

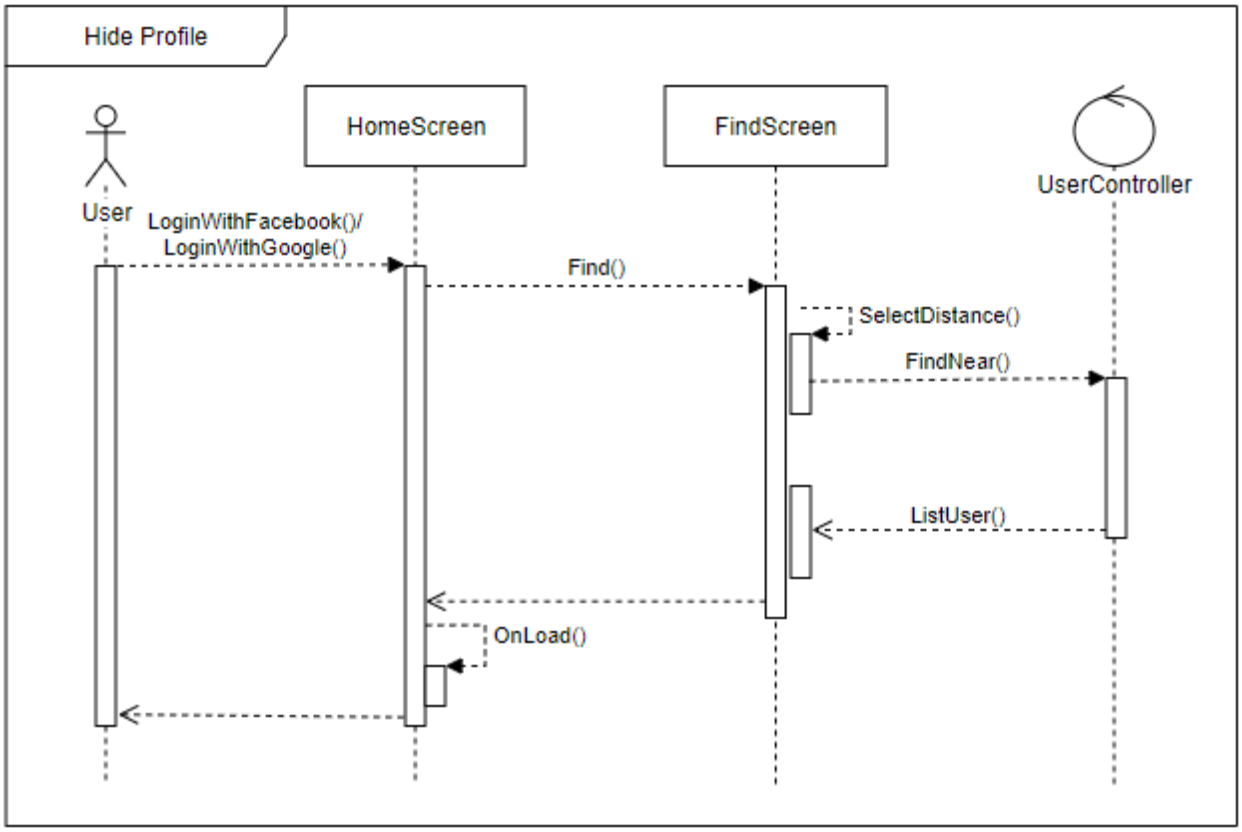


Figure 53 – Find sequence diagram

4.5.17. Get next generation pet

*** Get new generation class diagram**

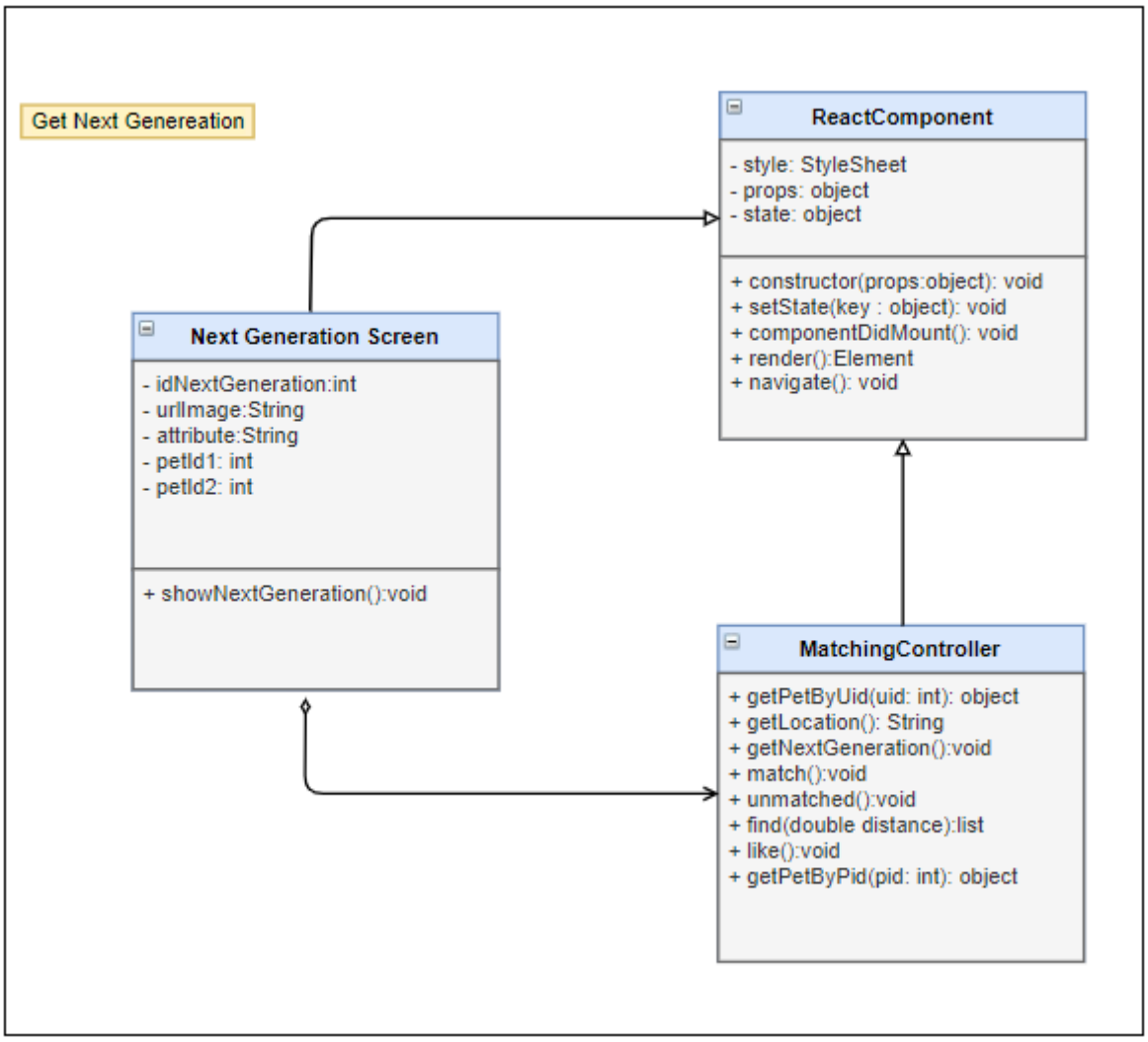


Figure 54 – Get next generation pet class diagram

Name	MatchingController	**Type**	Class
Description	Controller class to control User and some functions relative.		
Attributes			
Name	**Type**	**Visibility**	**Description**
Operations			
Name	**Type**	**Visibility**	**Description**
getPetByUid (id)	object	public	Get Information of list pet with UserId in database

getLocation ()	void	public	Get real location of user
getNextGeneration()	void	public	Get new baby pet for 2 parent pets
find(double distance)	list	public	find user around with distance
getPetByPid(pid: int)	object	public	Get Information of pet with PetId in database

Name		**Type**	Class
Description			
Attributes			
Name	**Type**	**Visibility**	**Description**
idNextGeneration	int	private	Id of pet is next generation
urlImage	String	private	url Image of pet is next generation
attribute	String	private	Attribute of next generation
petId1	int	private	Id of parent
petId2	int	private	Id of parent
Operations			
Name	**Type**	**Visibility**	**Description**
showNextGeneration()	void	public	Show image pet is next generation

*** Get new generation sequence diagram**

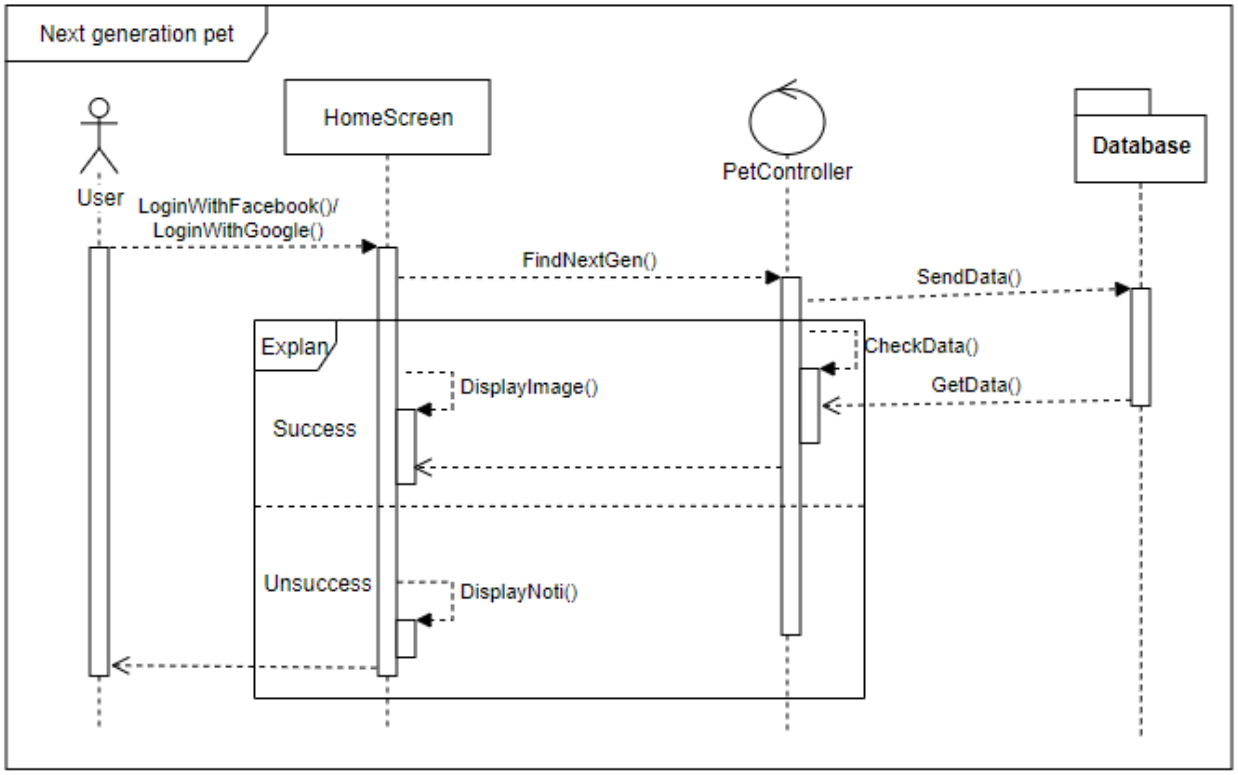


Figure 55 – Get new generation pet sequence diagram

4.5.18. Login/Logout for Admin

*** Login/Logout for Admin class diagram**

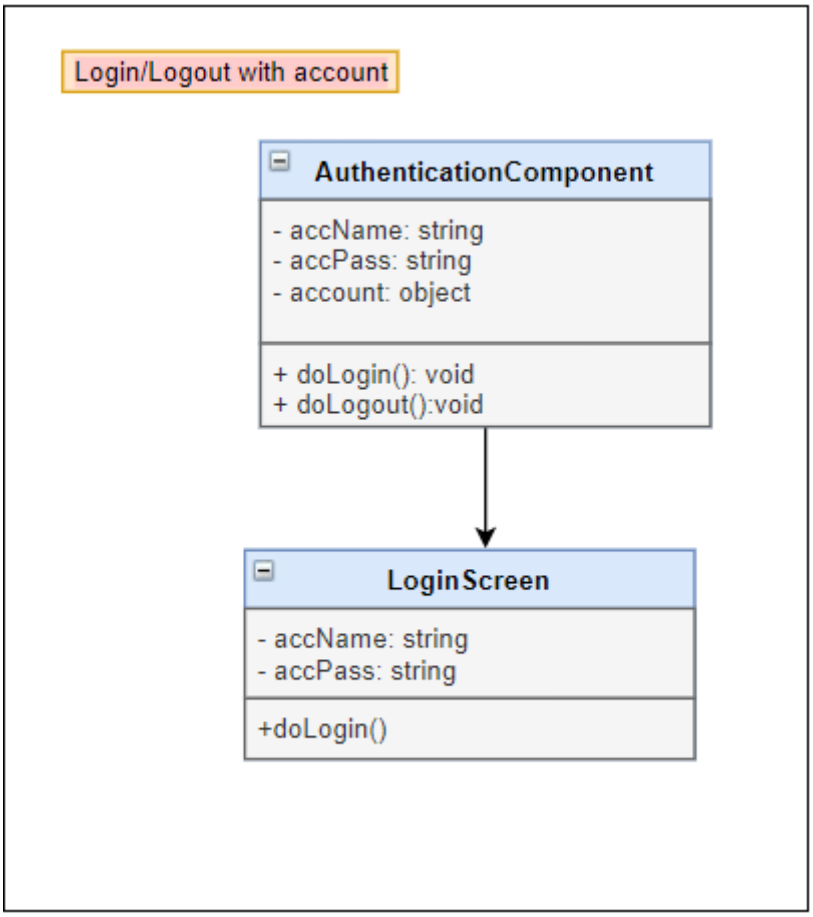


Figure 56 – Login/Logout with account

Name	AuthenticationComponent		**Type**	Class
Description	Authentication controller for admin			
Attributes				
Name	**Type**	**Visibility**	**Description**	
accName	String	private	User name	
accPass	String	private	Password	
account	object	private	Account and all of it's information	
Operations				
Name	**Type**	**Visibility**	**Description**	
doLogin()	void	public	Login function	

doLoout()	void	public	Logout function

Name	LoginScreen	**Type**	Class
Description	Screen login for admin		
Attributes			
Name	**Type**	**Visibility**	**Description**
accName	String	private	User name
accPass	String	private	Password
Operations			
Name	**Type**	**Visibility**	**Description**
doLogin	void	public	Login function

*** Login/Logout for Admin sequence diagram**

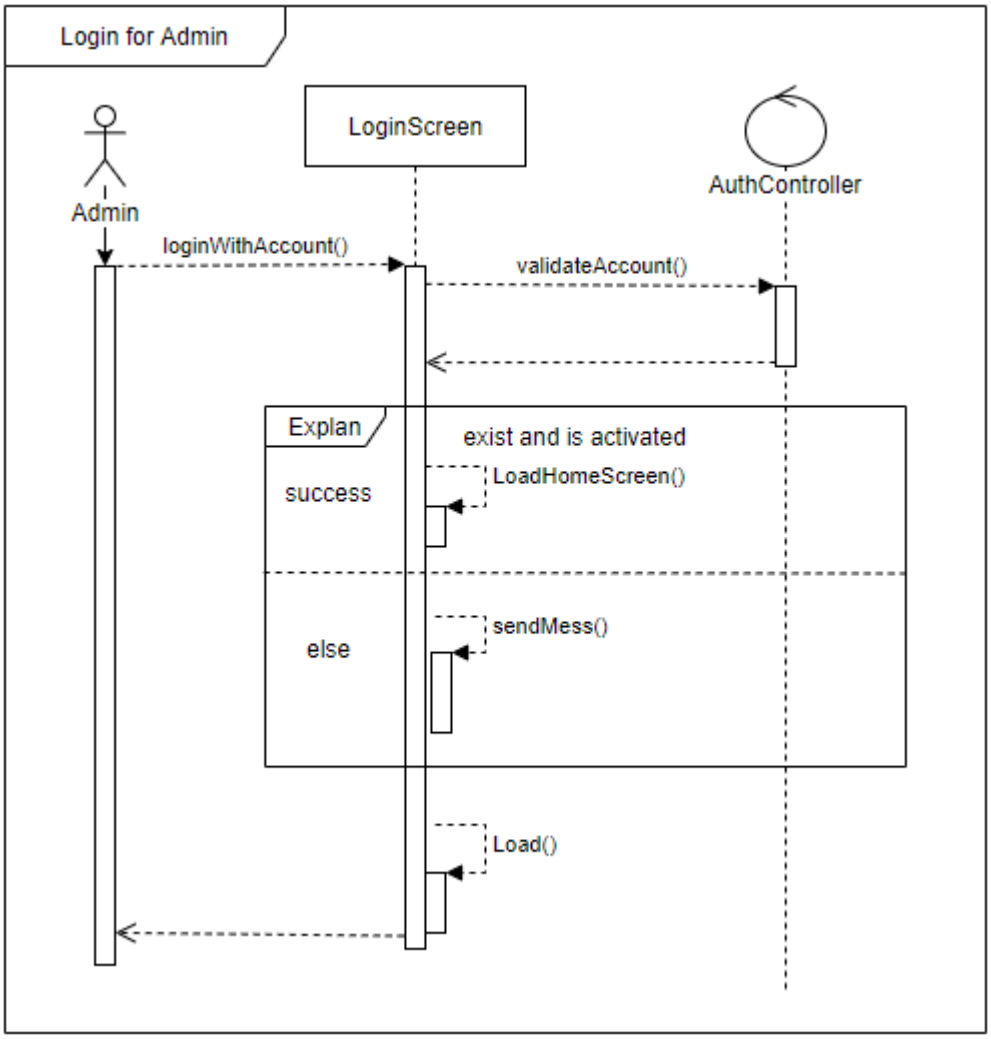


Figure 57 – Login for Admin sequence diagram

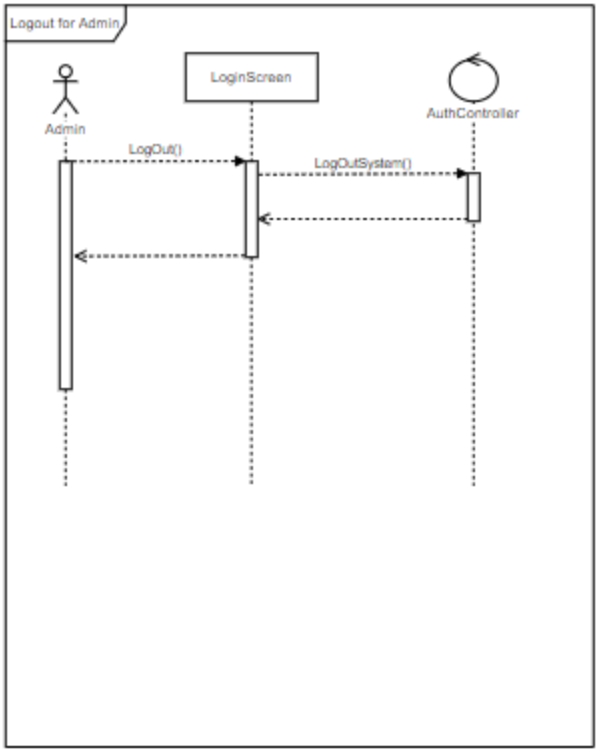


Figure 58 – Logout for Admin sequence diagram

4.5.20. Enable/Disable Account for Admin

*** Enable/Disable account class diagram**

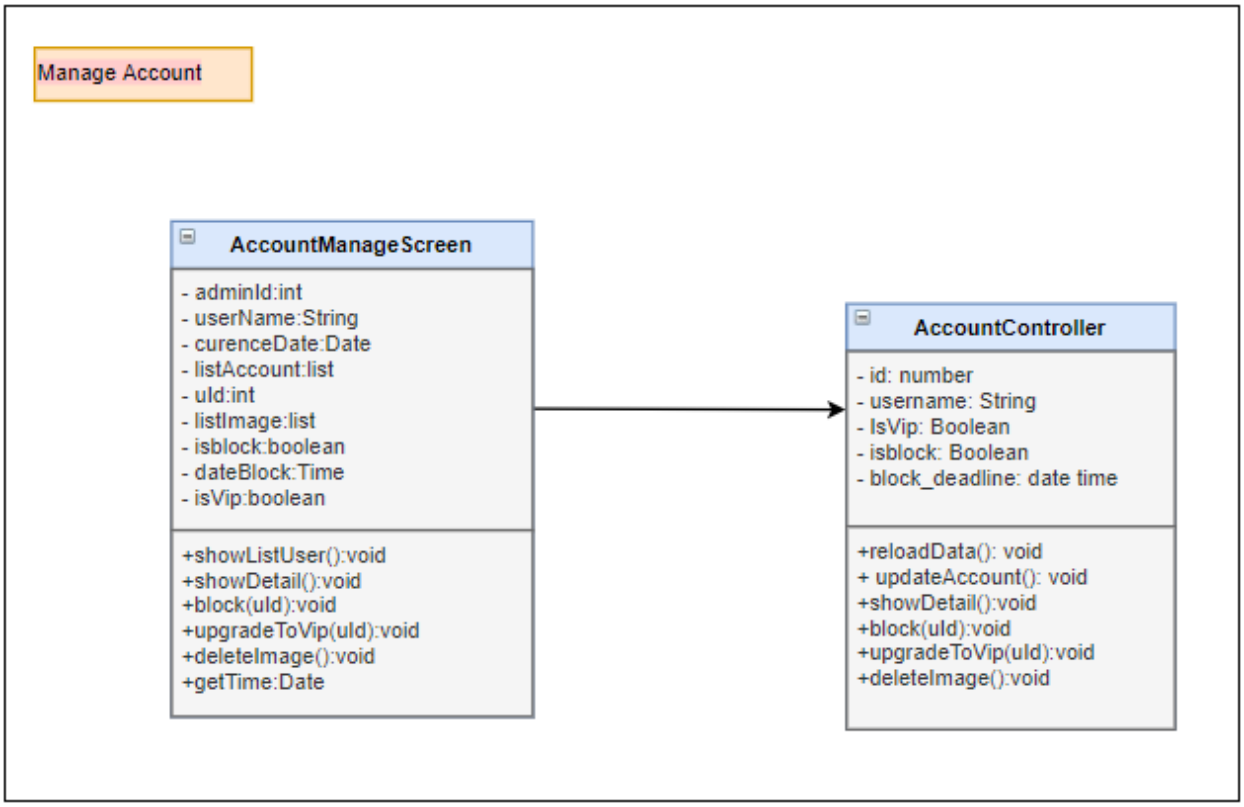


Figure 59 – Enable/Disable Account for Admin class diagram

Name	AccountController	**Type**	Class
Description	Controller control user account		
Attributes			
Name	**Type**	**Visibility**	**Description**
Id	Int	private	Id of user account
userName	String	private	User name of user
isVip	Boolean	private	Vip attribute of user
isBlock	Boolean	private	User blocked or not
block_deadline	date	private	Time block
Operations			
Name	**Type**	**Visibility**	**Description**
reloadData()	void	public	Reload data on screen

updateAccount()	void	public	Update information of account
showDetail()	void	public	Show detail information
Block(uID)	void	public	Block user
upgradeToVip(uID)	void	public	Upgrade user to Vip
deleteImage()	void	public	Delete image

Name	AccountManageScreen	**Type**	Class
Description	Screen manage account		
Attributes			
Name	**Type**	**Visibility**	**Description**
admintId	int	private	Admin id
userName	String	private	Username of user account
curenceDate	Date	private	Currence date
listAccount	list	private	List account of users
uId	int	private	User id
listImage	list	private	List image
isblock	boolean	private	User is blocked
dateBlock	Time	private	Time block
isVip	boolean	private	User is vip
Operations			
Name	**Type**	**Visibility**	**Description**
showListUser()	void	public	Show list account of users
showDetail()	void	public	Show detail informaiton of account
block(uId)	void	public	Block user
upgradeToVip(uId)	void	public	Upgrade user to vip
deleteImage()	void	public	Delete image
getTime()	Date	public	Get currence time

*** Enable/Disable account sequence diagram**

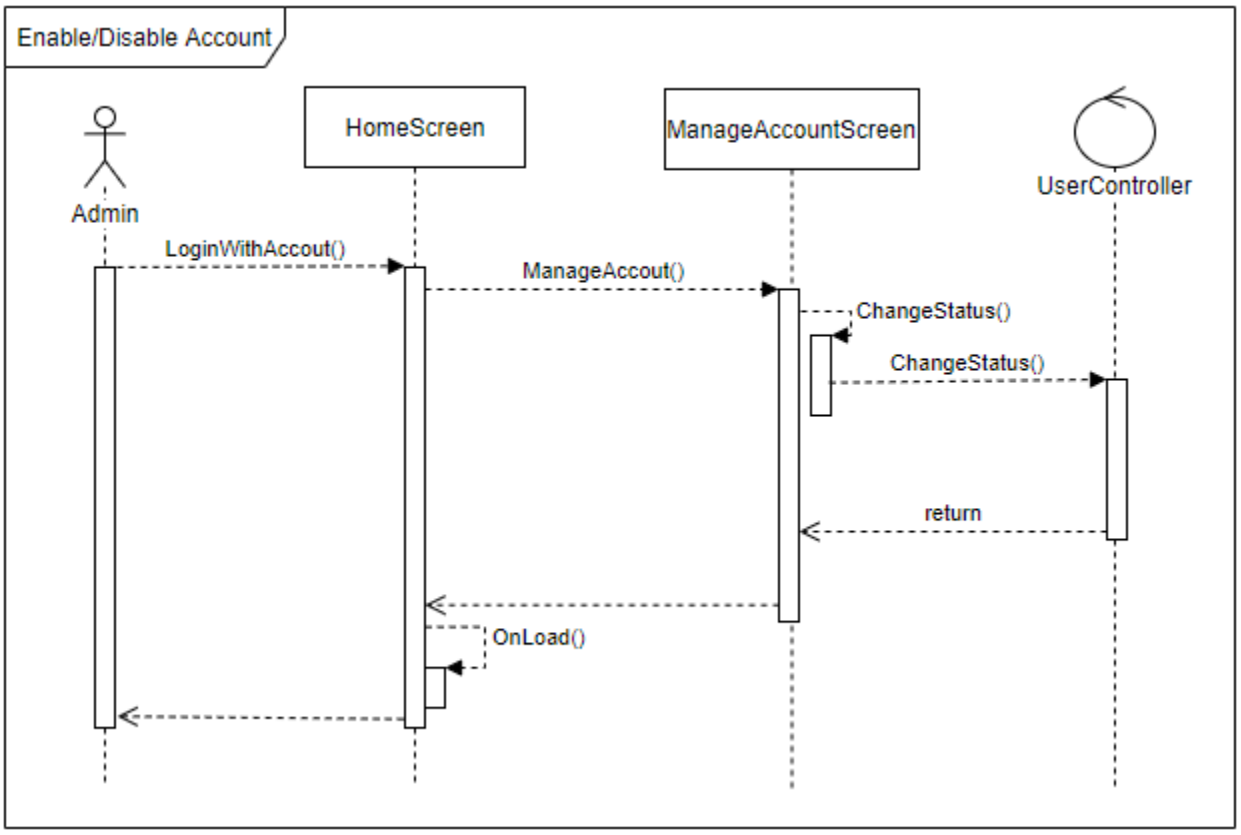


Figure 60 – Enable/Disable Account for Admin sequence diagram

4.5.22. Delete Images for Admin

*** Delete Image class diagram**

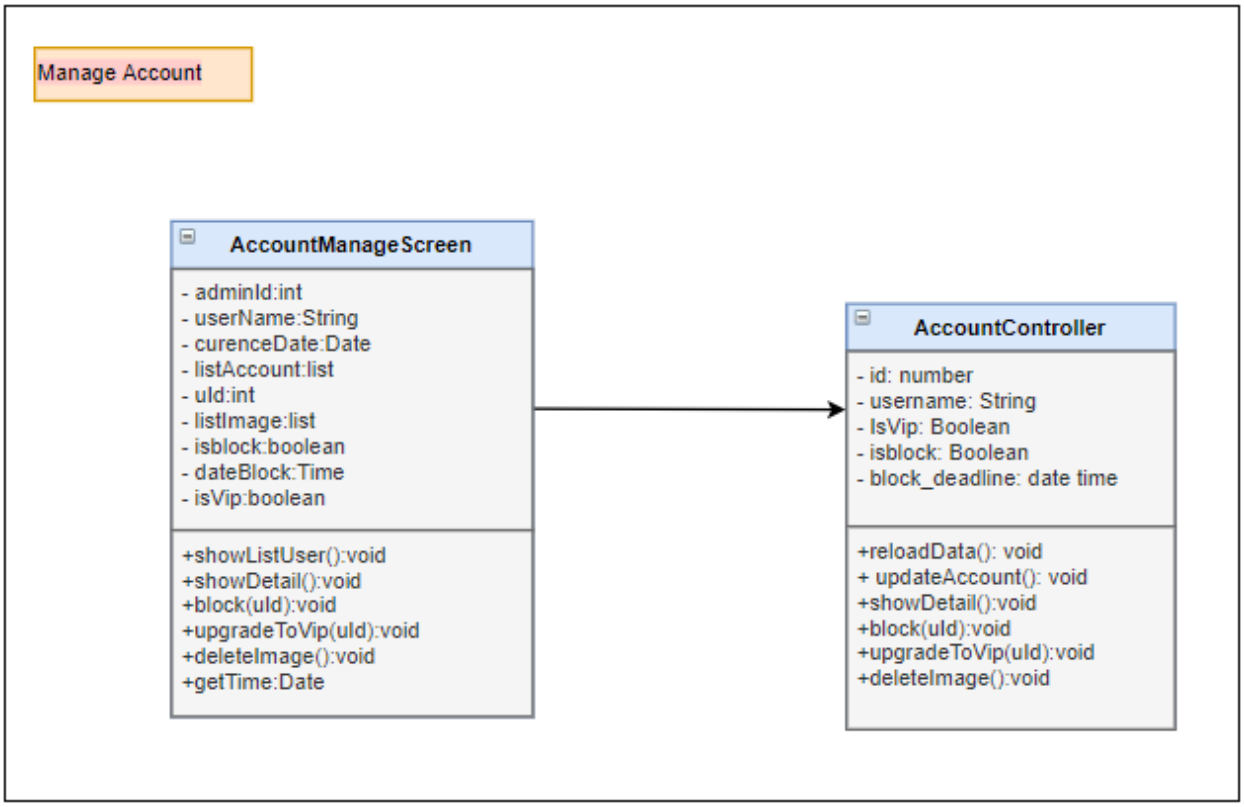


Figure 61 – Delete Images for Admin class diagram

Name	AccountController	**Type**	Class
Description	Controller control user account		
Attributes			
Name	**Type**	**Visibility**	**Description**
Id	Int	private	Id of user account
userName	String	private	User name of user
isVip	Boolean	private	Vip attribute of user
isBlock	Boolean	private	User blocked or not
block_deadline	date	private	Time block
Operations			
Name	**Type**	**Visibility**	**Description**
deleteImage()	void	public	Delete image

Name	AccountManageScreen	**Type**	Class
Description	Screen manage account		
Attributes			
Name	**Type**	**Visibility**	**Description**
admintId	int	private	Admin id
userName	String	private	Username of user account
curenceDate	Date	private	Currence date
listAccount	list	private	List account of users
uId	int	private	User id
listImage	list	private	List image
isblock	boolean	private	User is blocked
dateBlock	Time	private	Time block
isVip	boolean	private	User is vip
Operations			
Name	**Type**	**Visibility**	**Description**
deleteImage()	void	public	Delete image
getTime()	Date	public	Get currence time

*** Delete Image sequence diagram**

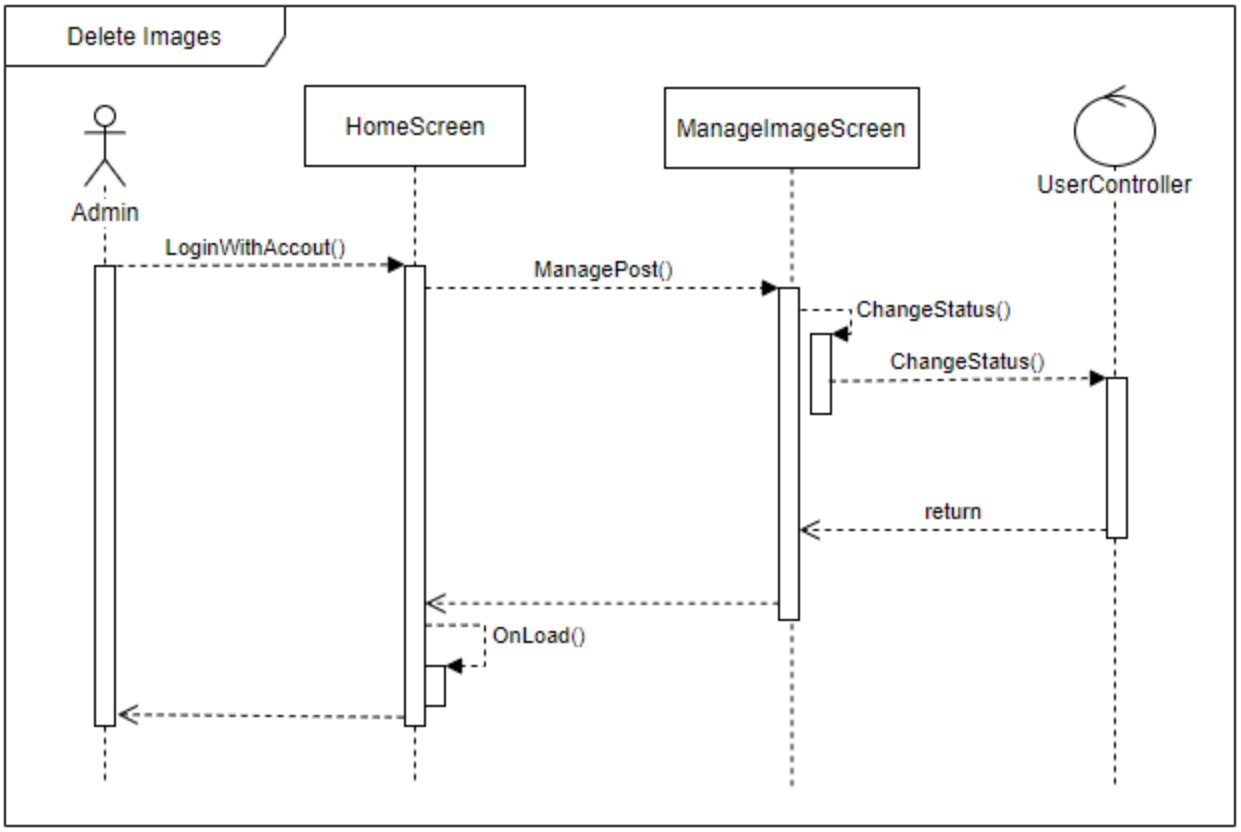


Figure 62 – Delete Images for Admin sequence diagram

4.5.23. Manage system for Admin

*** Manage system class diagram**

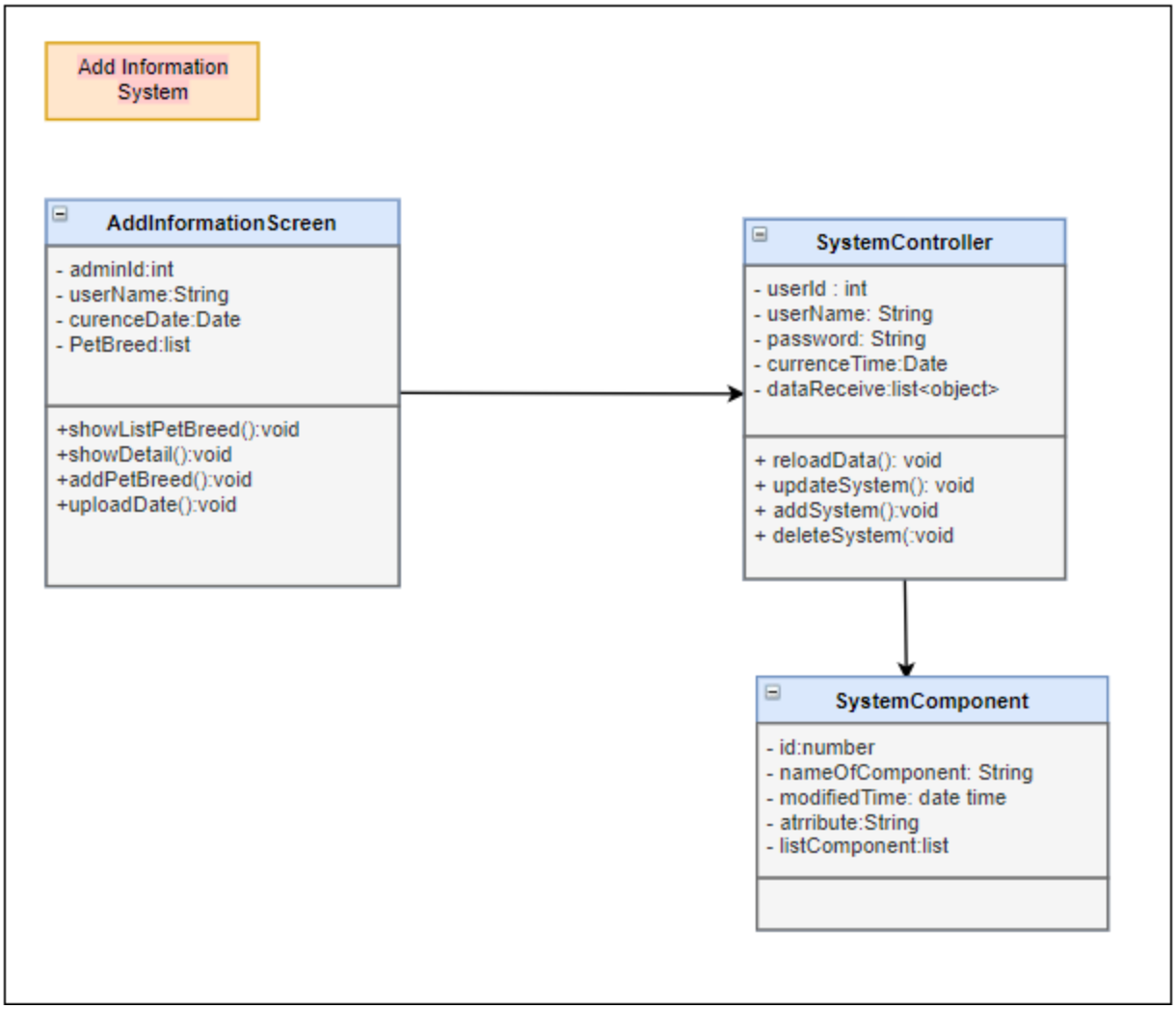


Figure 61 – Manage system for Admin class diagram

Name	SystemController	**Type**	Class
Description	Controller control system information		
Attributes			
Name	**Type**	**Visibility**	**Description**
userId	int	private	admin id
userName	String	private	Admin username
password	String	private	Admin password
currenceTime	Date	private	Currence time
dataReceive:list	list	private	Data receive

Operations			
Name	**Type**	**Visibility**	**Description**
reloadData	void	public	Reload data
updateSystem	void	public	Update information
addSystem	void	public	Add information
deleteSystem	void	public	Delete information

Name	SystemComponent	**Type**	Class
Description	Entity of system		
Attributes			
Name	**Type**	**Visibility**	**Description**
id	int	private	Id of component
nameOfComponent	String	private	Name of component
modifiedTime	date	private	currence time
atrribute:String	String	private	Attribute of component
listComponent	list	private	List component
Operations			
Name	**Type**	**Visibility**	**Description**

Name	AddInformationScreen	**Type**	Class
Description	Add Information Screen		
Attributes			
Name	**Type**	**Visibility**	**Description**
adminId	int	private	Admin id

userName	String	private	Username of admin account
curenceDate	Date	private	Curence date
PetBreed	list	private	List Pet breed
Operations			
Name	**Type**	**Visibility**	**Description**
showListPetBreed	void	public	Show list pet breed
showDetail	void	public	Show detail of pet breed
addPetBreed	void	public	Add pet breed
uploadDate	void	public	Upload date

*** Manage system sequence diagram**

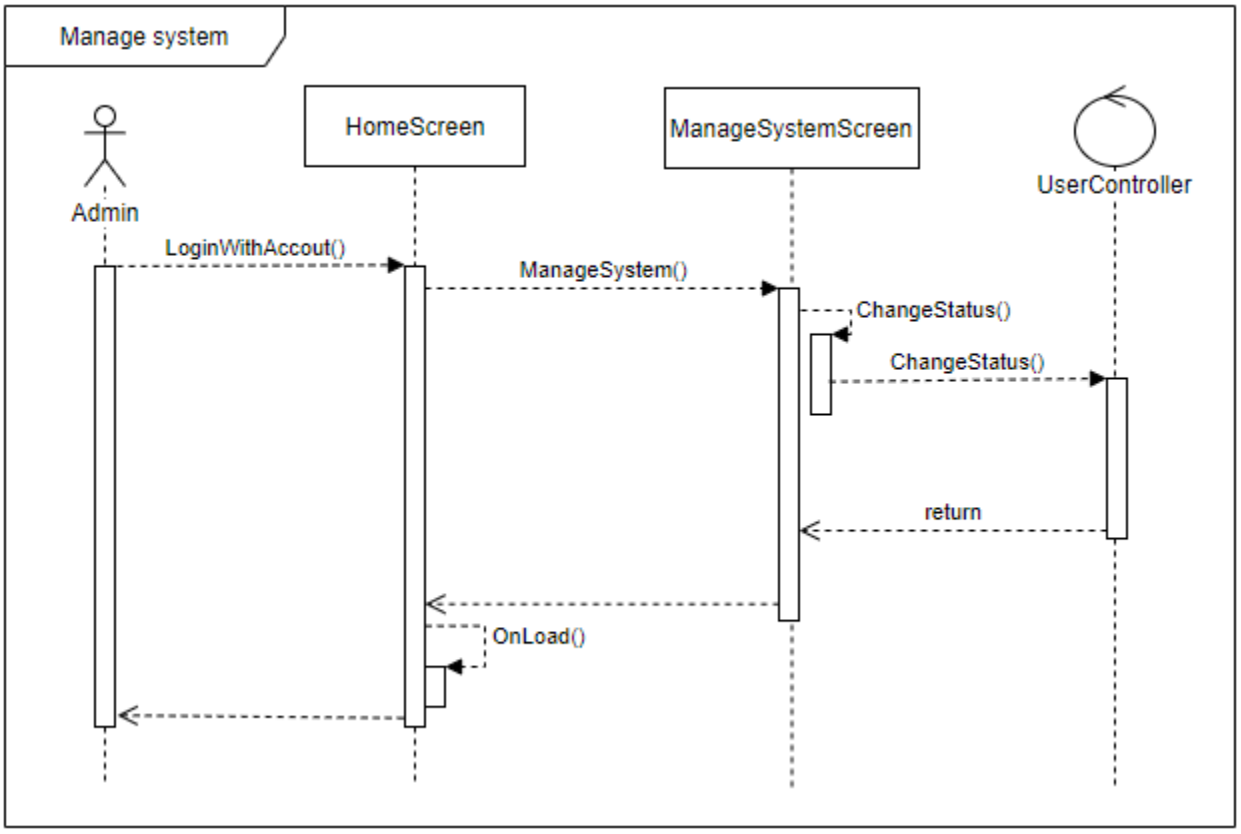


Figure 62 – Manage system for Admin sequence diagram

4.5.24. Receive Feedback for Admin

*** Receive Feedback class diagram**

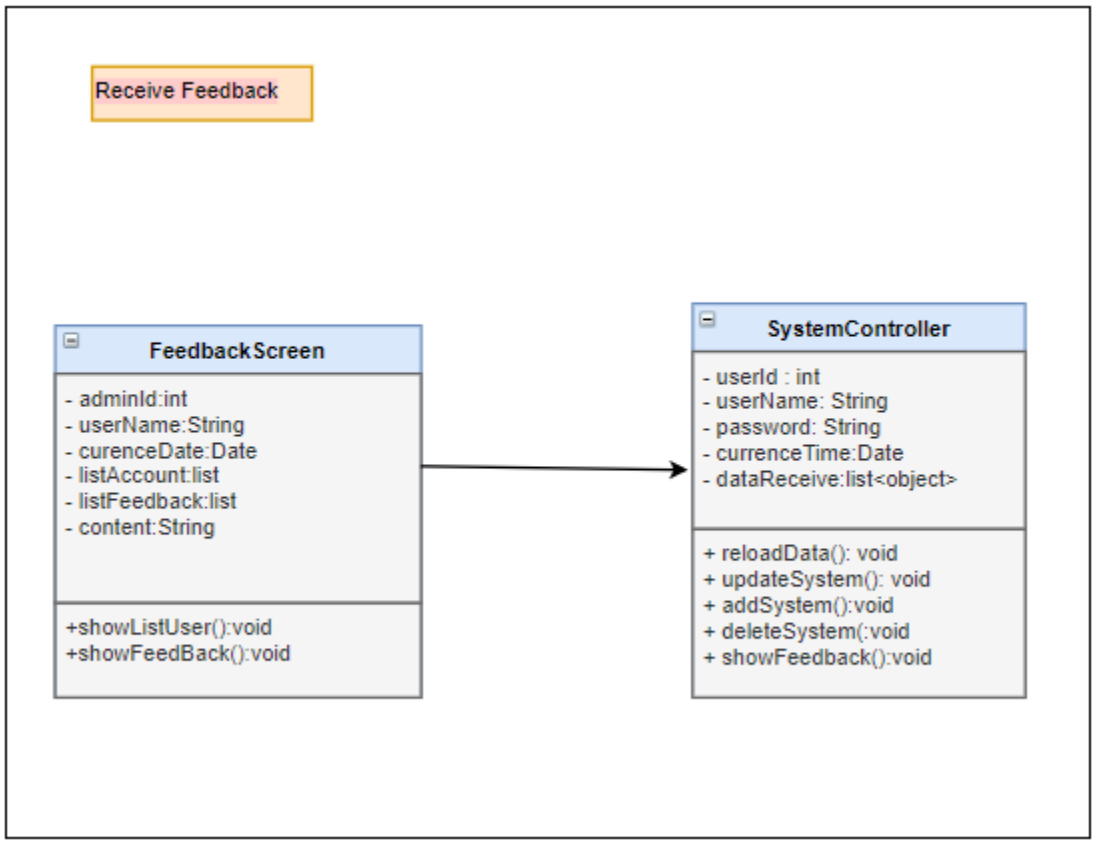


Figure 63 – Receive Feedback class diagram

Name	SystemController	**Type**	Class
Description	Controller control system information		
Attributes			
Name	**Type**	**Visibility**	**Description**
userId	int	private	admin id
userName	String	private	Admin username
password	String	private	Admin password
currenceTime	Date	private	Currence time

dataReceive:list	list	private	Data receive
Operations			
Name	**Type**	**Visibility**	**Description**
reloadData	void	public	Reload data
updateSystem	void	public	Update information
addSystem	void	public	Add information
deleteSystem	void	public	Delete information

Name	FeedbackScreen	**Type**	Class
Description	Show all of feedback from user		
Attributes			
Name	**Type**	**Visibility**	**Description**
adminId	int	private	Id of admin
userName	String	private	User name of admin account
curenceDate	Date	private	Currence date
listAccount	List	private	List all of account write feedback
listFeedback	List	private	List all of feedback
content	String	private	Content of each feedback
Operations			
Name	**Type**	**Visibility**	**Description**
showListUser()	Void	Public	Show list user feedback
showFeedback()	void	public	Show list feedback

*** Receive Feedback sequence diagram**

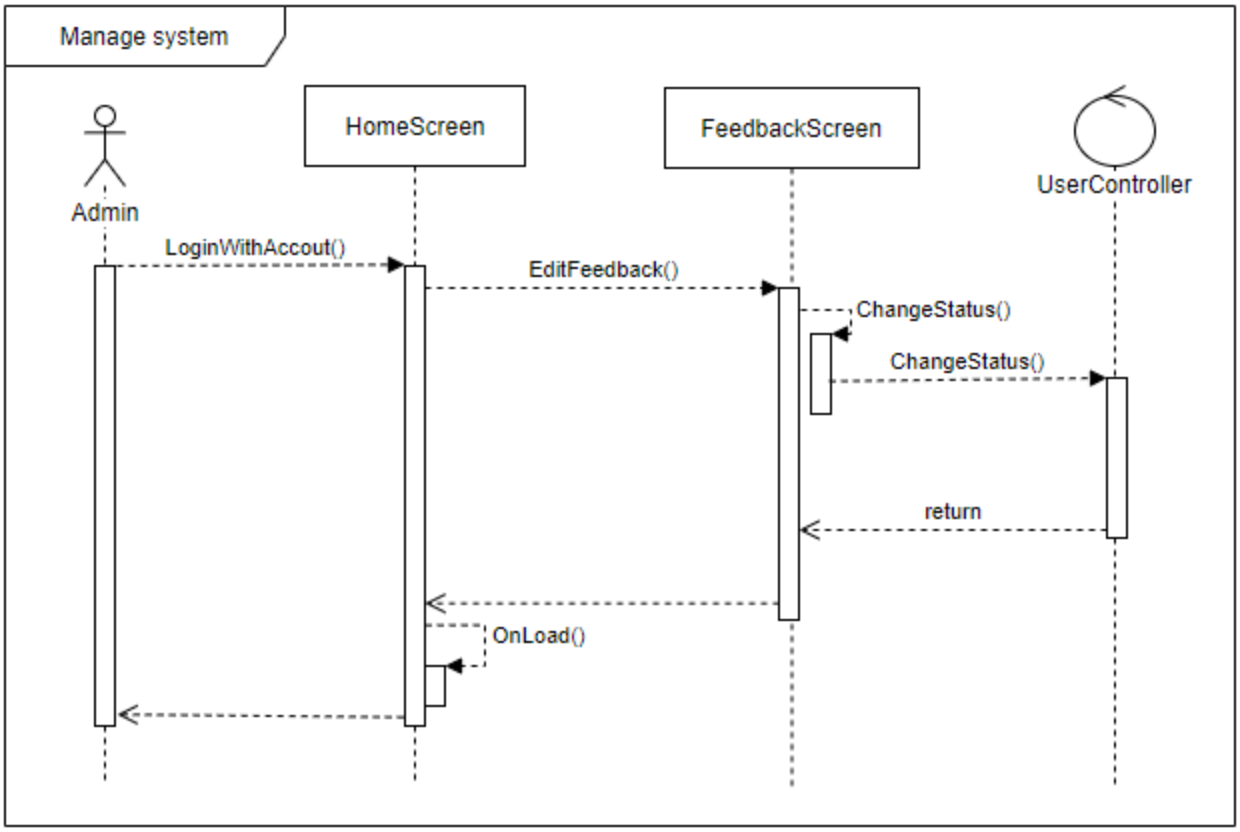


Figure 64 – Receive Feedback for Admin sequence diagram

CHAPTER 5: Software Testing Document

5.1. Introduction

5.1.1. Purpose

The primary purpose of this chapter is the process to check whether the software is defect-free or not. It is the process of verification and validation of software service or application by checking whether it is meeting the user requirements and what all is implemented as per the characteristics. Software testing plays a vital role in the process of developing a high quality software. Testing is necessary because we all make mistakes. Some of those mistakes are unimportant but some of them are expensive and dangerous. It contains the following sections:

- Scope of Testing
- Testing Tool and Environment
- Resources and responsibilities

- Test strategy: Test approach, test stage
- Test schedule
- Feature to be tested
- Feature not to be tested
- Defect Log
- Test report

5.1.2. Scope of testing

- There are four main stages of testing that need to be completed before a program can be cleared for use: unit testing, integration testing, system testing, and acceptance testing.

ID	Test Stages	Description
1	Unit testing	During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as designed. In this phase, a unit can refer to a function, individual program or even a procedure, and a White-box Testing method is usually used to get the job done. One of the biggest benefits of this testing phase is that it can be run every time a piece of code is changed, allowing issues to be resolved as quickly as possible. It's quite common for software developers to perform unit tests before delivering software to testers for formal testing.
2	Integration testing	Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to find

		interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they aren't properly integrated, it will affect the functionality of the software program. In order to run these types of tests, individuals can make use of various testing methods, but the specific method that will be used to get the job done will depend greatly on the way in which the units are defined.
3	System testing	System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven't played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.
4	Acceptance testing	The final level, Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business' needs. Once this process has been completed and the software has

		passed, the program will then be delivered to production.

- Type of testing

  The following type of testing is used in this project are:

  - GUI test
  - Function test
  - Regression test
  - Acceptance test include alpha test
  - Unit test

5.1.3. Range of testing

Team performs all functions defined in the SRS based on the approved version.

5.2. Test plan

5.2.1. Testing Tools and Environment

5.2.1.1. Testing tools

- **Front-end and Project Testing**
  - Chrome Developer Tool: To view logs.
  - Trello: Manage bug.
  - Microsoft Excel, Git: Manage testcase
- **API testing**
  - PostMan: to manage the list of all APIs and manually test APIs' result
- **Unit test**
  - Jest: Unit test for JavaScripts
- **UI testing**
  - React-Native-CLI client: View app when development

5.2.1.2. Testing environment

- **Device 1: Laptop Dell**
  - CPU: i7 7300hq

    - Ram: 8gb
    - Gpu: gtx1050ti
- **Device 2: Laptop HP**
    - CPU: i7 10510U
    - Ram: 8gb
    - Gpu: Geforce MX250
- **Device 3: Laptop Asus**
    - CPU: i7 9750H
    - Ram: 8gb
    - Gpu: GTX1650
- **Other device: Mobile phone**
    - OPPO F7
    - Samsung Galaxy A21s
    - Samsung Galaxy A80
    - OPPO A52

5.2.2. Resources and Responsibilities

ID	Resources	Responsibilities
1	Project Manager	- Responsible for project schedule and overall success of the project - Review test-case and report
2	Tester	- Preforming the actual system testing - Manage test resource and assign test tasks - Create test plan - Create test cases - Create test report - Execute test - Test log report
3	Developer	- Create unit test and integration test scripts - Fix bugs

5.2.3. Test Strategy

5.2.3.1. Test Model

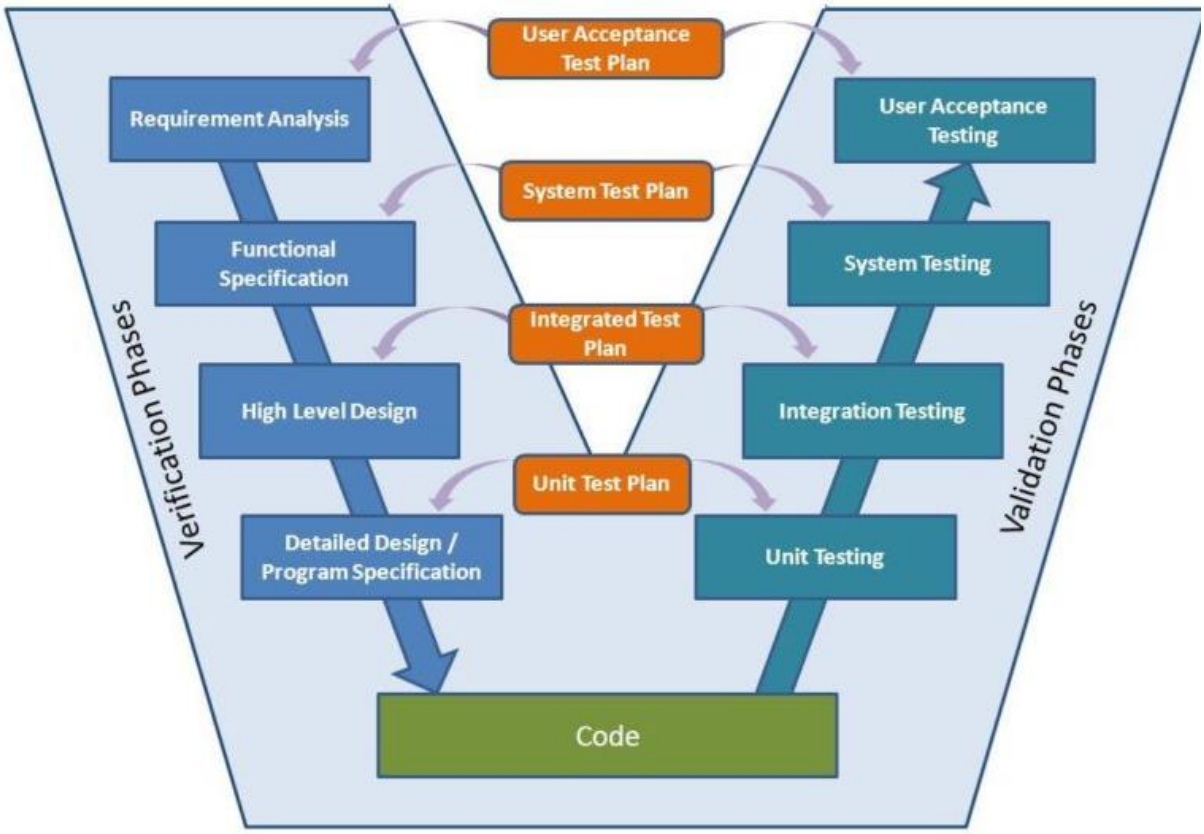


Figure 65 - V-model

Overall, we choose V-Model to implement testing process. With V-Model, software development is separated into two appropriate phase's groups: development and testing. In this model, the verification and validation will be done side by side. It emphasizes the strict process flow to develop a quality product. The errors occurred in any phase will be corrected in that phase. Proactive defect tracking defects, which are found at early stages even, may be in the development phase before application being tested.

Pet Dating API has 2 levels of test:

- Unit testing: Automation tests that cover logic of Models and Libraries
- API testing: Automation tests that involve testing APIs directly (in isolation) to determine whether APIs return the correct response (in the expected format) for a broad range of feasible requests, react properly to edge cases such as failures and unexpected/extreme inputs.

- Pet Dating Front-end works mostly with GUI instead of logic and it depends on Pet Dating API, so that Pet Dating Front-end apply system testing which coverage of whole Pet Dating system.

5.2.3.2. Test types

The following type of testing is used in this project are:

- Unit test:

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class.

- Unit test also includes database testing to verify constraint, transaction, default value, data types, data format, and check null and junk characters which are mentioned in database design and software requirement.
- Test case will have to cover all logic branch that function or method could execute with difference data input. Another alternative logic branch should be covered if not, that logic branch should be detected at API testing level.
- Implemented function's error message and database error message will be included in this test.

- API test:

API TESTING is a software testing type that validates Application Programming Interfaces (APIs). The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs (keyboard) and outputs, you use software to send calls to the API, get output, and note down the system's response. API tests are very different from GUI Tests and won't concentrate on the look and feel of an application. It mainly concentrates on the business logic layer of the software architecture.

- Involves testing APIs directly to determine if they meet expectations for functionality, reliability, performance, and security. API testing will test all of individual implemented API of PetDating API.
- Test case will verify constraint of data which be mention in Business rule
- Basically, almost all API test cases are executed as automation test. After that all API with standard sample datasets will be saved and confirmation tests will be executed by using Postman with developer's local database.

- UI test:

Graphical User Interface testing is mainly about ensuring that the UI functions in the right, an app follows its written specifications and that defects are identified. Other than that, we check that the design elements are good. This involves checking the screens with the controls like menu bars, toolbars, color, fonts, sizes, icons, content, buttons, etc. How do they respond to user input? GUI test will be performed fully on all screens.

- This test targets to cover the verification of the overall look and feel of the OCFU system including initial position, font, text size, color, focus, initial button, tab order, label, screen sizes and sentences width.
- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, it must be able to provide inputs to the input fields.
  - Check if Error Messages are displayed correctly.
  - Check if Font used in application is readable.
  - Check if the alignment of the text is proper.
  - Check if the Color of the font and warning messages is aesthetically pleasing.
  - Check if images have good clarity.
  - Check if images are properly aligned.
  - Check the positioning of GUI elements for different screen resolution.

- Regression testing

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the

product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

- Acceptance testing
    - This test type will be executed by tester with designed test cases, acceptance test is a test type conducted to determine if the requirements of a specification or contract are met
    - It also includes alpha testing; alpha testing takes place at close relation user's site and are free test to detect bug and strange behavior. By that, development team will improve UX and UI of system

5.2.3.3. Test stage

The table below describes the stages in which common tests are executed:

Type of test	Stage of test			
	Unit test	Integration test	System test	Acceptance test
Unit tests	x	x	x	
API tests	x	x		
UI tests			x	x
Regression tests	x	x	x	x
Acceptance tests			x	x

5.2.3.4. Test schedule

Task		**Start date**	**End date**
Phase 1	Create test plan	10/06/2010	11/06/2020
	Create and execute unit test	12/06/2020	28/06/2020
	Create and execute integration test	12/06/2020	28/06/2020
Phase 2	Create test plan	30/06/2020	01/07/2020
	Create and execute unit test	01/07/2020	28/07/2020

	Create and execute integration test	03/07/2020	30/07/2020
Phase 3	Create test plan	20/07/2020	21/07/2020
	Create and execute system test	25/7/2020	22/08/2020
	Create and execute acceptance test	28/07/2020	15/08/2020

5.2.3.5. Deliverables

No	Deliverables	Responsibilities	Delivered date
1	Test plan	Test Leader	21/07/2020
2	Unit test	Developer	28/07/2020
3	Integration test	Tester + Developer	30/07/2020
4	System test	Tester + Developer + PM	12/08/2020
5	Acceptance test	All member	15/08/2020
6	Defect logs	All member	18/08/2020
7	Final test summary report	Test Leader	20/08/2020

5.2.4. Features to be tested

- All features that are listed in Requirement that application have.
- GUI of the Mobile application.

5.3. Test case

5.3.1. Unit test

Unit tests are done by the developers in order to test individual units, and approved by the test leader.

This is done to ensure that important functions, such as algorithms for verb conjugation and name conversion, return the correct values.

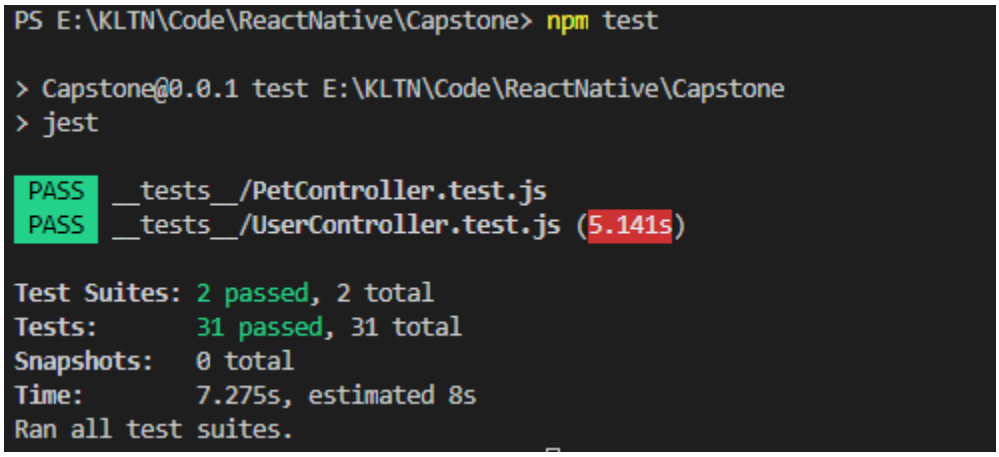


Figure 66 - Unit test with jest

5.3.2. System testing

Detail test cases will be described in **"Petdating_Test Case_ver01.xls"** file.

5.3.3. Acceptance test

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. But our project will use the Check Lists as a substitute for Acceptance testing.

The content of the Checklist is shown in the table below

No	Checklist	Yes	No
	General		
1	Text does not have grammatical and spelling errors.	√	
2	All buttons are functional.	√	
3	All mandatory fields are validated and indicated by asterisk (*) symbol.	√	
4	All error messages are displayed using red color.	√	
5	All "Delete" functions ask for confirmation.	√	
	GUI and usability		
6	Screen design follows the standard of the project.	√	
7	Waiting icon appear for waiting display data.	√	
8	Color red is used for all error messages.	√	

9	Project state is displayed and colored with appropriate state	√	
10	Design style is friendly and easy to use	√	
11	The text easy to understand. Don't use slang, acronyms, and abbreviations.	√	
12	The static text is clear, concise, and meaningful.	√	
13	The screen designed to fit with multiple screen size.	√	

5.3.4. Defect Log

Excel and GitHub are used to manage bugs:

- When a bug is found, testers change the status and assign for developer.
- Developers fix bugs and change status of bugs to resolve.
- If the bug is fixed then tester change the status to close, otherwise will be re-open again to fix.

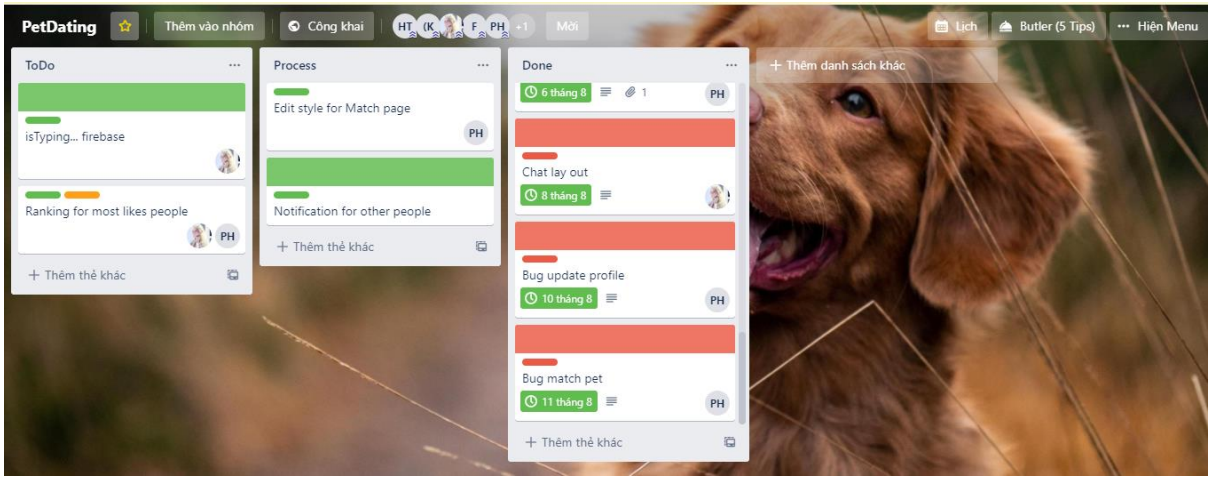


Figure 674 - Defect logs with Trello

5.4. Test Report

5.4.1. Unit test

	Function Name	Number of test case (Estimate)	Pass	Fail
1	validateUser	8	8	0
2	convertToAge	11	11	0

3	ValidatePet	12	12	0
Total		31	31	0

5.4.2. Integration and system tests

TEST REPORT

Project Name	*Pet Dating*	**Creator**		HieuKT, KhuongNDA
Project Code	*PD*	**Reviewer/Approver**		ThinhND, HieuKT
Document Code	*PD_Test Report_v1.0*	**Issue Date**		*20/8/2020*
Notes				

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	PD_Login screen	7	0	0	0	7
2	PD_Home screen	34	0	0	0	34
3	PD_Drawer screen	11	0	0	0	11
4	PD_Dashboard screen	26	0	0	0	26
5	PD_Profile screen	72	0	0	0	72
6	PD_Filter screen	9	0	0	0	9
7	PD_Settings screen	13	0	0	0	13
8	PD_Ranking screen	6	0	0	0	6
9	PD_Admin screen	89	0	0	0	89
10	PD_API	38	0	0	0	38
11	PD_Premium screen	10	0	0	0	10
	Sub total	315	0	0	0	315

Test coverage **100,00 %**
Test successful coverage **100,00 %**

Figure 68 - Integration and system test report

CHAPTER 6: USER MANUAL

6.1. Development and deployment guidelines

6.1.1. Development guideline

6.1.1.1. Environment for development

☞ Laptop Dell Inprison

➢ Integrated development environment:

• Visual Studio Code version 1.48.2

➢ Node.js version 12.8.1

➢ XAMPP 7.3.12

➢ React-native-cli version 2.0.1

☞ Android Phone with Android 7.0

6.1.1.2. Setup development environment

6.1.1.2.1. Install Visual Studio Code

• Go to URL: https://code.visualstudio.com/, click on “Download” button, the Download page appears, choose the Windows package (32 bit or 64 bit, depends on the system), the Save pop-up appears, click Save button to start download the installation file

Double click the file **“VSCodeUserSetup-x64-1.41.0.exe”** to start install Visual Studio Code

Figure 5 - Interface of Visual Studio Code

• Start using Visual Studio Code by opening it from Desktop

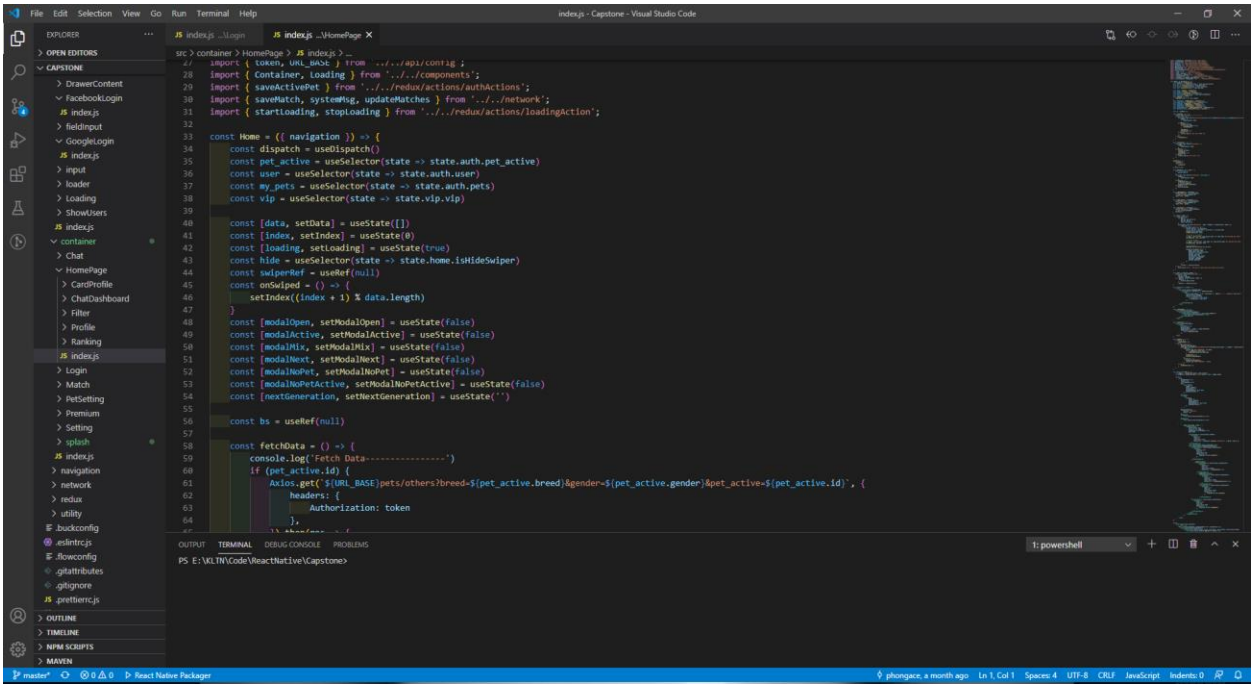


Figure 6 - Interface of Visual Studio Code

6.1.1.2.2. Install Node.js version 12.8.1

• Go to URL:https://nodejs.org/en/download/releases/, choose the 12.8.1 version and click Download, the Index page of the corresponding version appears, choose the node-v12.8.1-win-

x64.zip hyperlink, the Save pop-up appears, click Save button to start download the installation folder.

Figure 7 - Nodejs setup folder

Extract the folder **"node-v12.8.1-win-x64.zip"**

• Open the folder, double click the file **"node.exe"** to start install Nodejs.

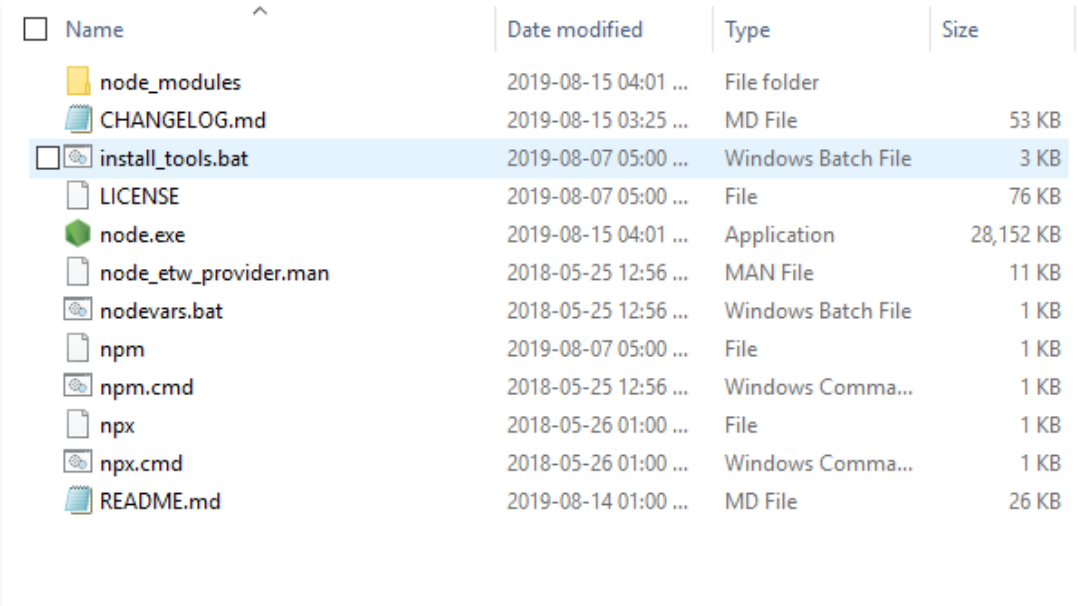


Figure 8 - Nodejs setup folder after extracting

• After the setup file running completely, check if Nodejs was installed successfully by opening Command Prompt and running the command "node -v"

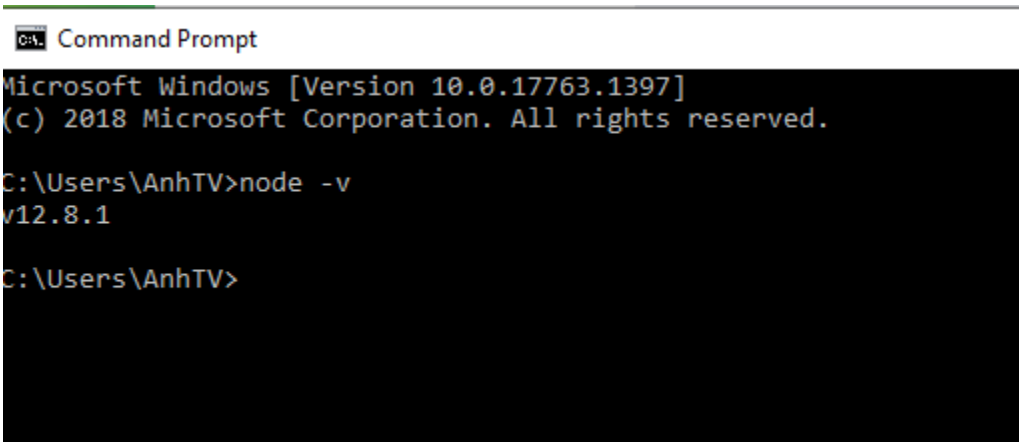


Figure 9 - Nodejs was installed succesfully

6.1.1.2.3. Install react-native-cli version 2.0.1

• Open Command Prompt and running the command "*npm install -g react-native-cli*"

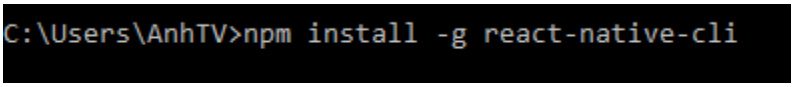


Figure 10 - Command for installing expo-cli

• After the command running succesfully, check if expo-cli was installed successfully by running the command "*react-native -v*"

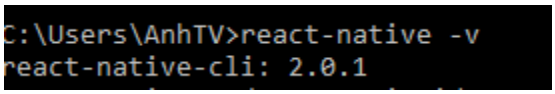


Figure 11 - expo-cli was installed successfully

6.1.2. Deployment guideline

6.1.2.1. Environment for deployment

☞ Heroku.com

☞ Google Firebase with following services:

➢ Authentication

➢ Realtime Database

6.1.2.2. Setup deployment

6.1.2.2.1. Setup hosting on Heroku.com

• Go to https://heroku.com to create an account

- Go to https://heroku.com to create app

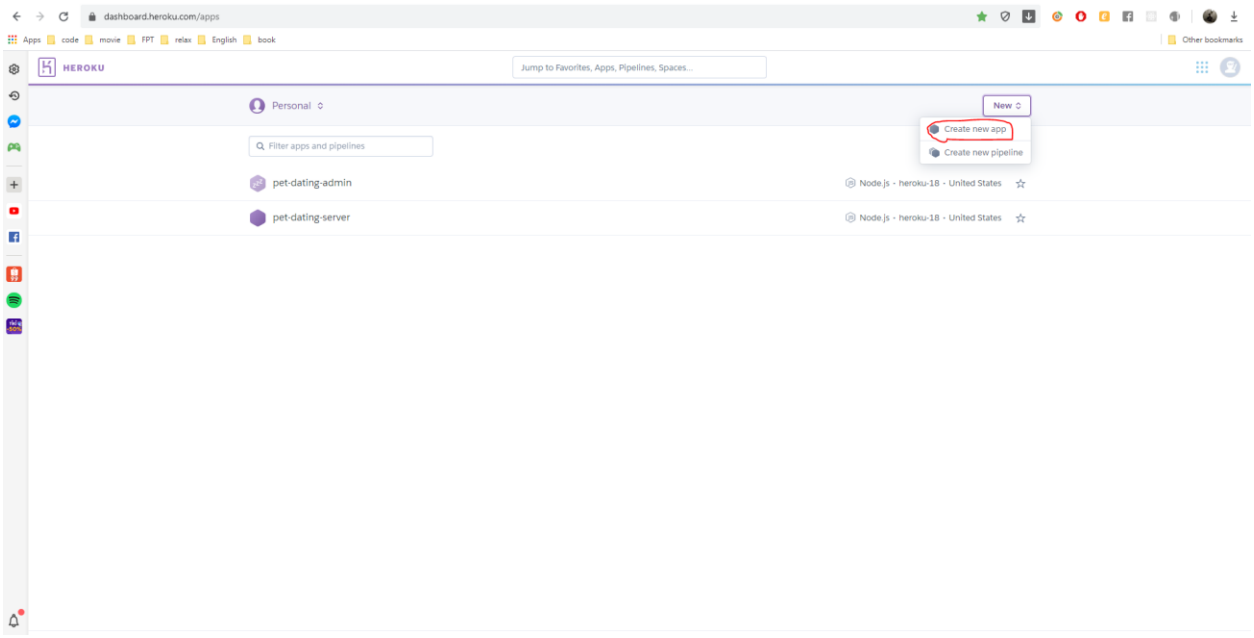


Figure 12 - Create application interface on Heroku.com

- Enter app name and press Create app button

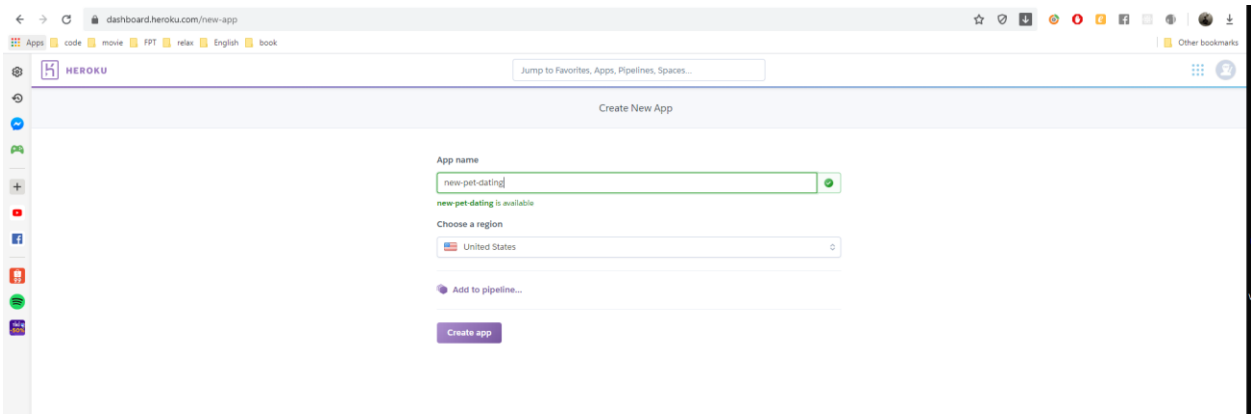


Figure 13 – create app interface

- Enter app name and press Create app button and do follow guideline to setup Heroku environment

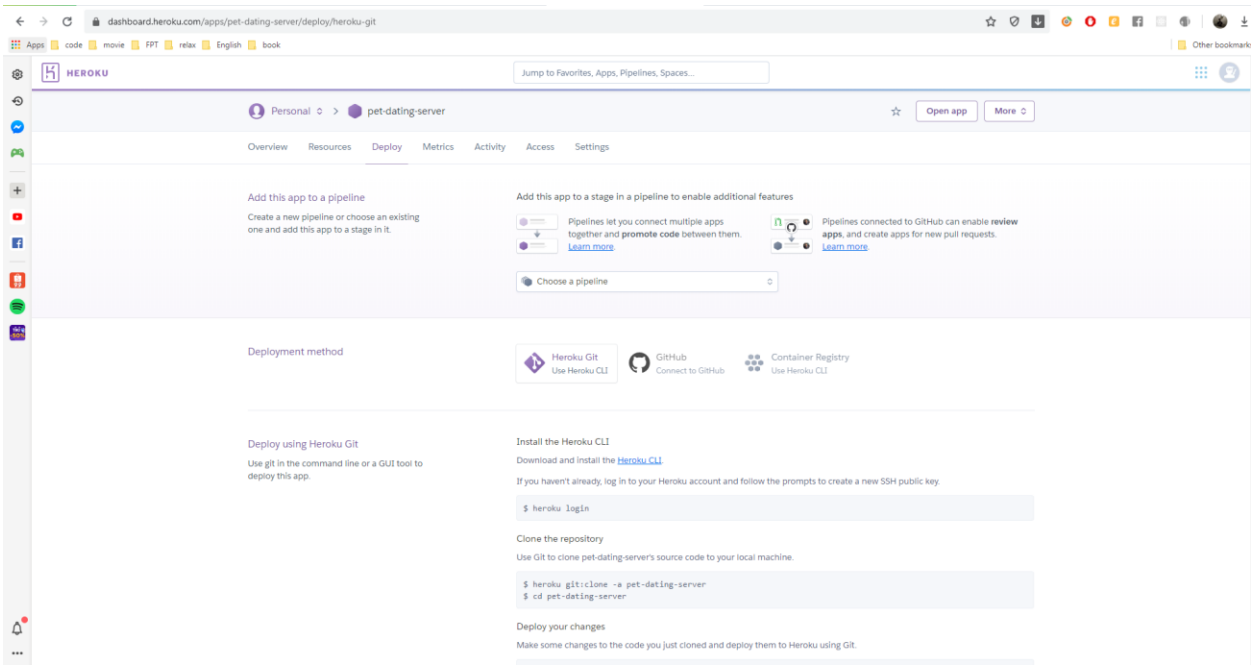


Figure 111 – deploy interface

- Go to folder contain source code, open command line and enter ‘git push heroku master’

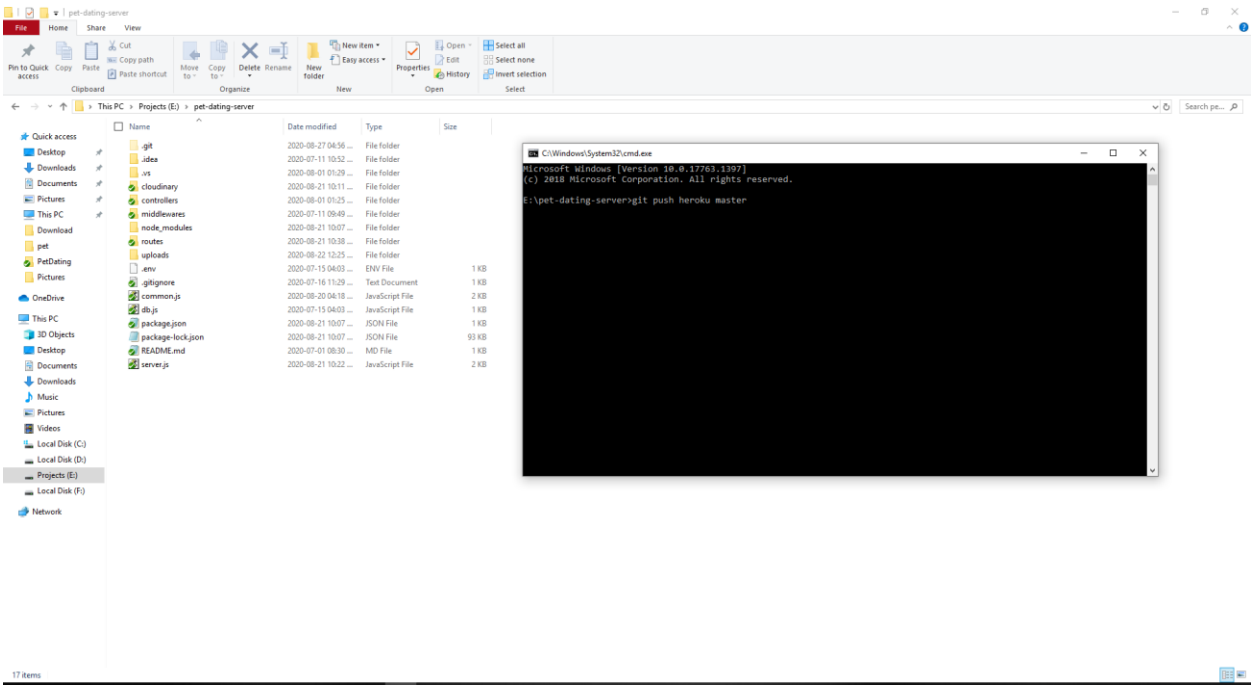


Figure 112 – deploy command line

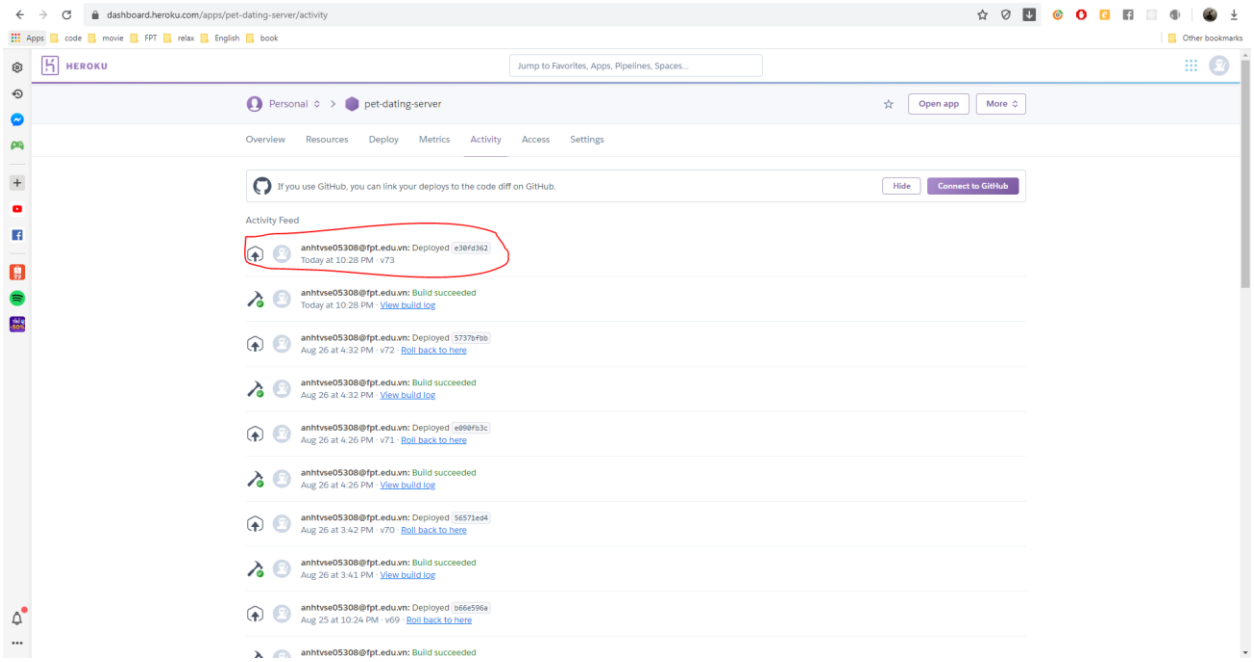


Figure 113 – result interface

6.1.2.2.2. Setup services of Google Firebase

- Go to https://console.firebase.google.com/u/0/ and signing in with Google account
- Click Add project to create a project with Firebase console

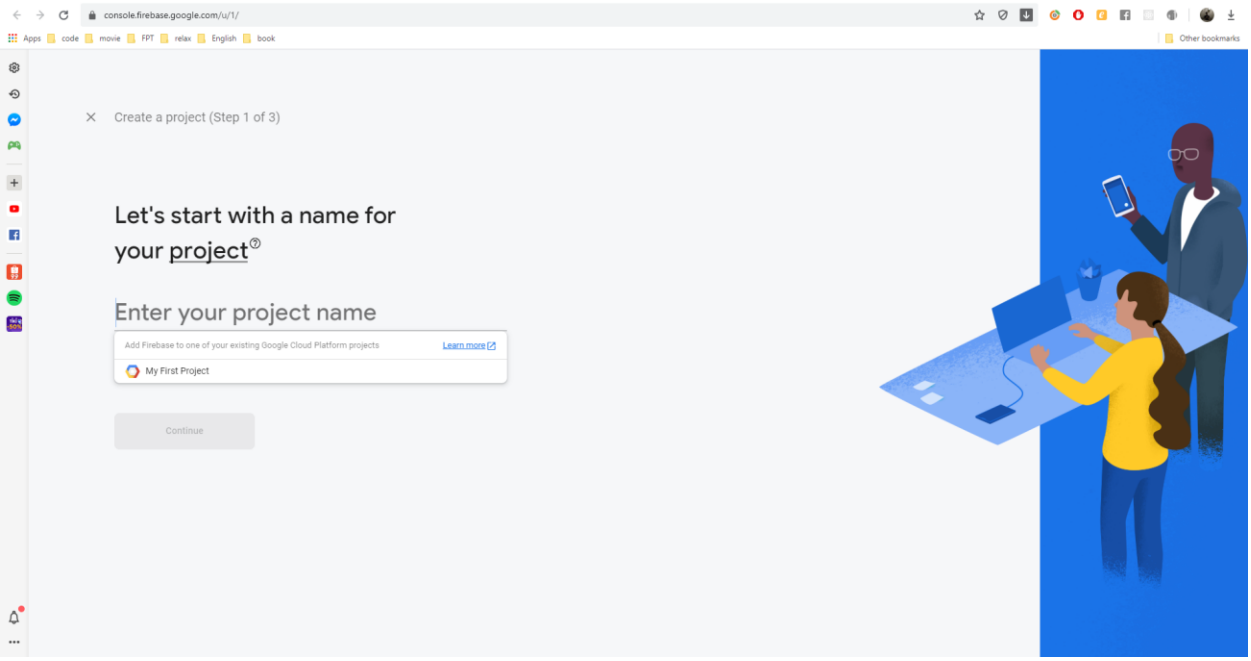


Figure 114 - Create project interface

- From the sidebar menu, click Authentication
- Switch to Sign-in method tab, in the Sign-in providers section, switch Disable to Enable on Google and Facebook

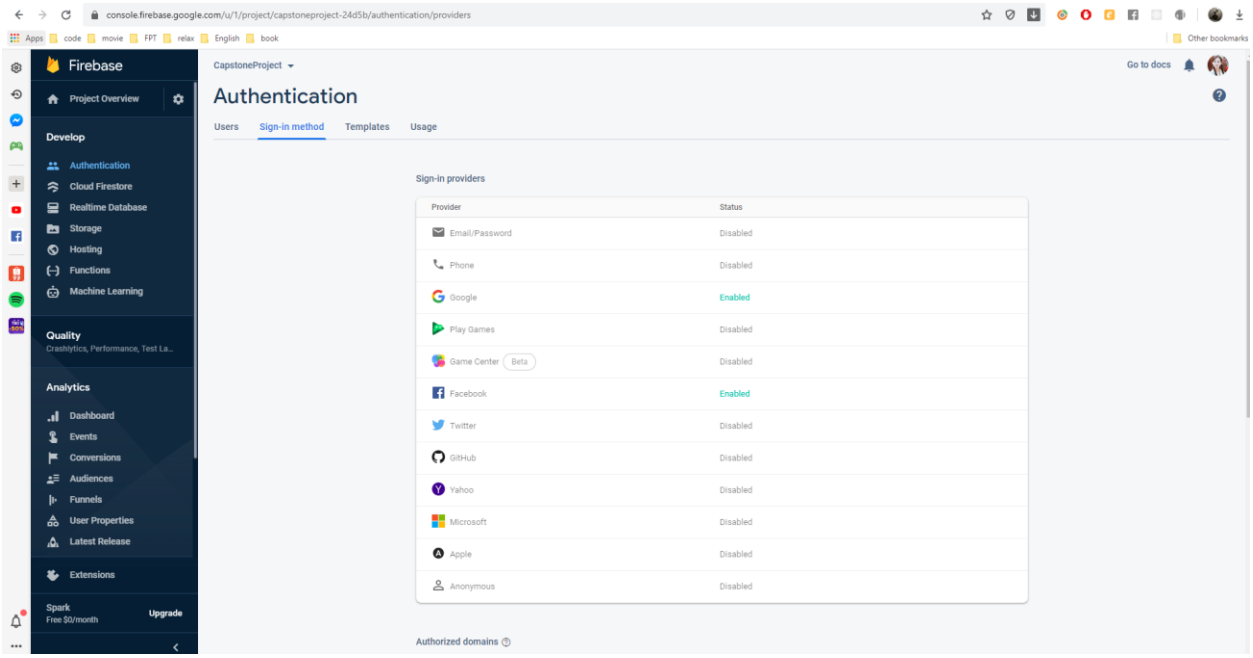


Figure 14 - Sign-in method set up interface for database

- From the sidebar menu, click Database
- Go to Realtime Database section and click Create database button
- Choose option “Start in locked mode”, then click Enable button

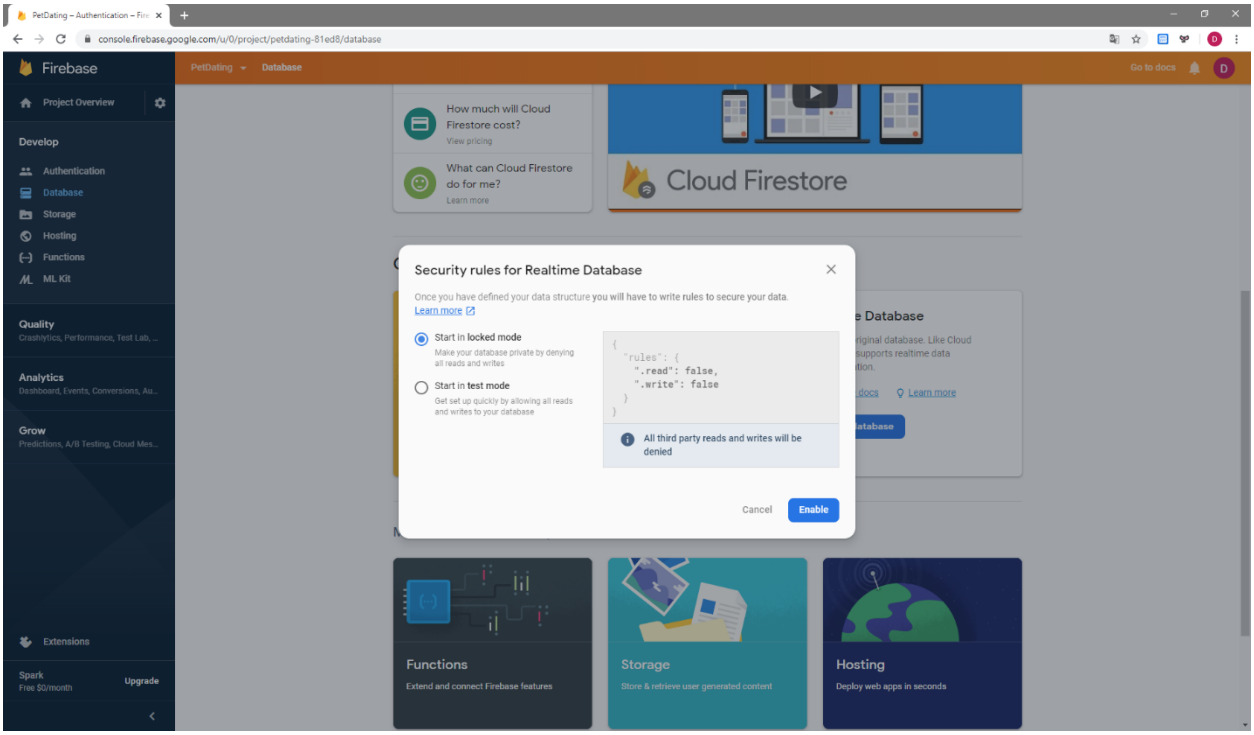


Figure 116 - Create database interface

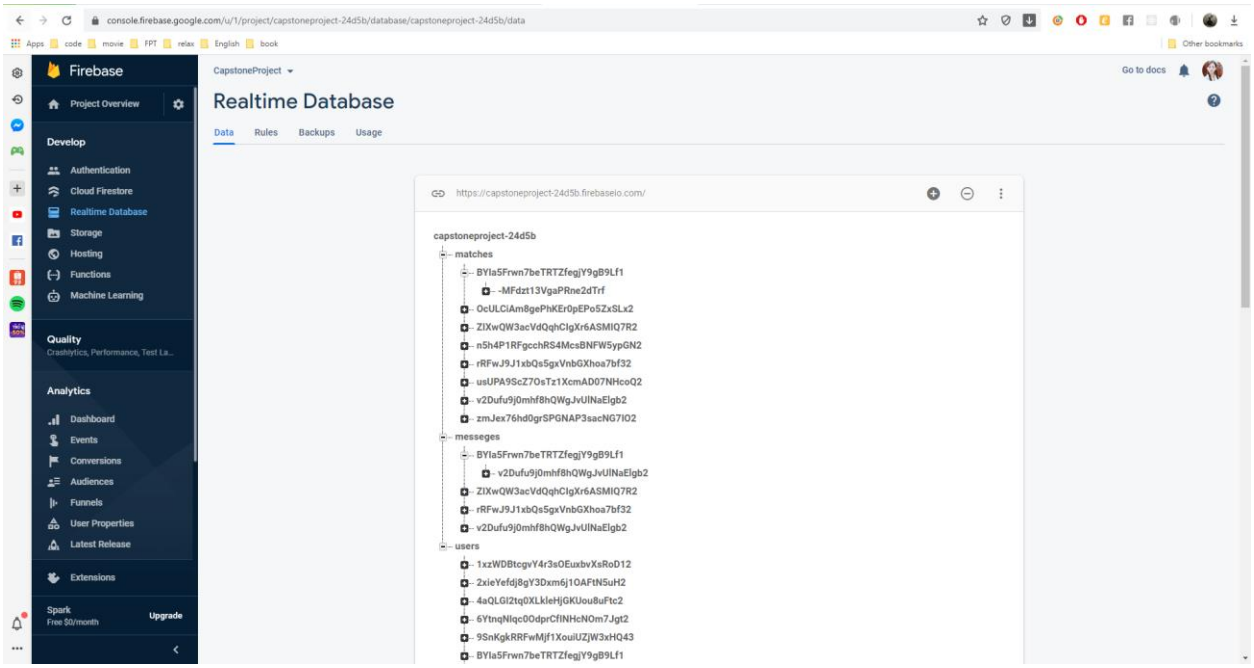


Figure 117 - Realtime Database interface

6.2. User guideline

6.2.1. Login, Logout

6.2.1.1. Login

- Open app
- Choose “LOGIN WITH GOOGLE” or “LOGIN WITH FACEBOOK”
- Fill in with Google or Facebook information

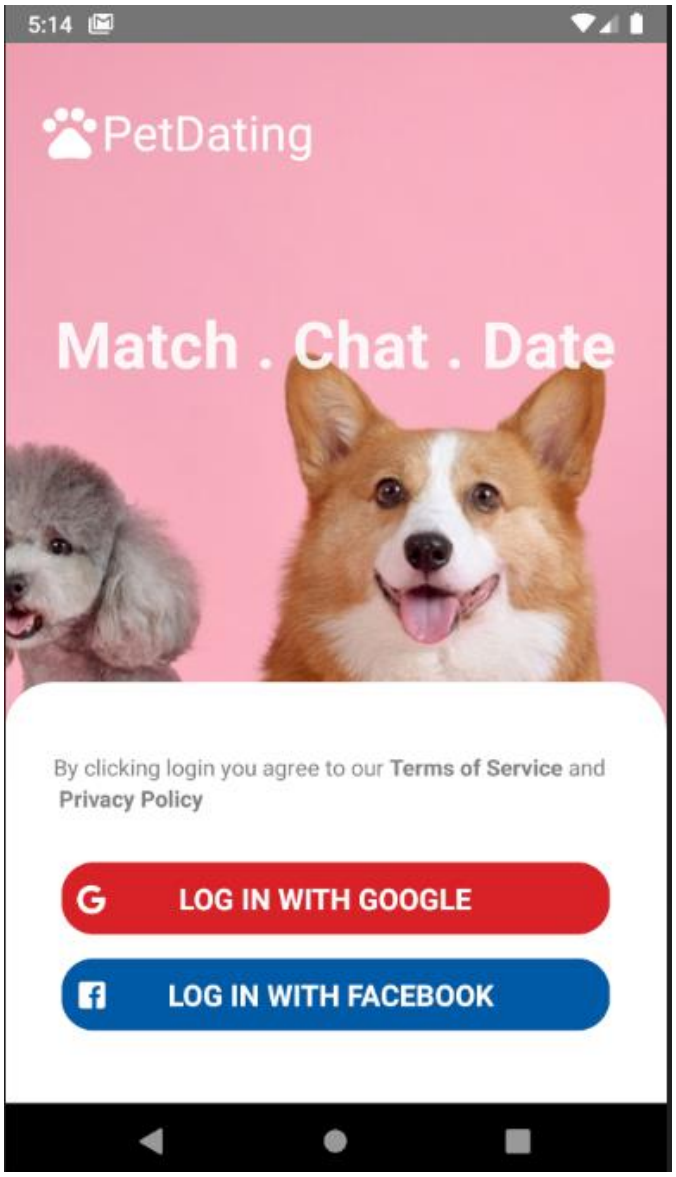


Figure 65 - Login interface

6.2.1.2. Logout

- Switch to Option tab

• Press “Sign out”

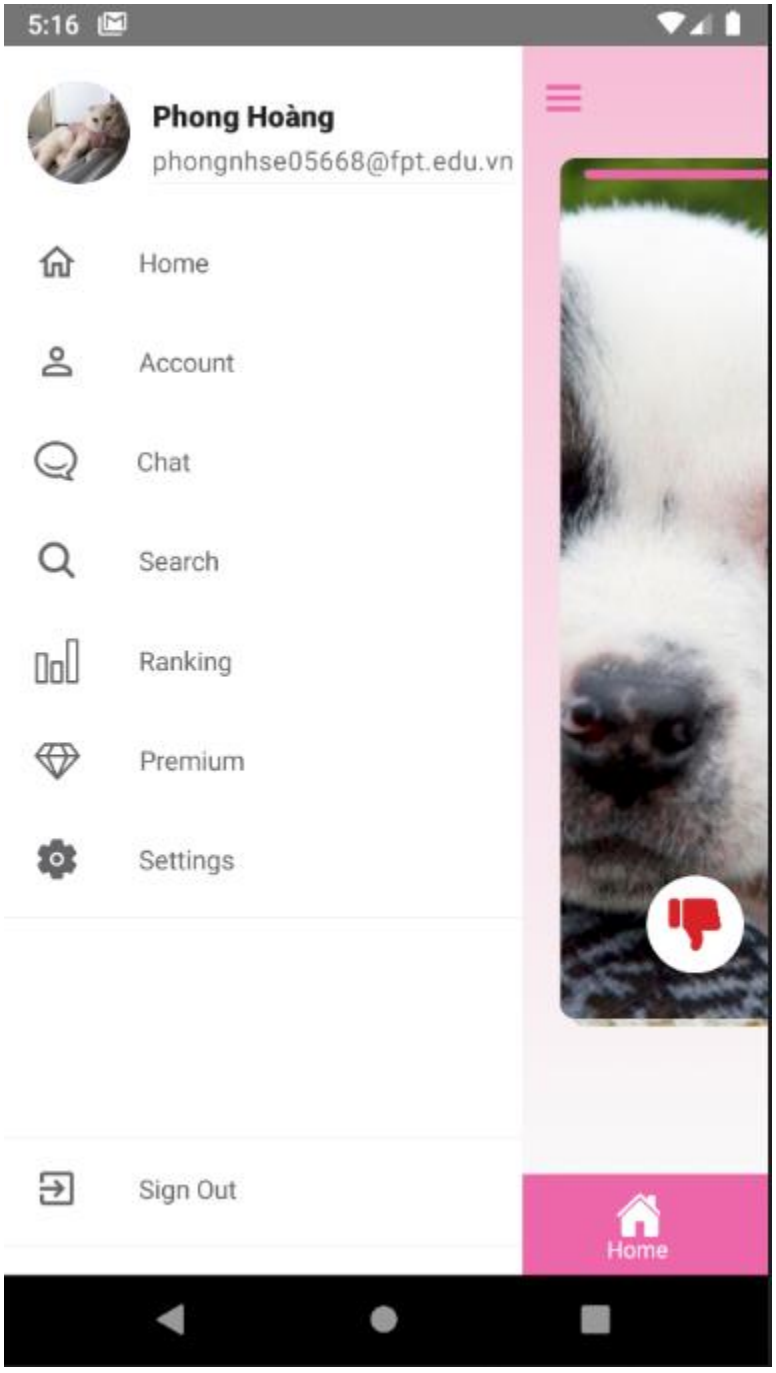


Figure 66 - Logout option in Option tab

6.2.2. Home page

- After logging in, home screen will appear

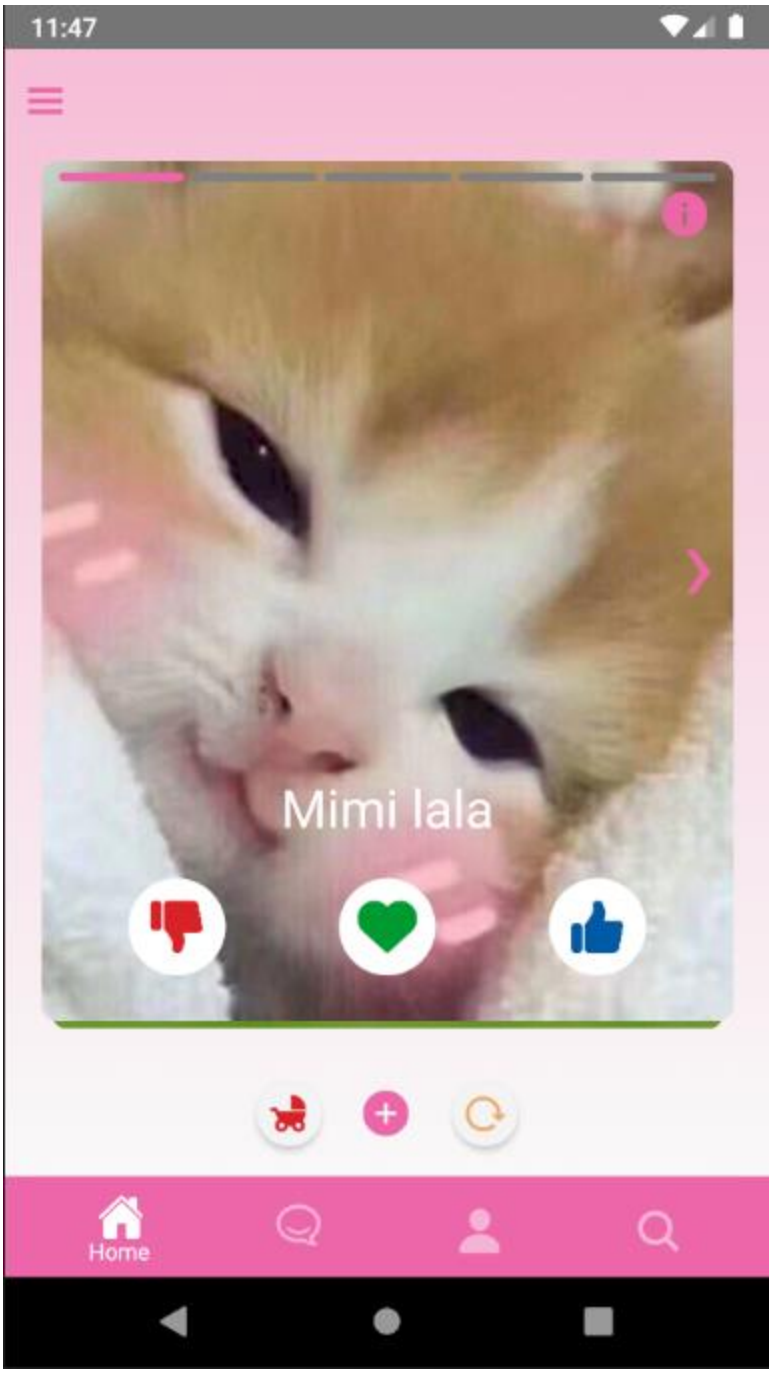


Figure 67- Home page

6.2.3. Pet's profile

- Press "i" in the upper right corner of the card to enter the profile
- Here will display information about the pet and its owner

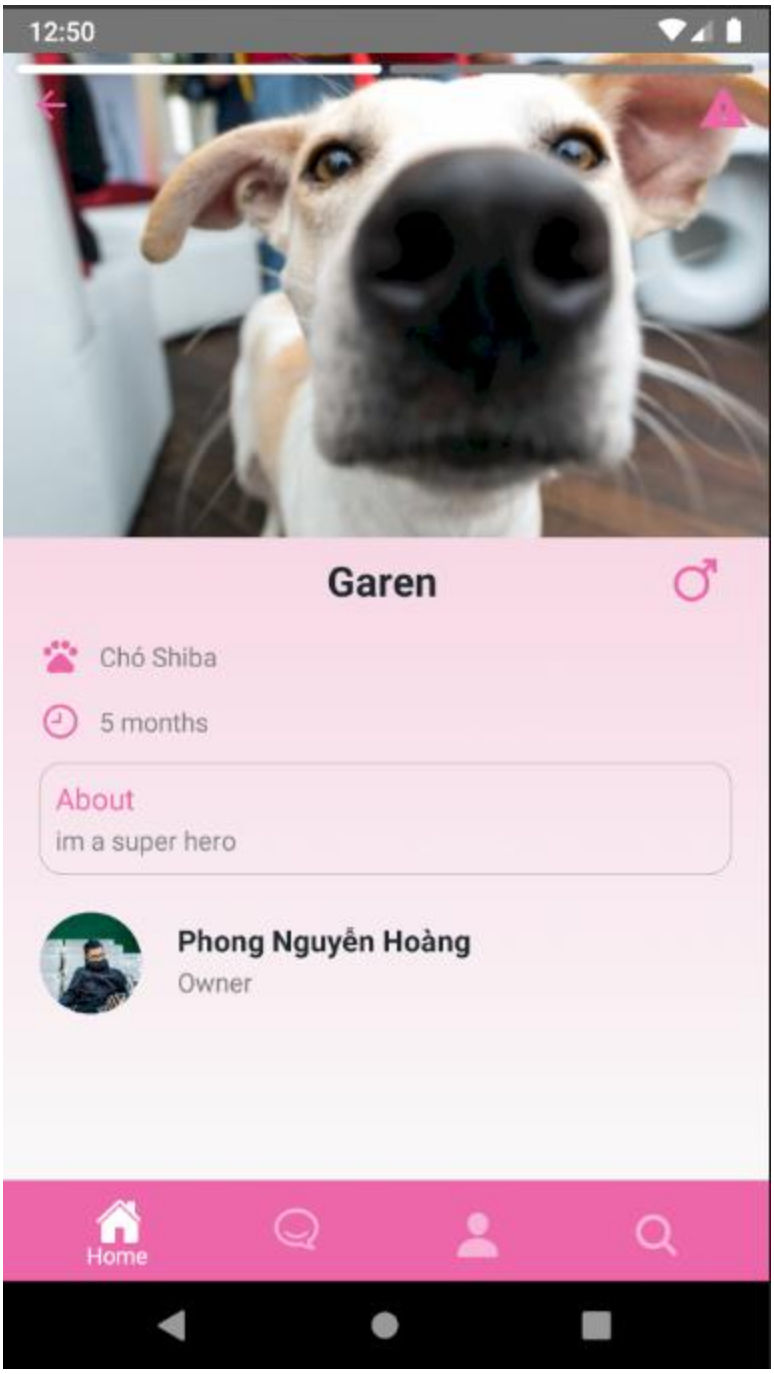


Figure 68- Pet's profile

6.2.4. Report

- Press the button in the top right corner to report this pet

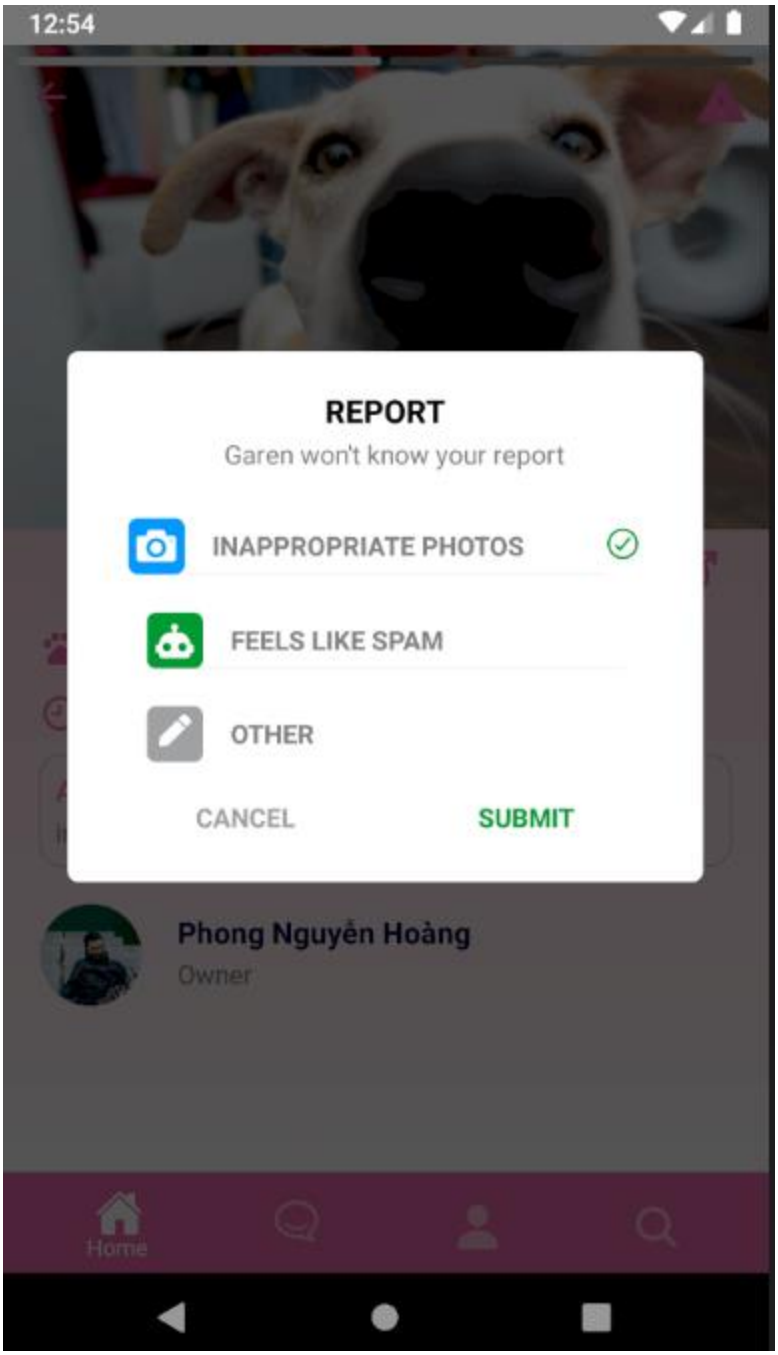


Figure 69- Report pet

6.2.5. React pet

- Press Like, Nope and Match underneath the card to interact with the pet on the screen
- After the interaction is completed, it will switch to a new pet

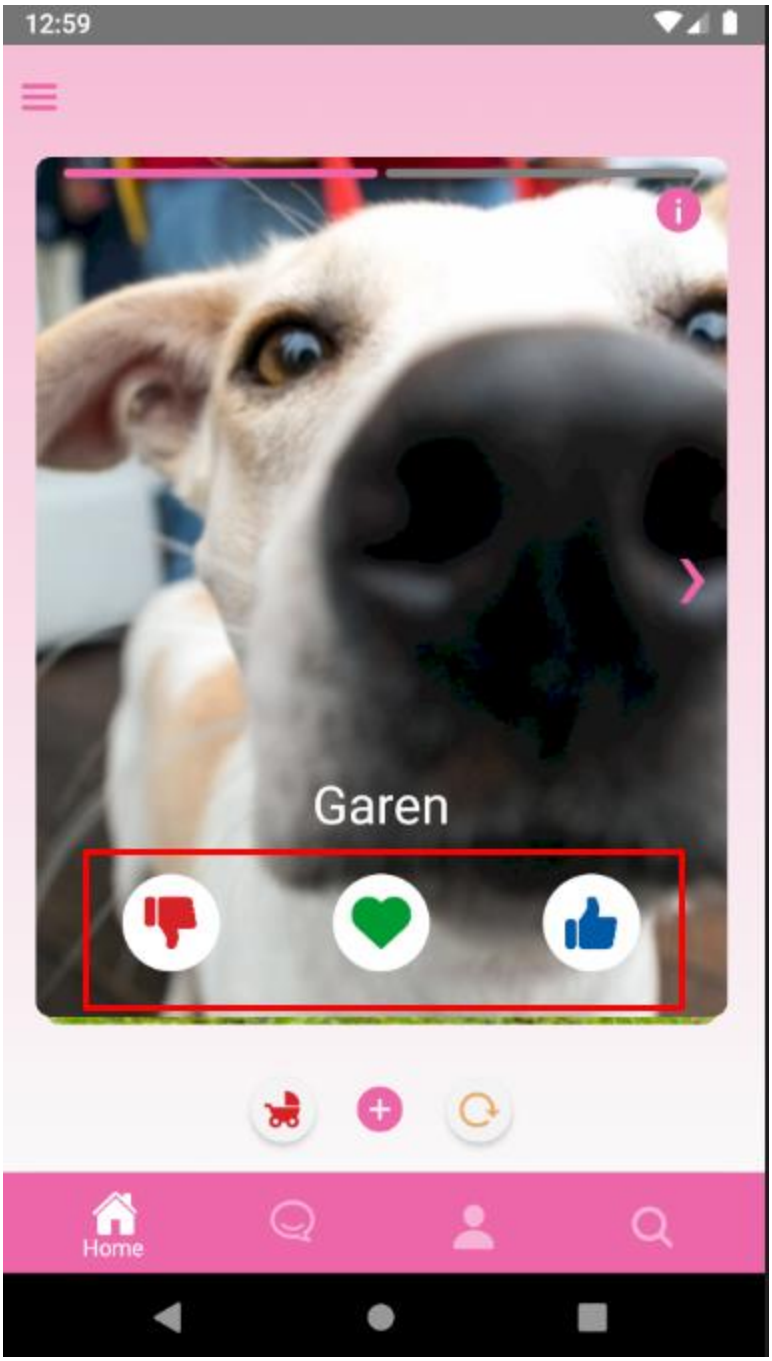


Figure 70- React pet

6.2.6. Rollback

- If you want to return to the pet you have just finished interacting with, press roll back (Premium only)

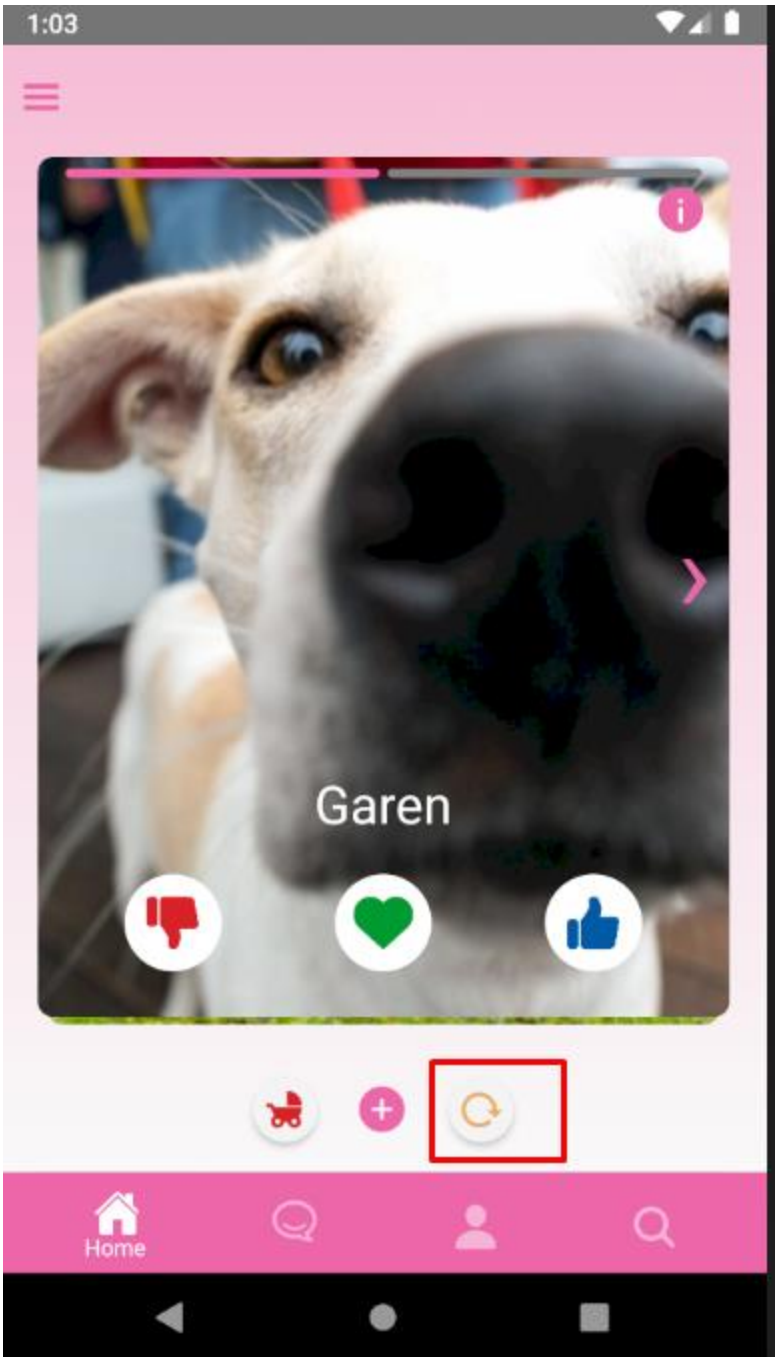


Figure 71- Rollback

6.2.7. Next generation

- If you want to see pictures of children that your pet and other people's pets, press next generation (Premium only)

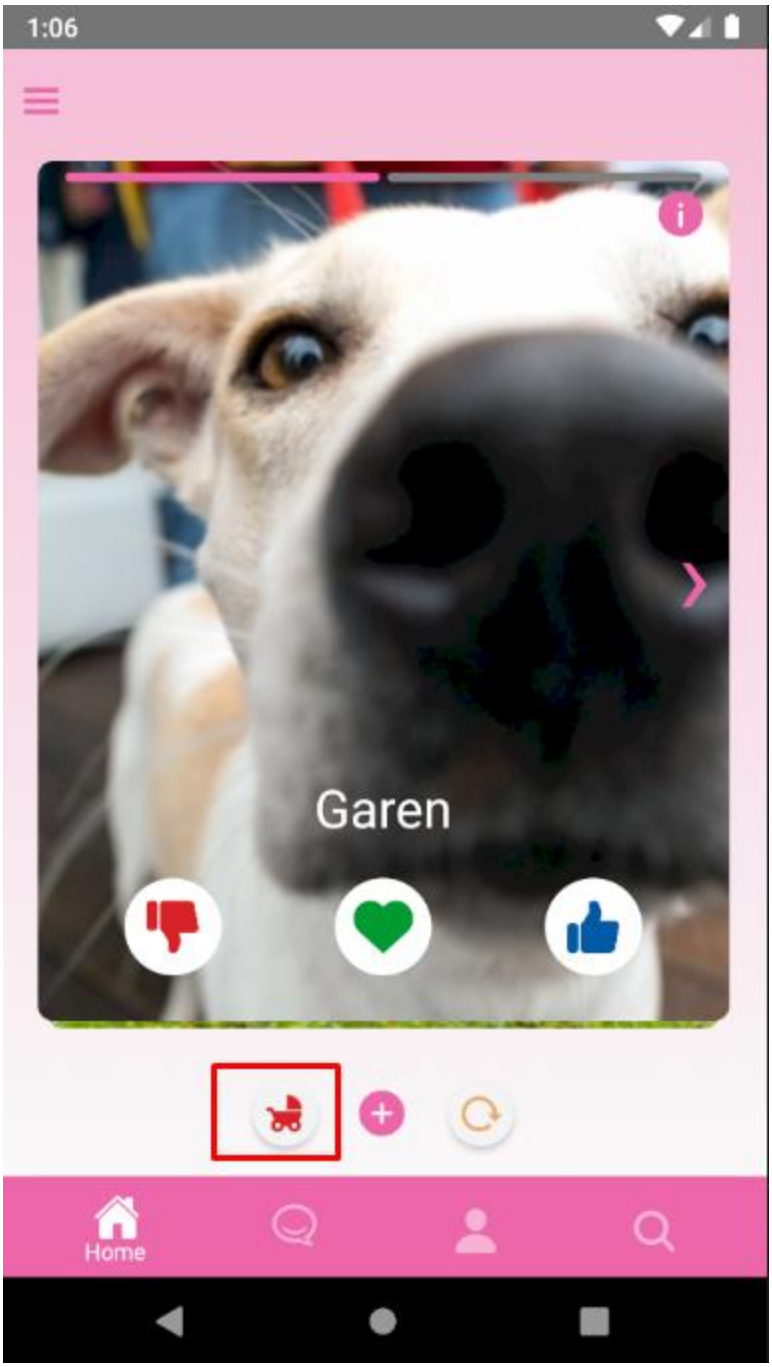


Figure 72- Next generation

6.2.8. Edit user's information

- Edit your personal information
- Fill information in text field

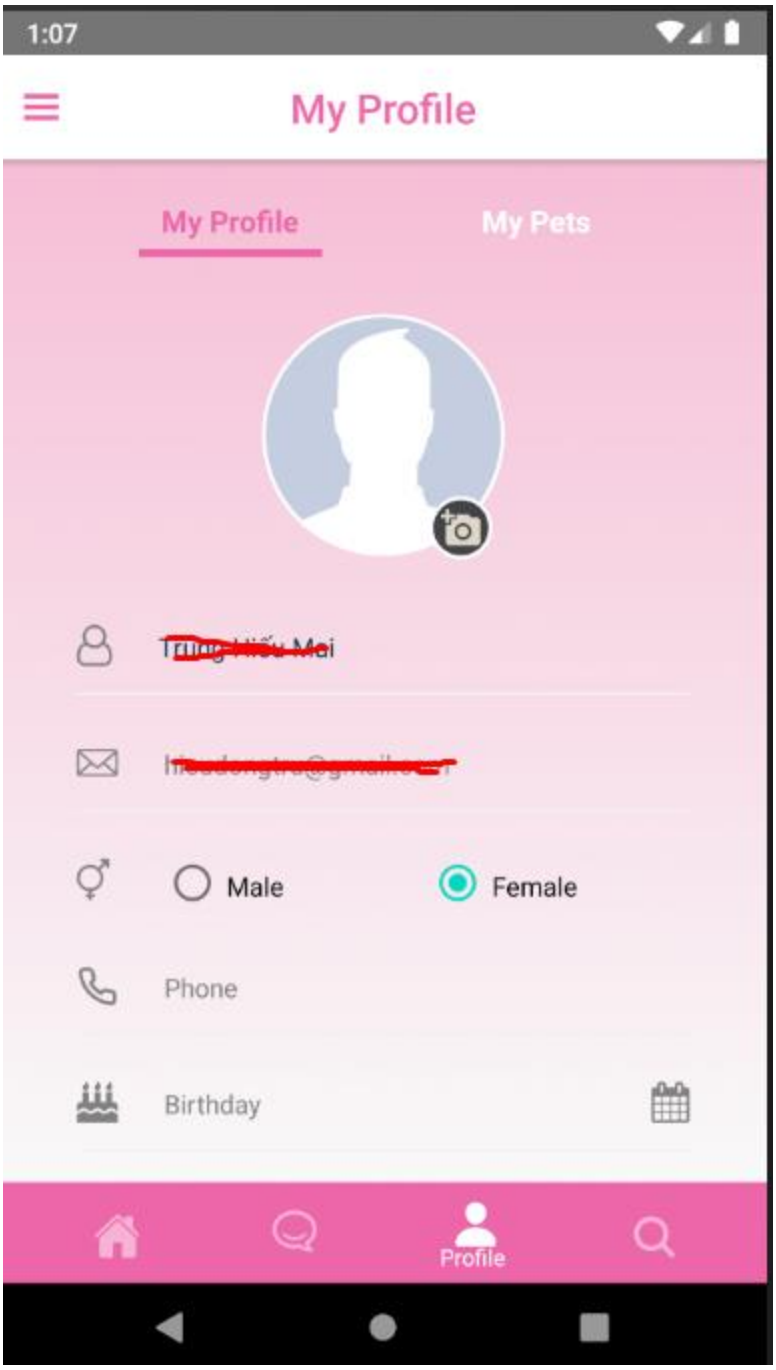


Figure 73- Edit profile

6.2.9. Add new pet

- Press “Add a new pet” to add a new pet for yourself

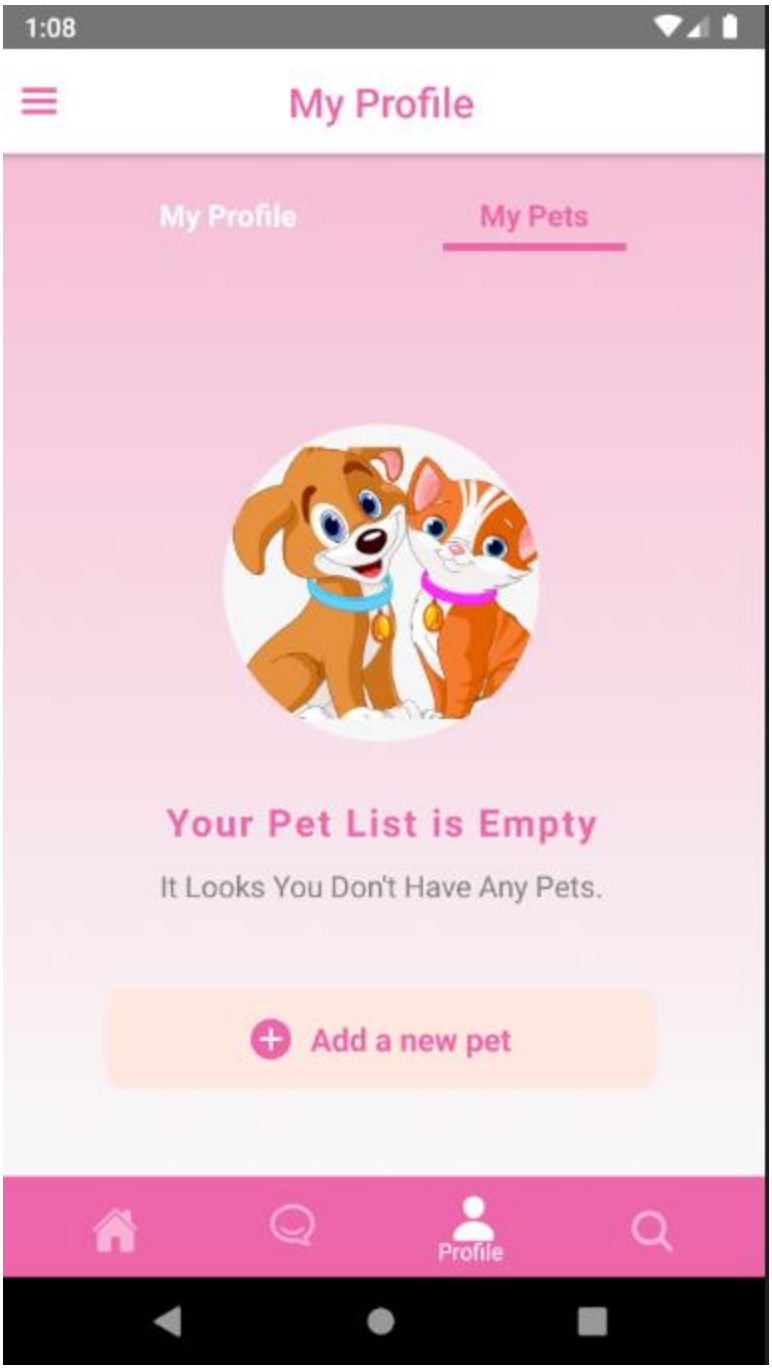


Figure 74- Add new pet

6.2.10. Pet's information

- Fill all needed information, and press "Submit" to create a new pet

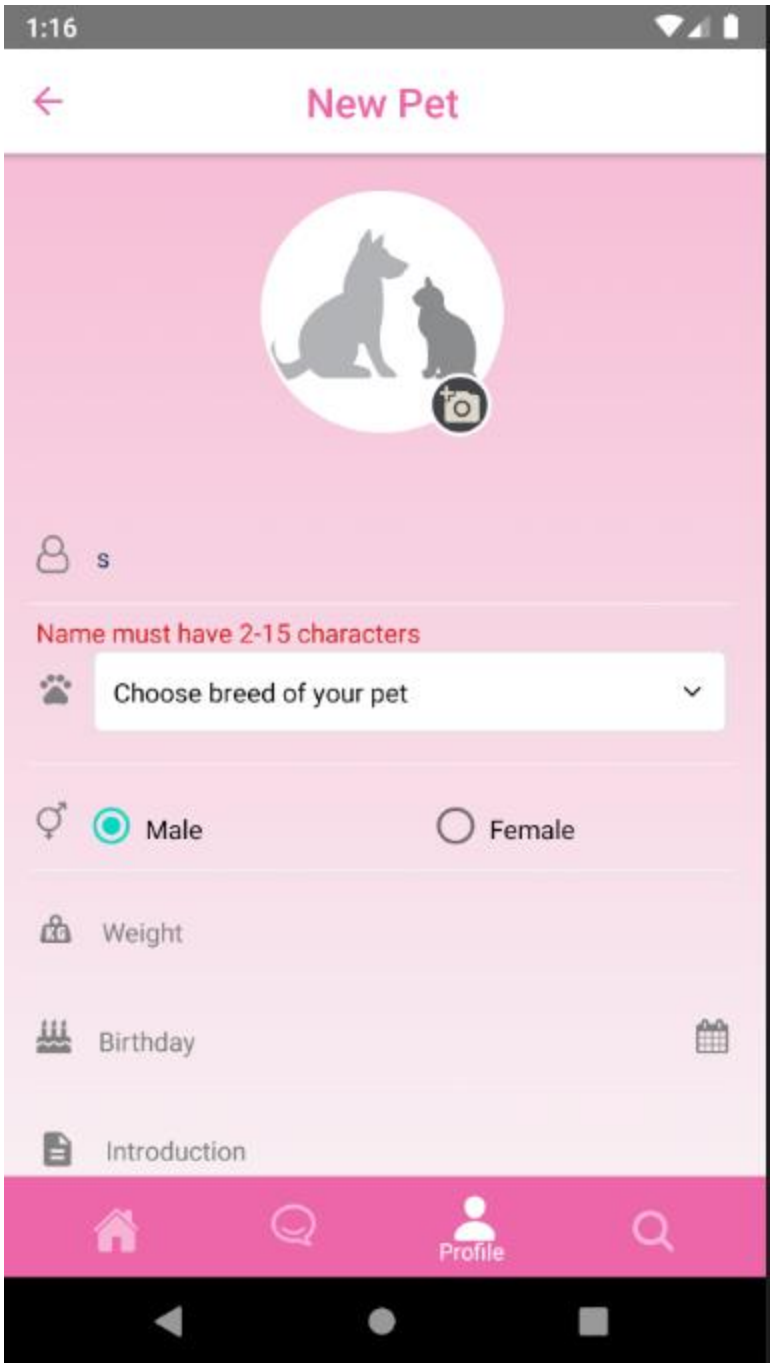


Figure 75- Pet's information

6.2.11. Find

- Drag the slider and click "Find" to find users with a radius of 1 to 100 km

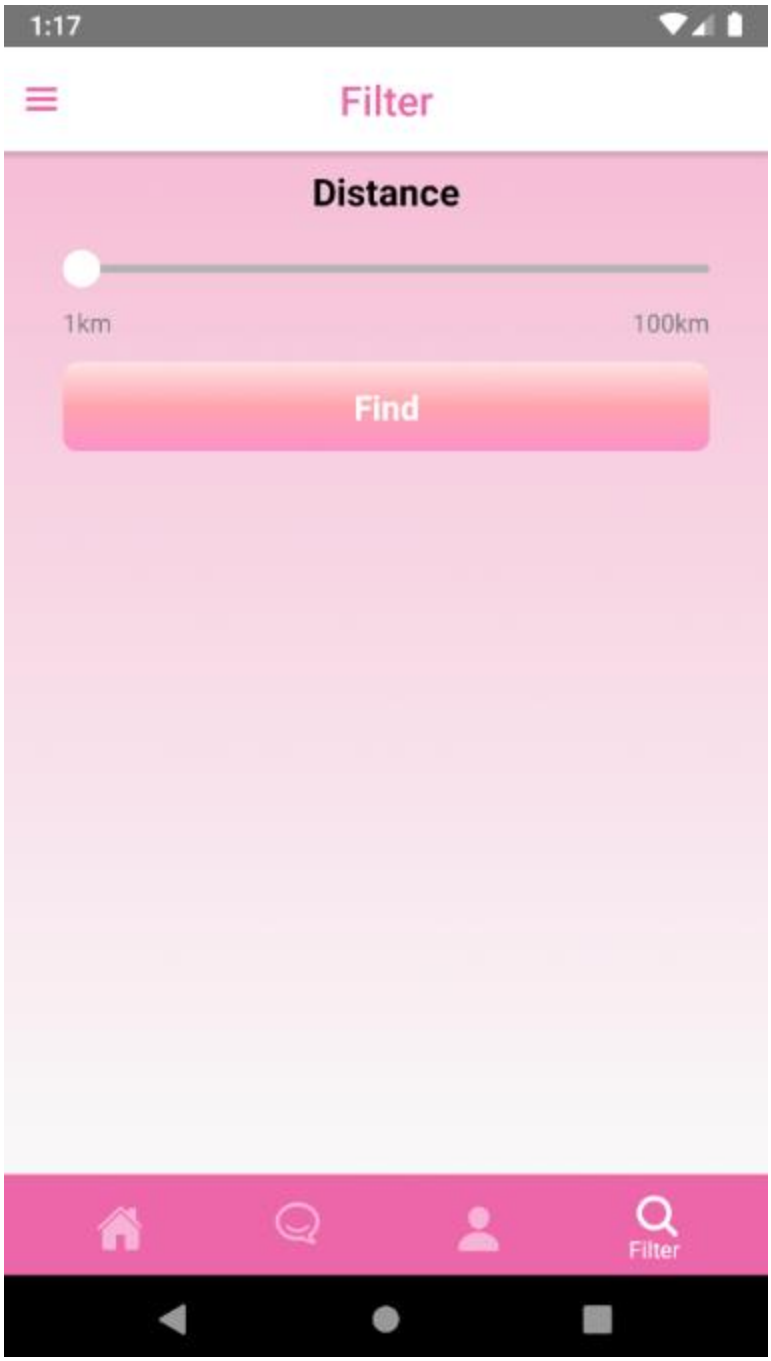


Figure 76- Find nearly

6.2.12. Chatting

- You can send message to everyone you matched

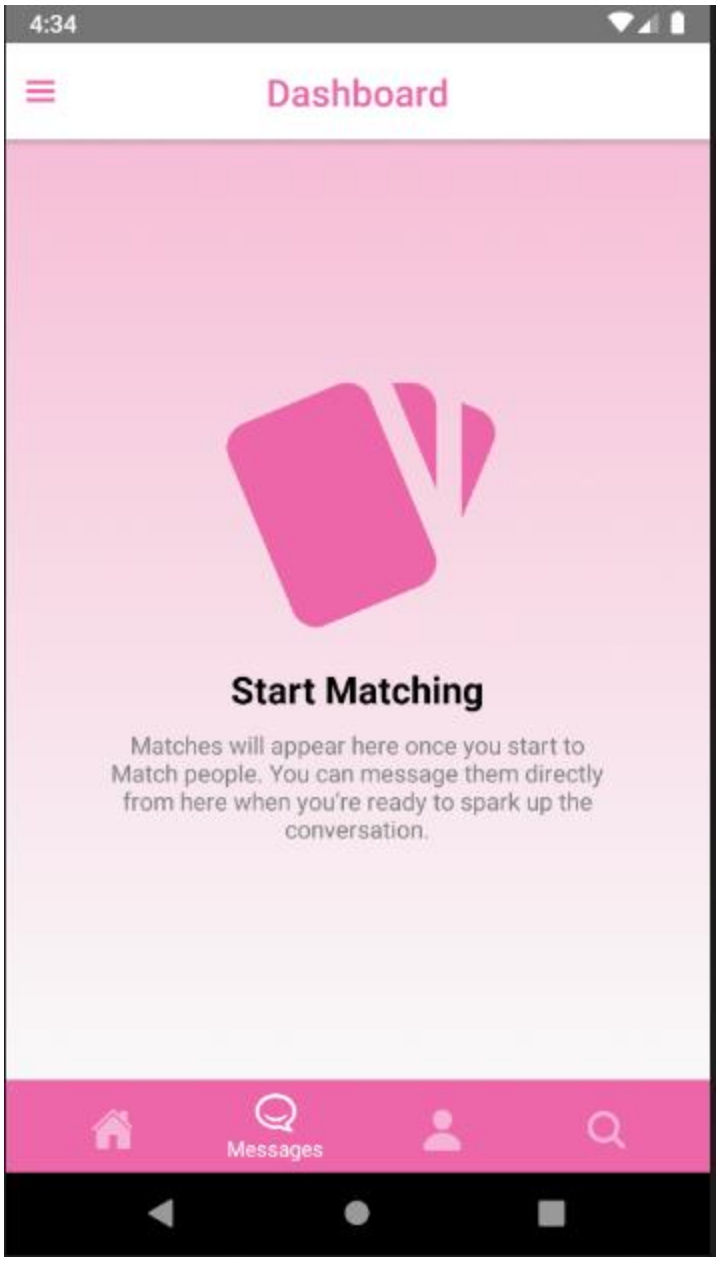
- 

Figure 77- Chatting

6.2.13. Ranking

- Press “Ranking” to see top 10 matched and liked

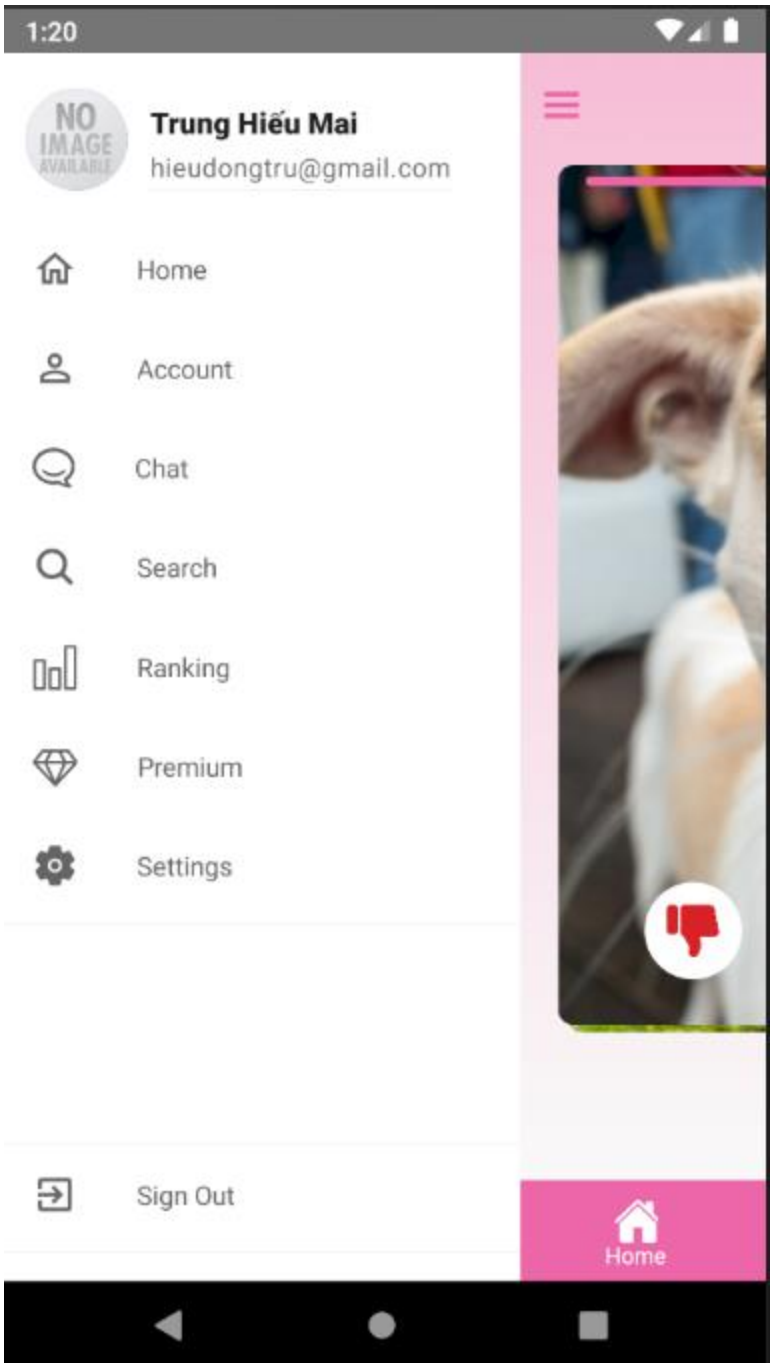


Figure 78- Ranking in menu

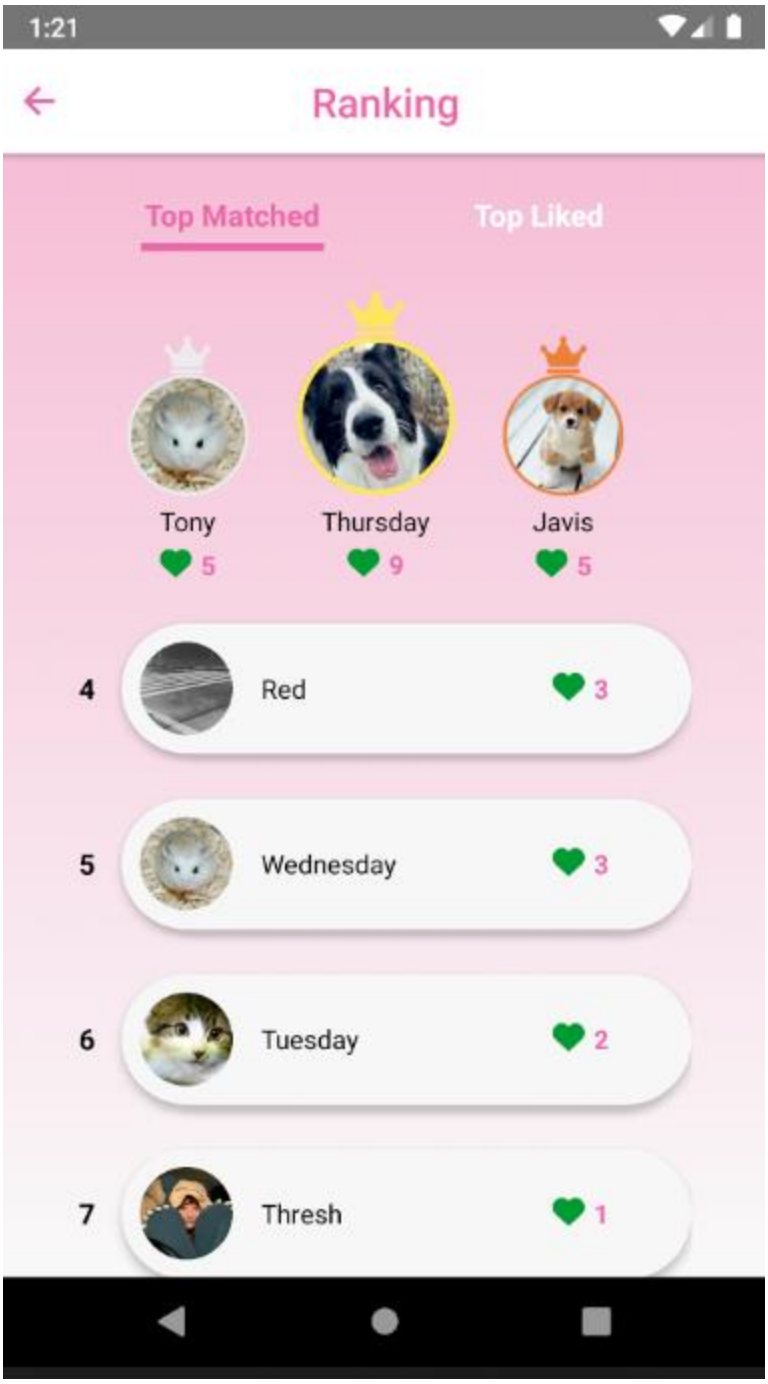


Figure 79- Matching rank

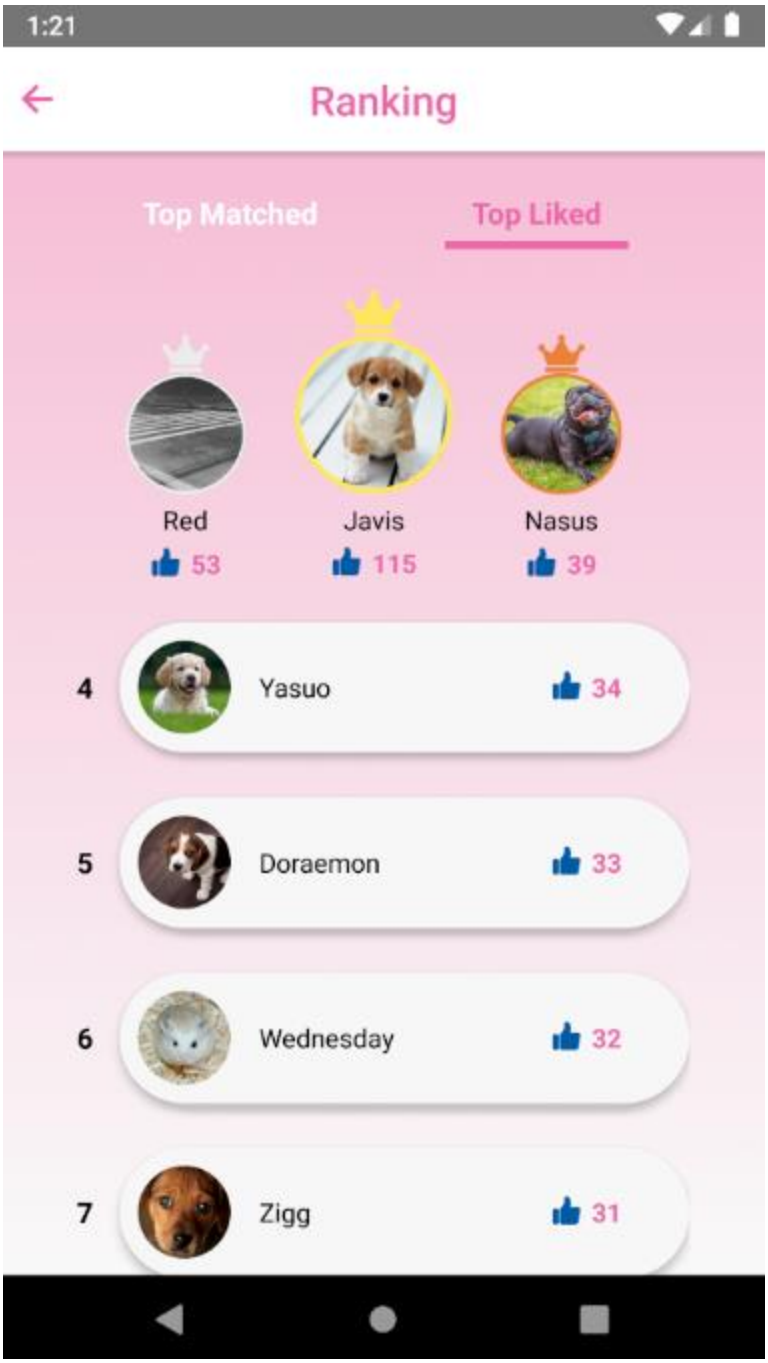


Figure 80- Top like ranking

6.2.14. Premium

- Press “Upgrade Premium” to upgrade your account from regular to premium to unlock all the features

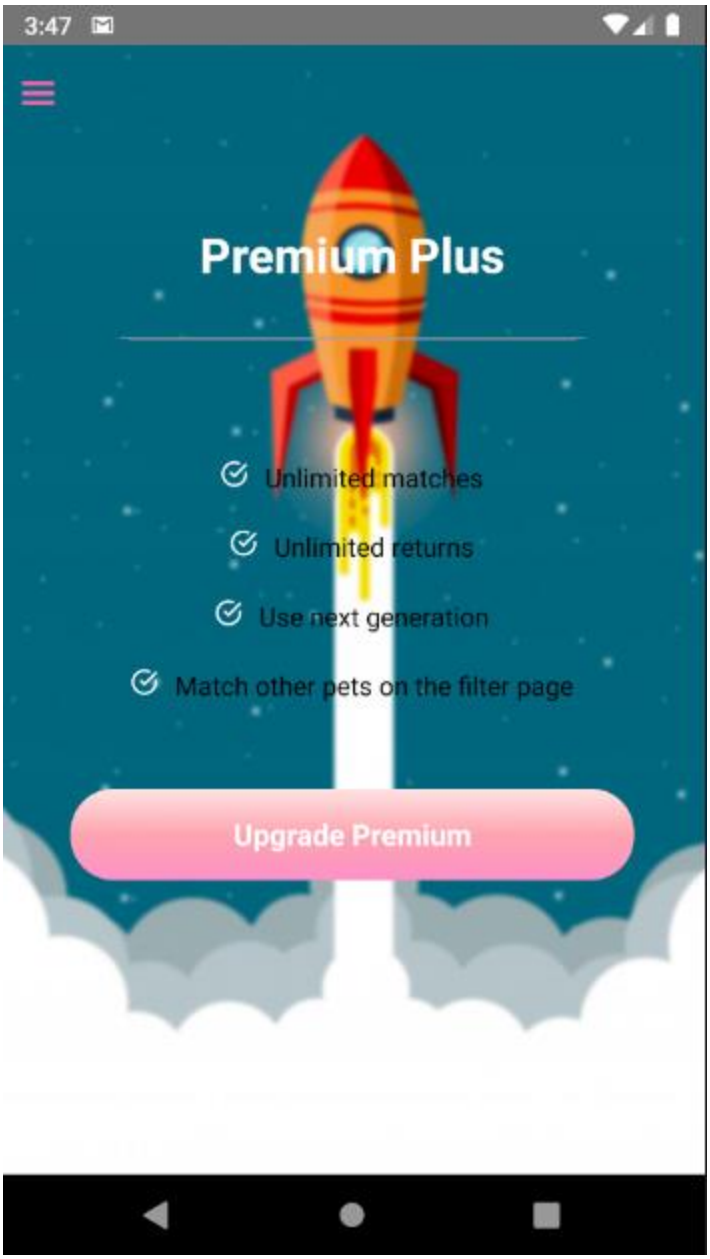


Figure 81- Upgrade Premium

- Press “Use Premium” 1 in 2 to choose which method you want to pay for premium upgrades

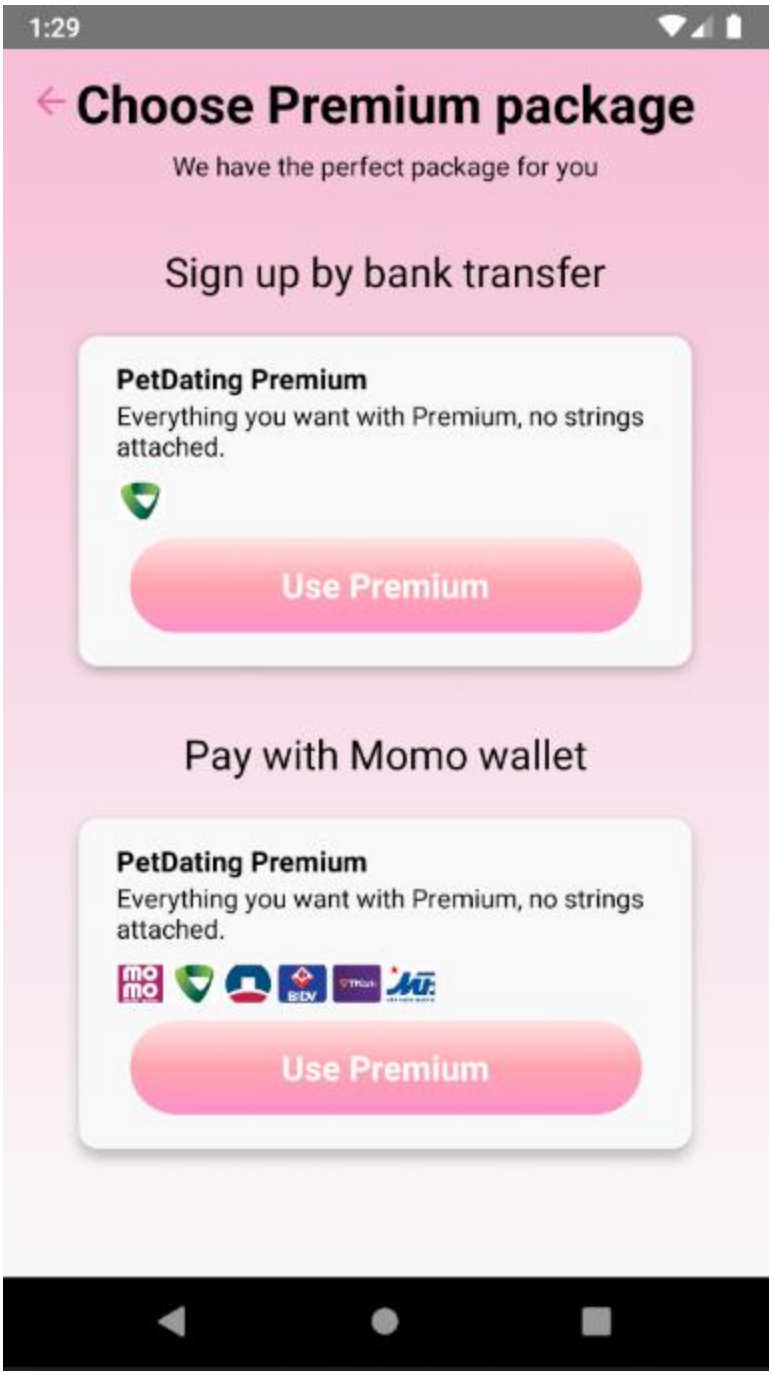


Figure 82- Plans to pay

- Choose 1 in 3 plans to pay for an upgrade
- Click on send photo to resend the invoice photo you paid to us
- When you choose invoice photo completed, the “Upgrade Premium” button will appear and press it to send us invoice photo to confirm

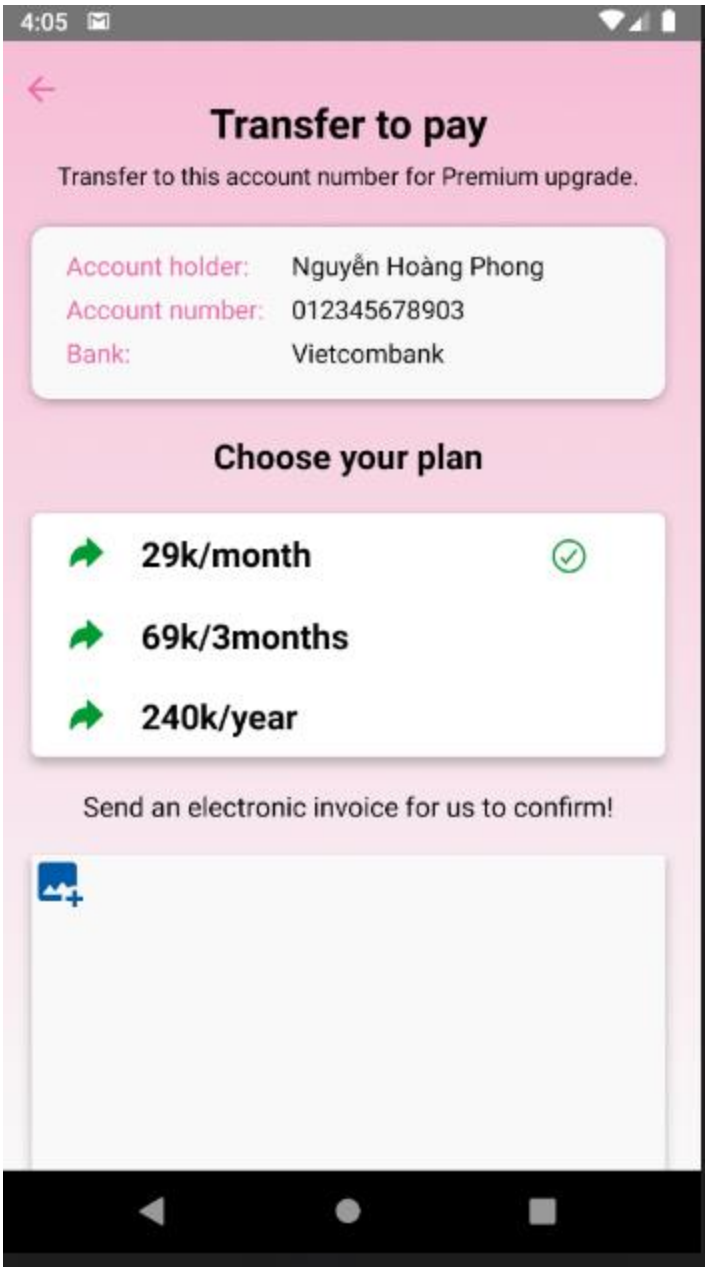


Figure 83- Confirm by image

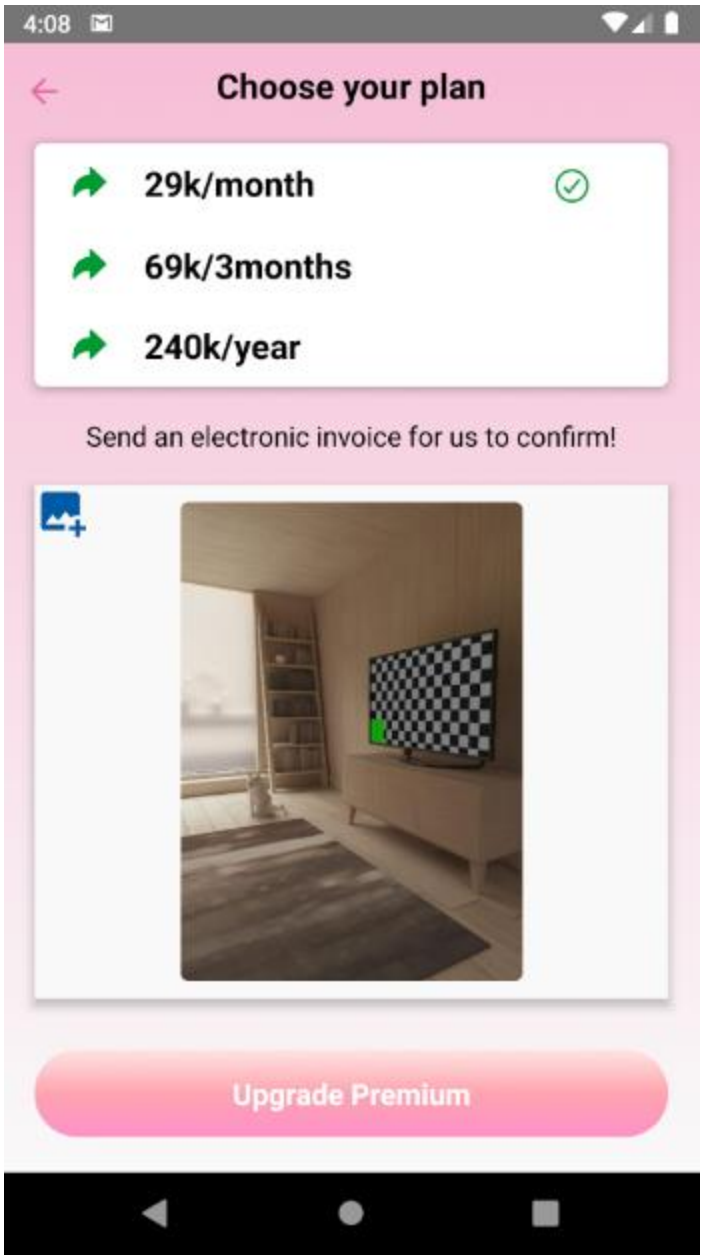


Figure 84- Confirm by image

6.2.15. Settings

- Press “Settings” to use the features like Hide Profile, Feedback and Delete Account

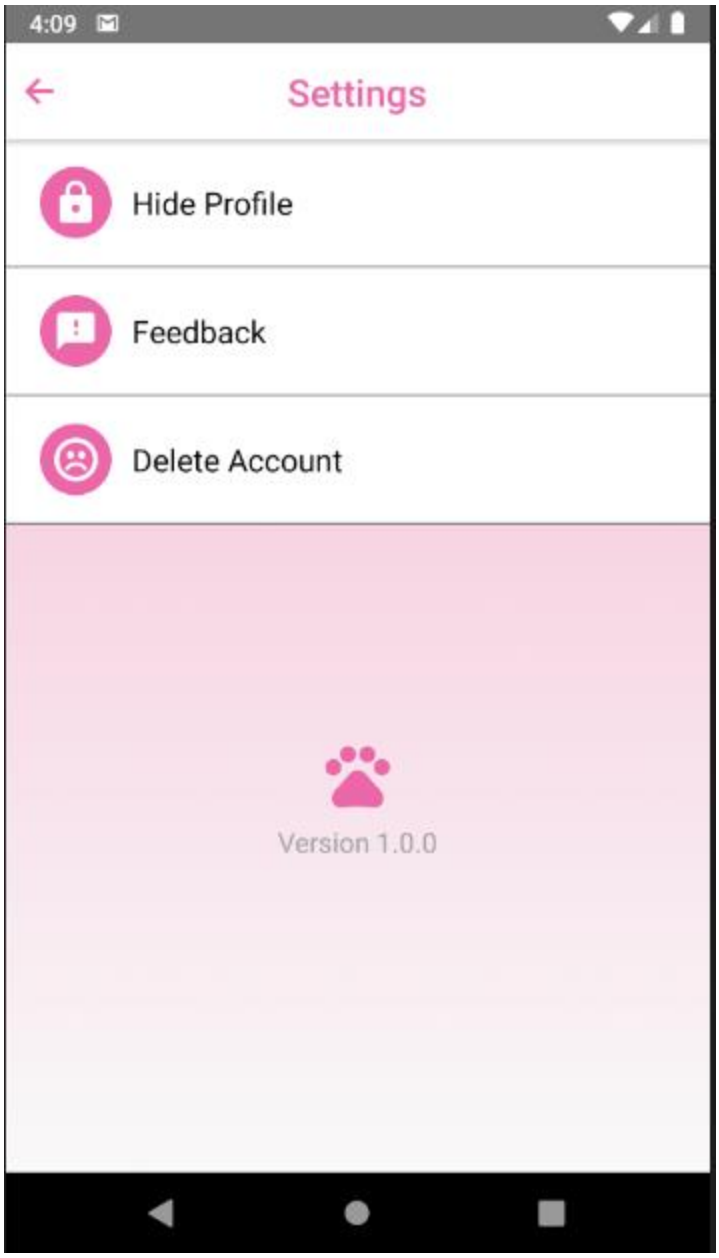


Figure 85- Setting

6.2.15.1. Hide Profile

- Click on checkbox to hide other pets and nobody can see you on this app anymore

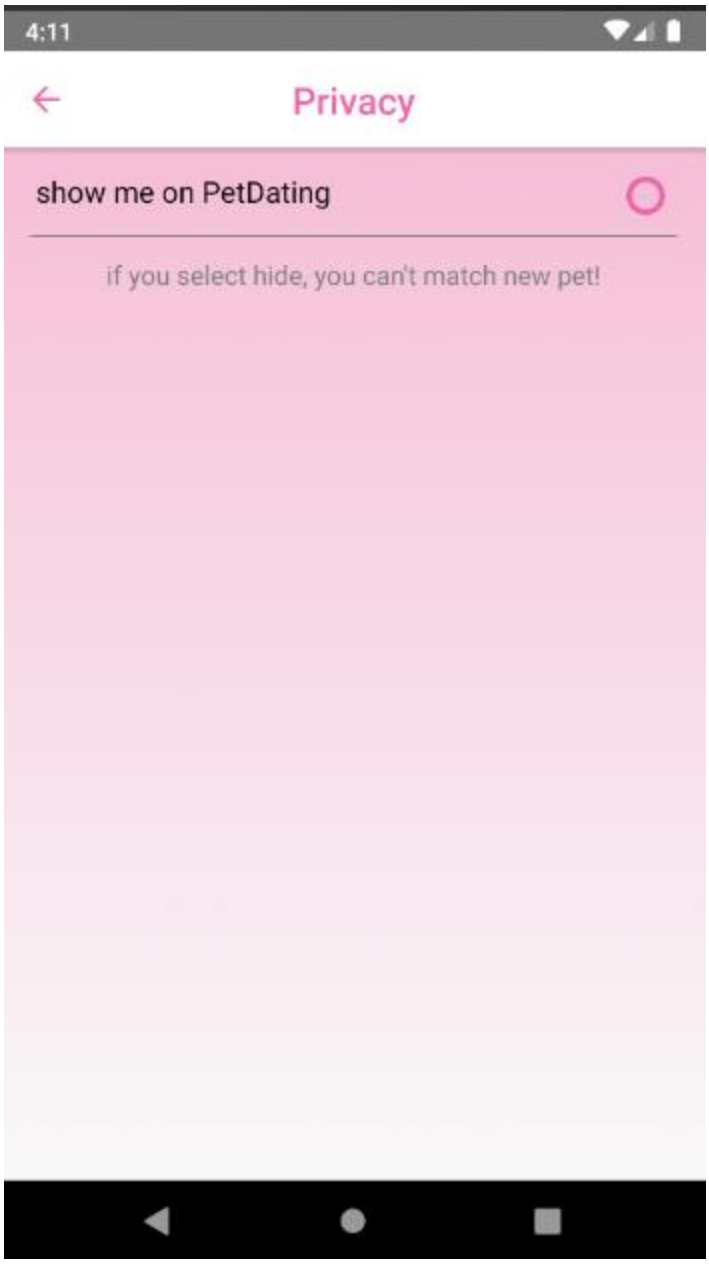


Figure 86- Hide account

6.2.15.2. Feedback

- Choose 1 of 3 reasons or submit content you want to report to us by press “Submit”

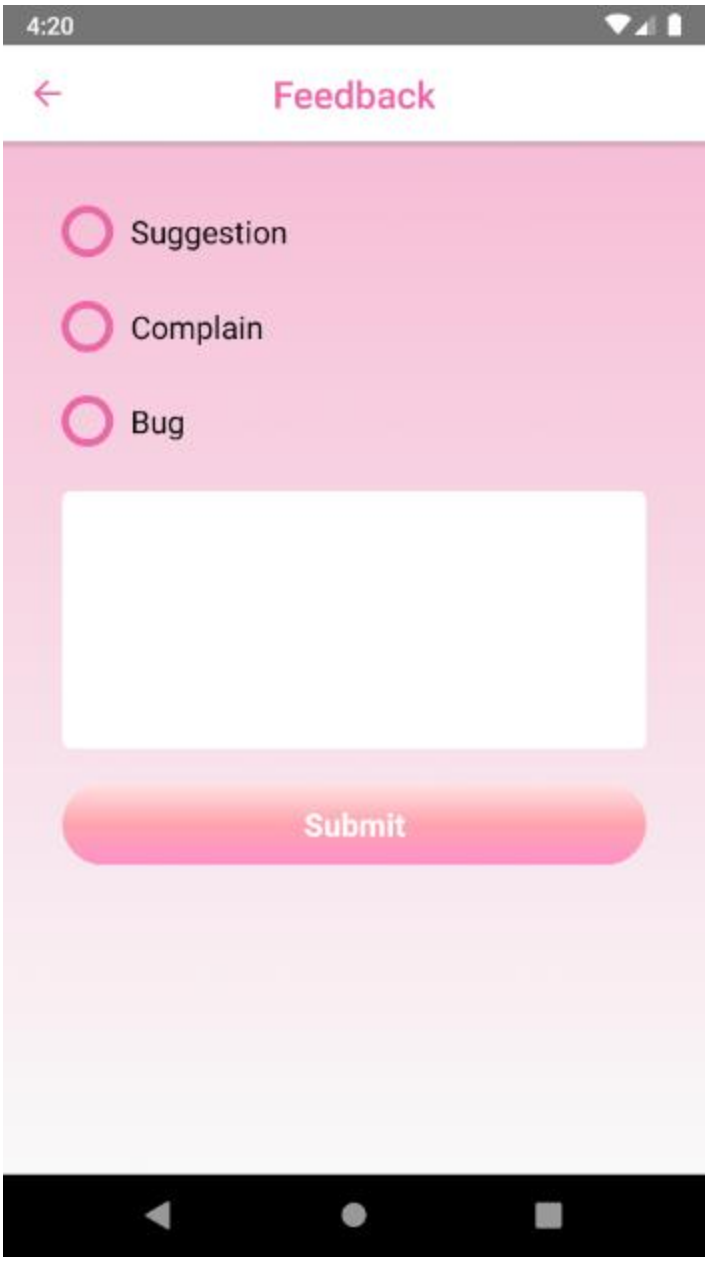


Figure 87- Send feedback

6.2.16.3. Delete Account

- Press “PAUSE MY ACCOUNT” to keep remain your account and just hidden your account
- Press “Delete My Account” to delete your account

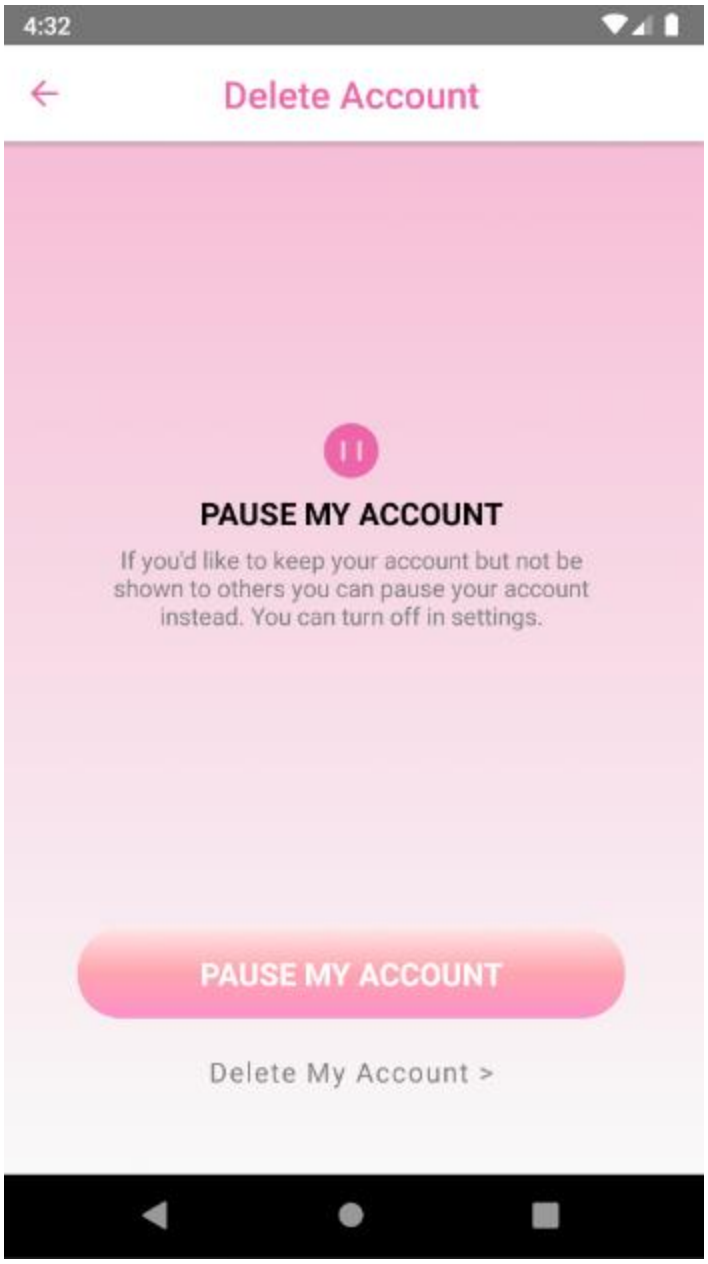


Figure 88- Delete account