|  |  |
| --- | --- |
|  | **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 4](#_Toc49285976)

[1.1. The problem 4](#_Toc49285977)

[1.2. Existing solutions 5](#_Toc49285978)

[1.3. Proposal 6](#_Toc49285979)

[1.3.1. The idea 6](#_Toc49285980)

[1.3.2. The Proposal 7](#_Toc49285981)

[1.3.3. Business flow 7](#_Toc49285982)

[CHAPTER 2: SOFTWRE PROJECT MANAGEMENT PLAN (SPMP) 8](#_Toc49285983)

[2.1. Purpose 8](#_Toc49285984)

[2.2. Software Process Model 9](#_Toc49285985)

[2.2.1. About the SCRUM 9](#_Toc49285986)

[2.2.2. Advantages and disadvantages of SCRUM 10](#_Toc49285987)

[2.3. Project Organization 11](#_Toc49285988)

[2.3.1. Organization structure 11](#_Toc49285989)

[2.3.2. Roles and responsibilities 11](#_Toc49285990)

[2.4. Tools and Techniques 13](#_Toc49285991)

[2.5. Project Management Plan 14](#_Toc49285992)

[2.5.1. Project schedule 14](#_Toc49285993)

[2.5.2. Meeting minutes 14](#_Toc49285994)

[2.5.3. Risk management plan 15](#_Toc49285995)

[2.5.4. Communication plan 16](#_Toc49285996)

[2.5.5. Coding convention 17](#_Toc49285997)

[CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATION 17](#_Toc49285998)

[3.1. Purpose 17](#_Toc49285999)

[3.2. Functional Requirement 18](#_Toc49286000)

[3.2.1. Use case diagram 18](#_Toc49286001)

[.2.2. Business rules 20](#_Toc49286002)

[3.2.3. Use case list 20](#_Toc49286003)

[3.2.4. Use case specification 22](#_Toc49286004)

[3.3. Non-functional Requirement 62](#_Toc49286005)

[3.3.1. Security 62](#_Toc49286006)

[3.3.2. Availability 62](#_Toc49286007)

[3.3.3. Usability 63](#_Toc49286008)

[CHAPTER 4: SPOFTWARE DESIGN 63](#_Toc49286009)

[4.1. Purpose 63](#_Toc49286010)

[4.2. Overview of System Architecture 63](#_Toc49286011)

[4.2.1. Diagram 63](#_Toc49286012)

[4.2.2. Component Explanation 64](#_Toc49286013)

[4.3. Application Custom MVC Design 66](#_Toc49286014)

[4.4. Database Design 66](#_Toc49286015)

[4.4.1. Entity relationship diagram 66](#_Toc49286016)

[4.4.2. Database diagram 69](#_Toc49286017)

[4.5. Detailed Design 76](#_Toc49286018)

[4.5.1. Login 76](#_Toc49286019)

[s4.5.2. Logout 79](#_Toc49286020)

[4.5.3. Matching 81](#_Toc49286021)

[4.5.4. Chat 85](#_Toc49286022)

[4.5.5. Ranking 93](#_Toc49286023)

[4.5.6. Feedback 96](#_Toc49286024)

[4.5.7. React pet 99](#_Toc49286025)

[4.5.8. View other user 103](#_Toc49286026)

[4.5.9. View other user’s pet 105](#_Toc49286027)

[4.5.10. Hide profile 109](#_Toc49286028)

[4.5.11. Edit user’s profile 112](#_Toc49286029)

[4.5.12. Delete user’s profile 115](#_Toc49286030)

[4.5.13. Create new pet 118](#_Toc49286031)

[4.5.14. Edit pet’s profile 121](#_Toc49286032)

[4.5.15. Delete pet’s profile 124](#_Toc49286033)

[4.5.16. Find 127](#_Toc49286034)

[4.5.17. Get next generation pet 130](#_Toc49286035)

[4.5.18. Login/Logout for Admin 132](#_Toc49286036)

[4.5.20. Enable/Disable Account for Admin 136](#_Toc49286037)

[4.5.22. Delete Images for Admin 139](#_Toc49286038)

[4.5.23. Manage system for Admin 141](#_Toc49286039)

[4.5.24. Receive Feedback for Admin 144](#_Toc49286040)

[CHAPTER 5: Software Testing Document 147](#_Toc49286041)

[5.1. Introduction 147](#_Toc49286042)

[5.1.1. Purpose 147](#_Toc49286043)

[5.1.2. Scope of testing 147](#_Toc49286044)

[5.1.3. Range of testing 147](#_Toc49286045)

[5.2. Test plan 147](#_Toc49286046)

[5.2.1. Testing Tools and Environment 147](#_Toc49286047)

[5.2.2. Resources and Responsibilities 147](#_Toc49286048)

[5.2.3. Test Strategy 147](#_Toc49286049)

[5.2.4. Features to be tested 148](#_Toc49286050)

[5.3. Test case 148](#_Toc49286051)

[5.3.1. Unit test 148](#_Toc49286052)

[5.3.2. System testing 148](#_Toc49286053)

[5.3.3. Acceptance test 148](#_Toc49286054)

[5.3.4. Defect Log 148](#_Toc49286055)

[5.4. Test Report 148](#_Toc49286056)

[5.4.1. Unit test 148](#_Toc49286057)

[5.4.2. Integration and system tests 148](#_Toc49286058)

[CHAPTER 6: USER MANUAL 148](#_Toc49286059)

[6.1. Development and deployment guidelines 148](#_Toc49286060)

[6.1.1.1. Environment for development 148](#_Toc49286061)

# CHAPTER 1: INTRODUCTION

## 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.

## 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**

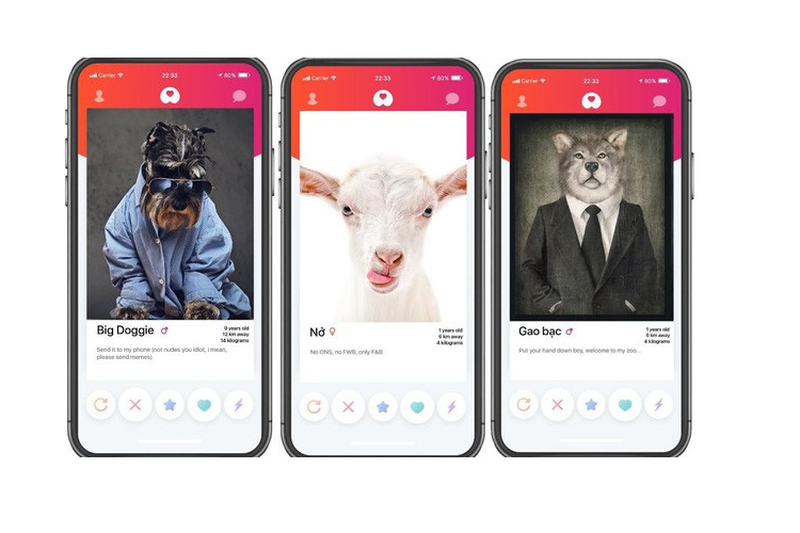


Figure - Get bone (app)

|  |
| --- |
| **Advantages** |
| * 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**

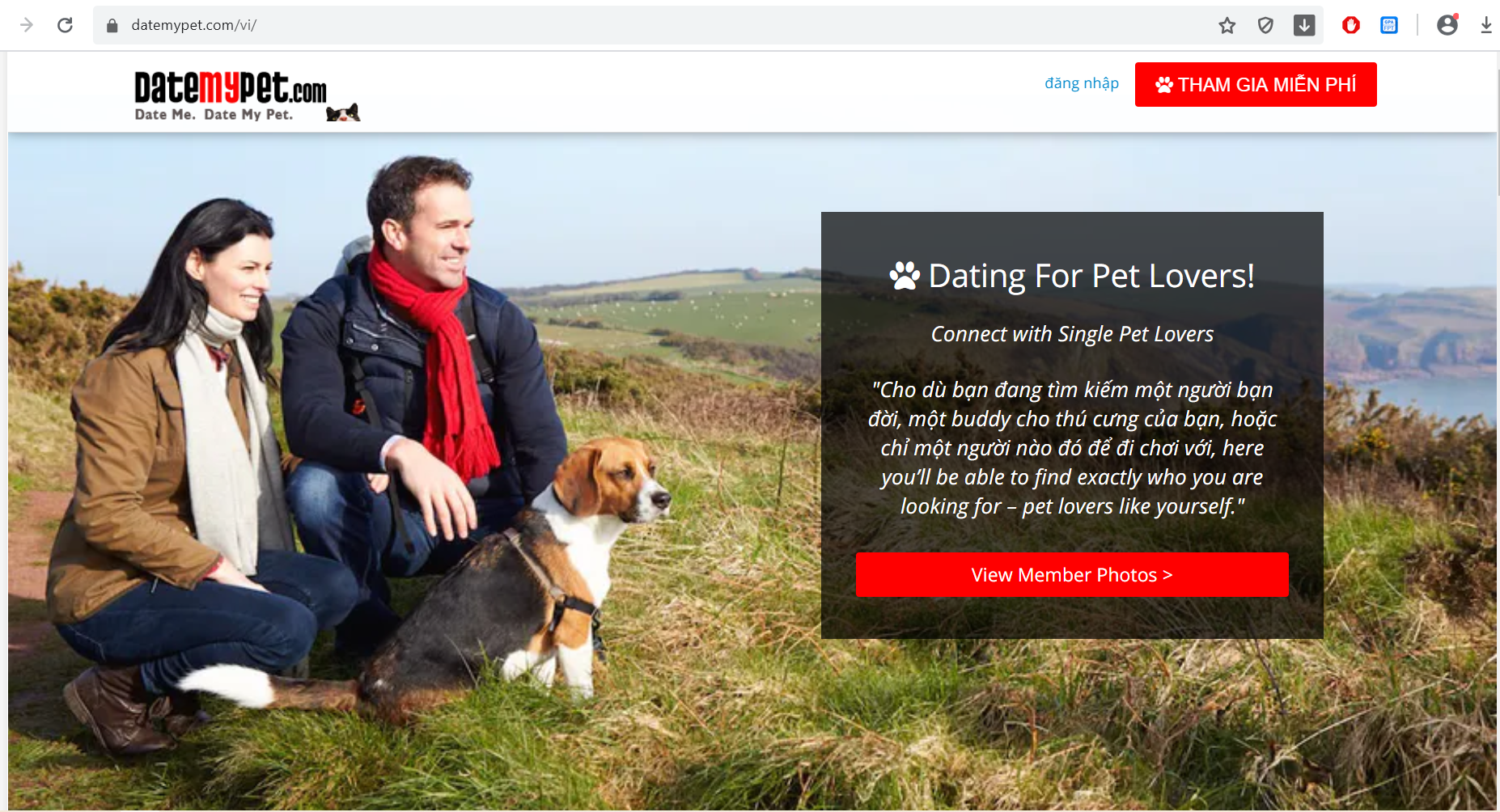


Figure 2 - Datemypet.com

|  |
| --- |
| **Advantages** |
| * Provide news and more information about pets. * Not support finding mates. * Only news, no communication between users |

## Proposal

### 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.

### The Proposal

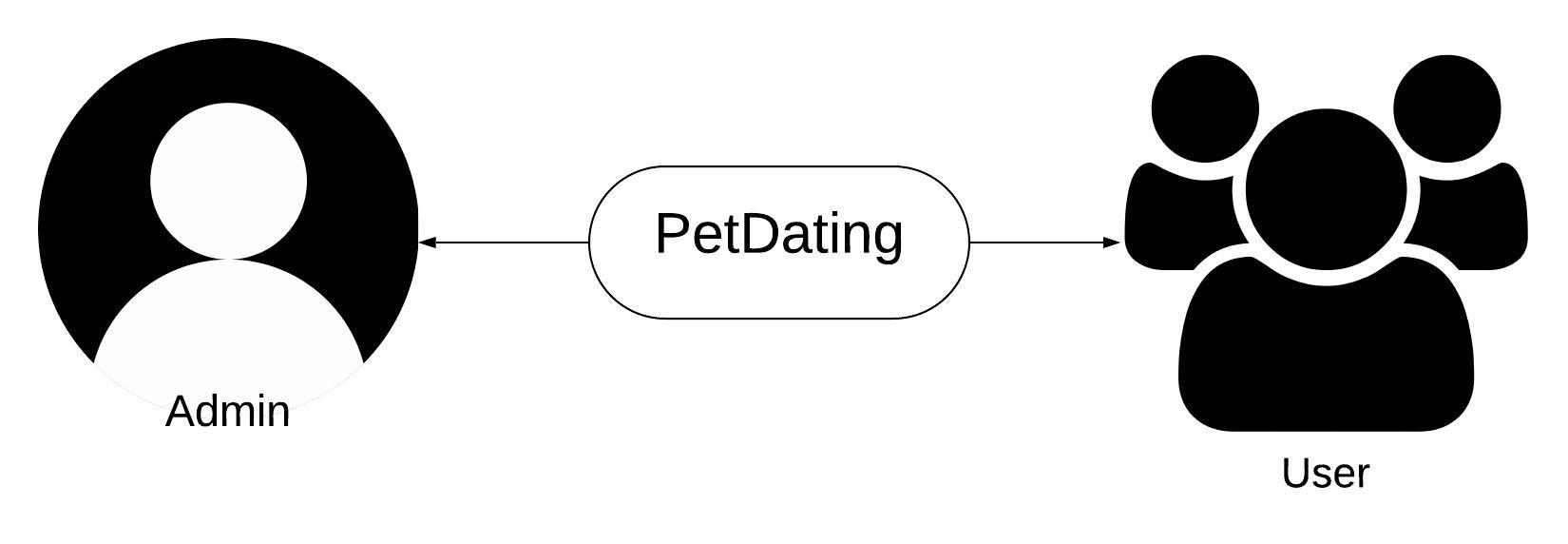


Figure - Roles in PD system

### 1.3.3. Business flow

* **Review report process**

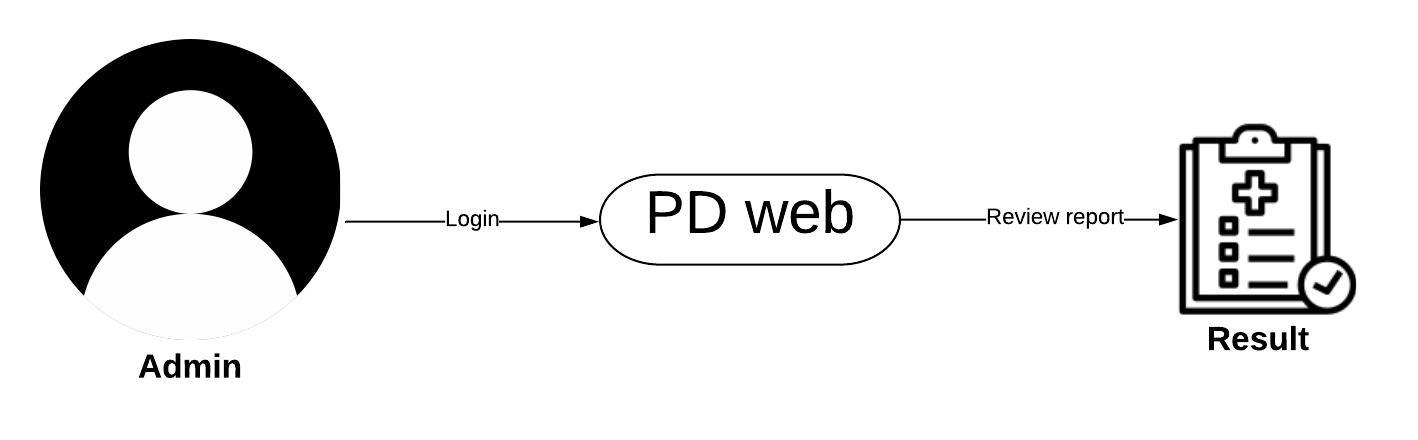


Figure 4 - Process of review report on PD

* **Ban and unban user**

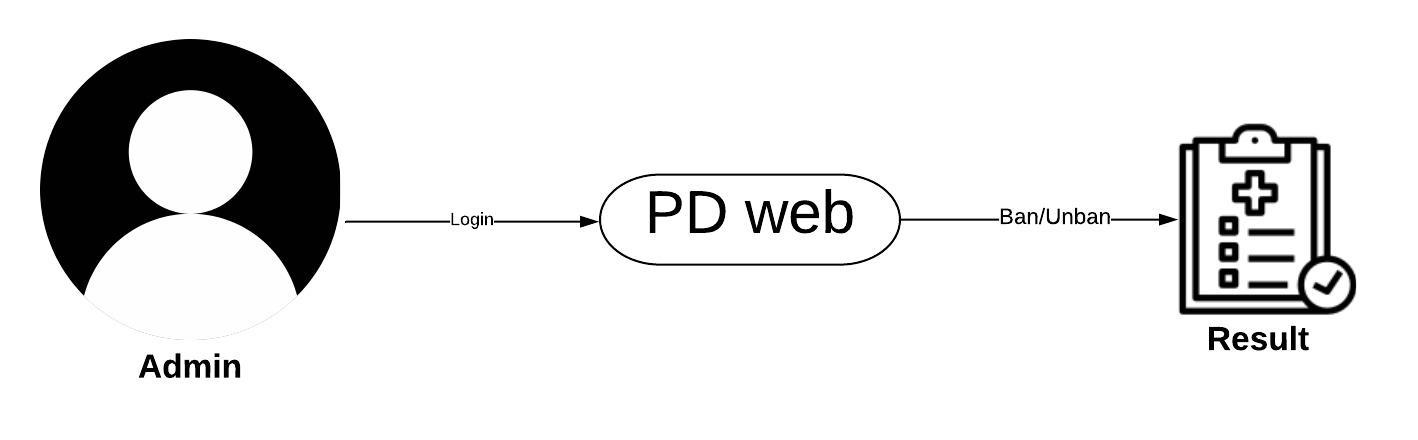


Figure 5 - Process of ban/unban on PD

* **Dating process**

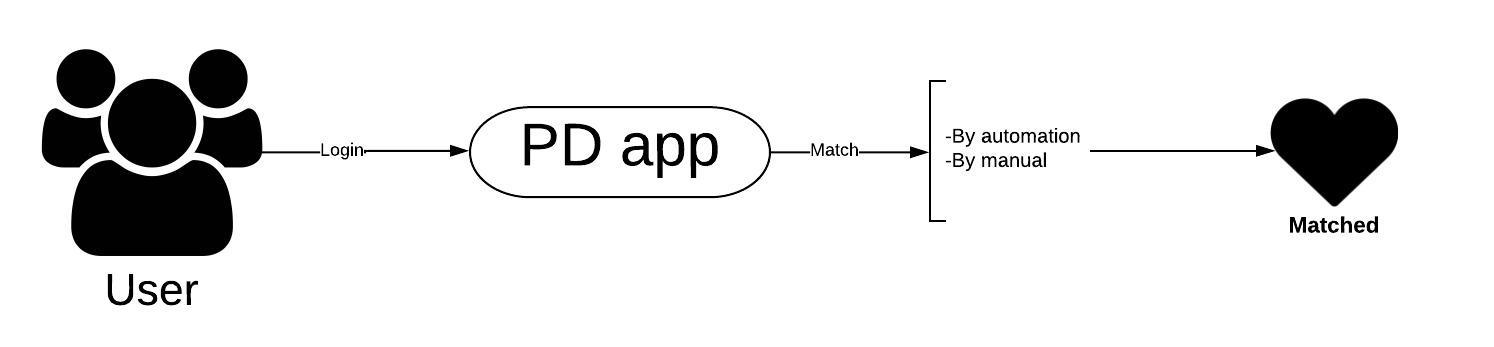


Figure 6 - Process of Dating on PD

* **User management process**

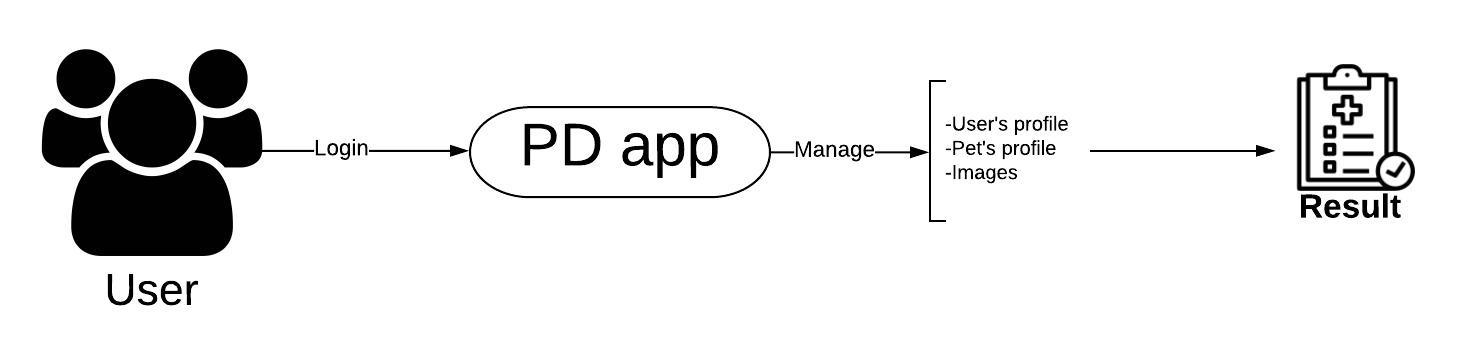


Figure 7 – Process of user management

# 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

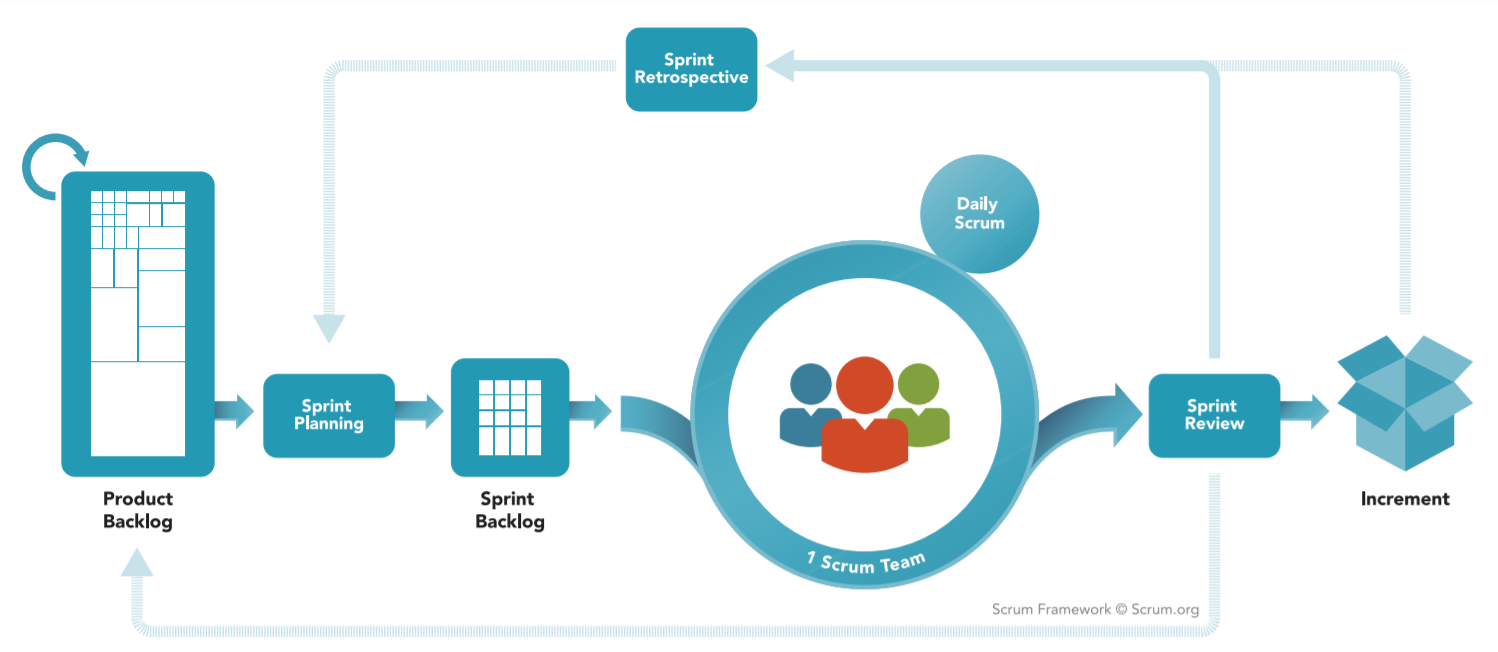


Figure 8 - Agile Model – SCRUM

### 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

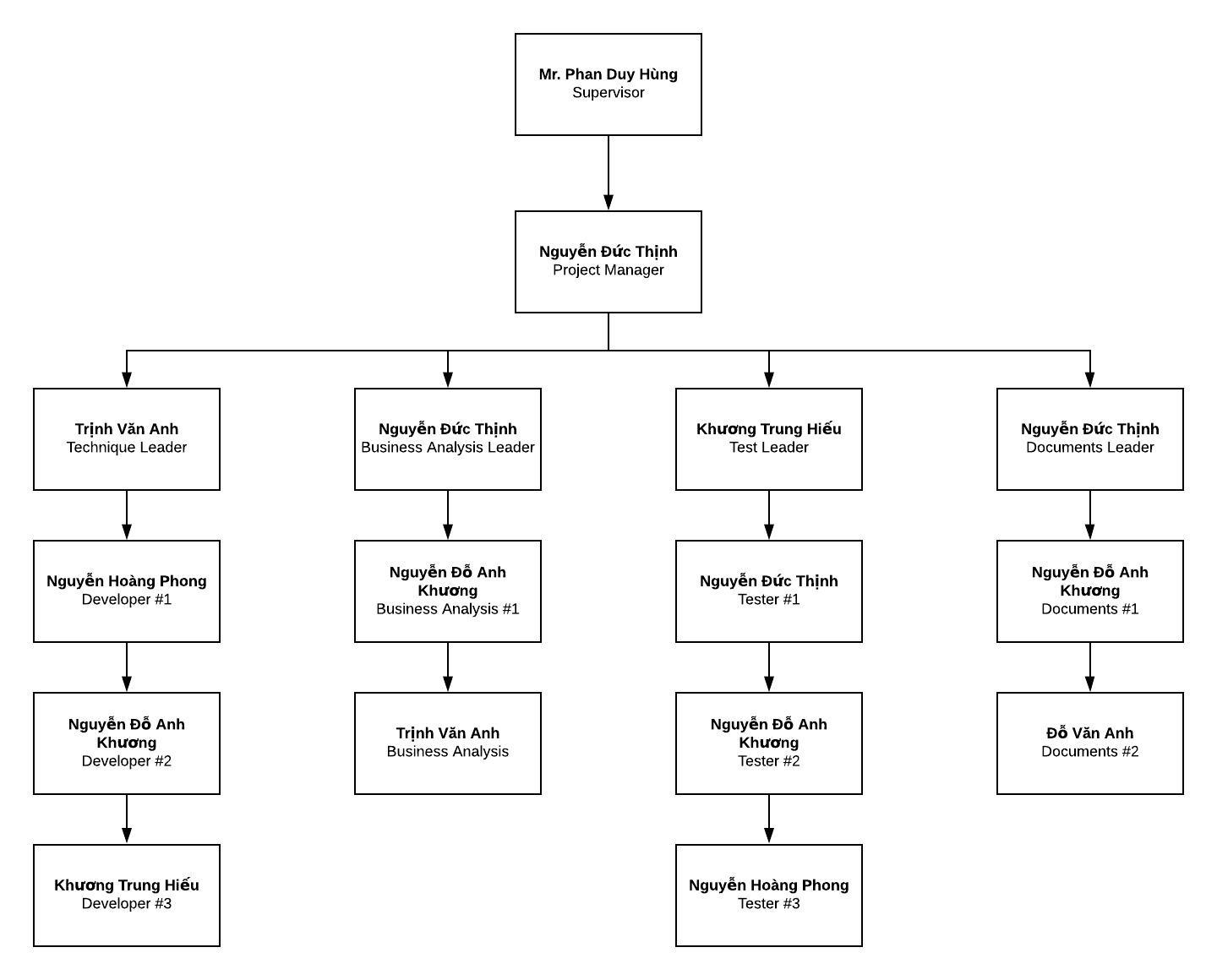


Figure - Organization structure

### 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. * Propose ideas or issues of the team to supervisor, university. | Nguyễn Đức Thịnh |
| **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

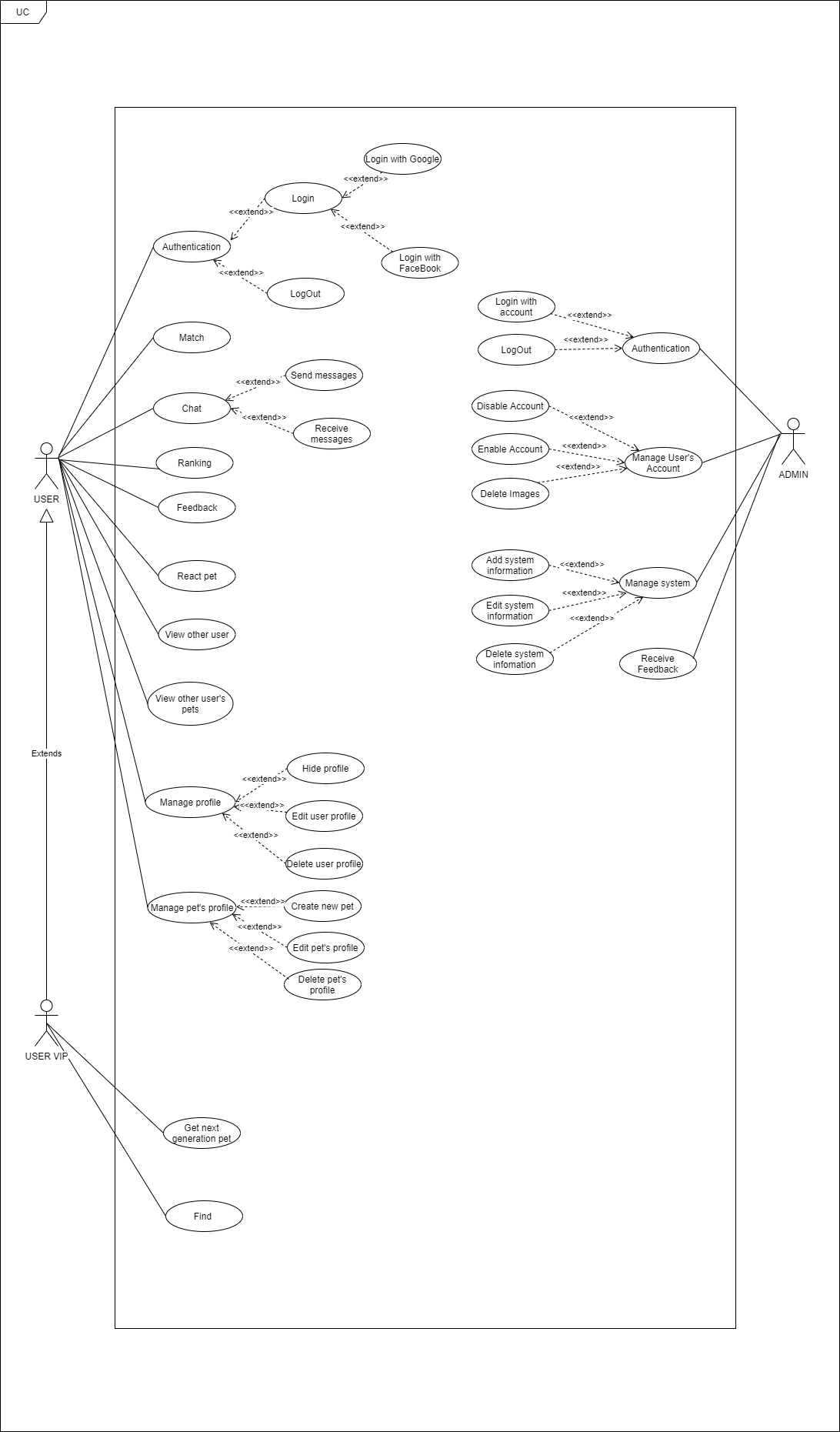


Figure 10 – Use case diagram

### .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

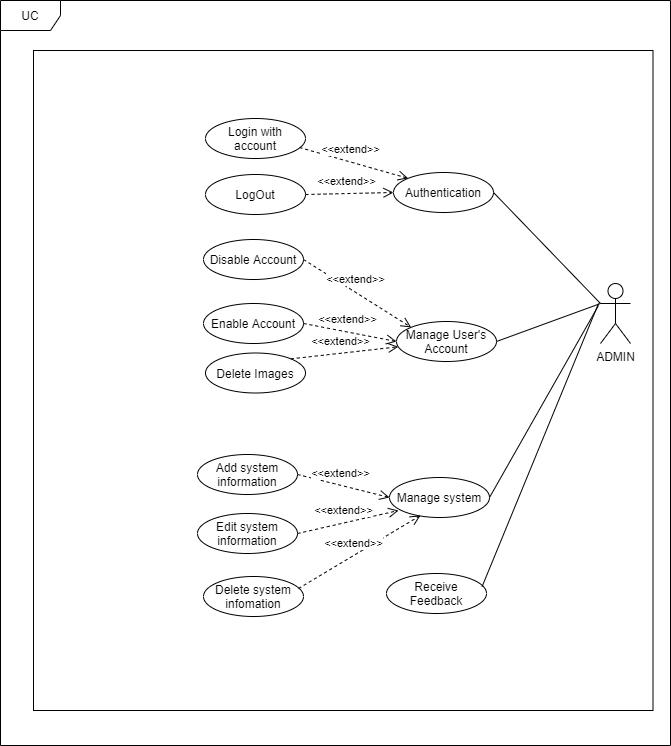


Figure 11 - Use case diagram of Admin actor

##### 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

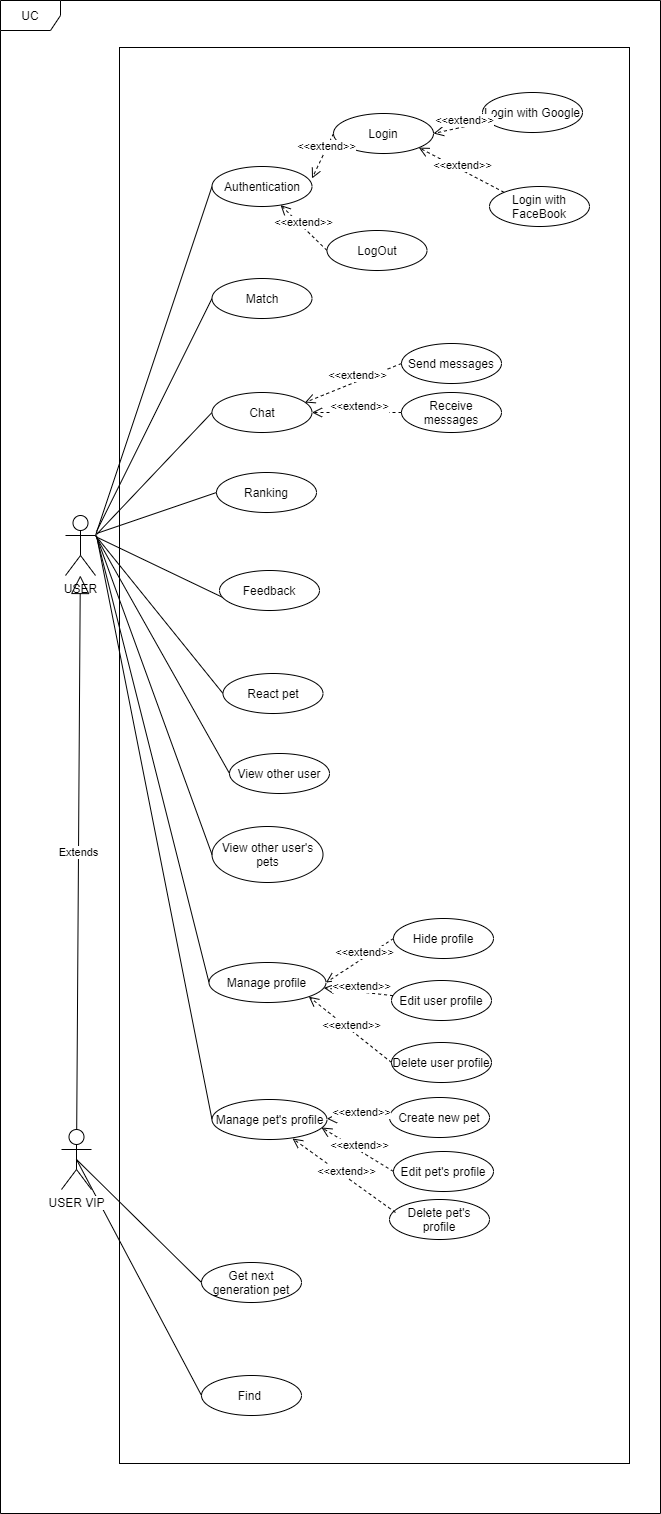


Figure 12 - Use case diagram of User actor

##### 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

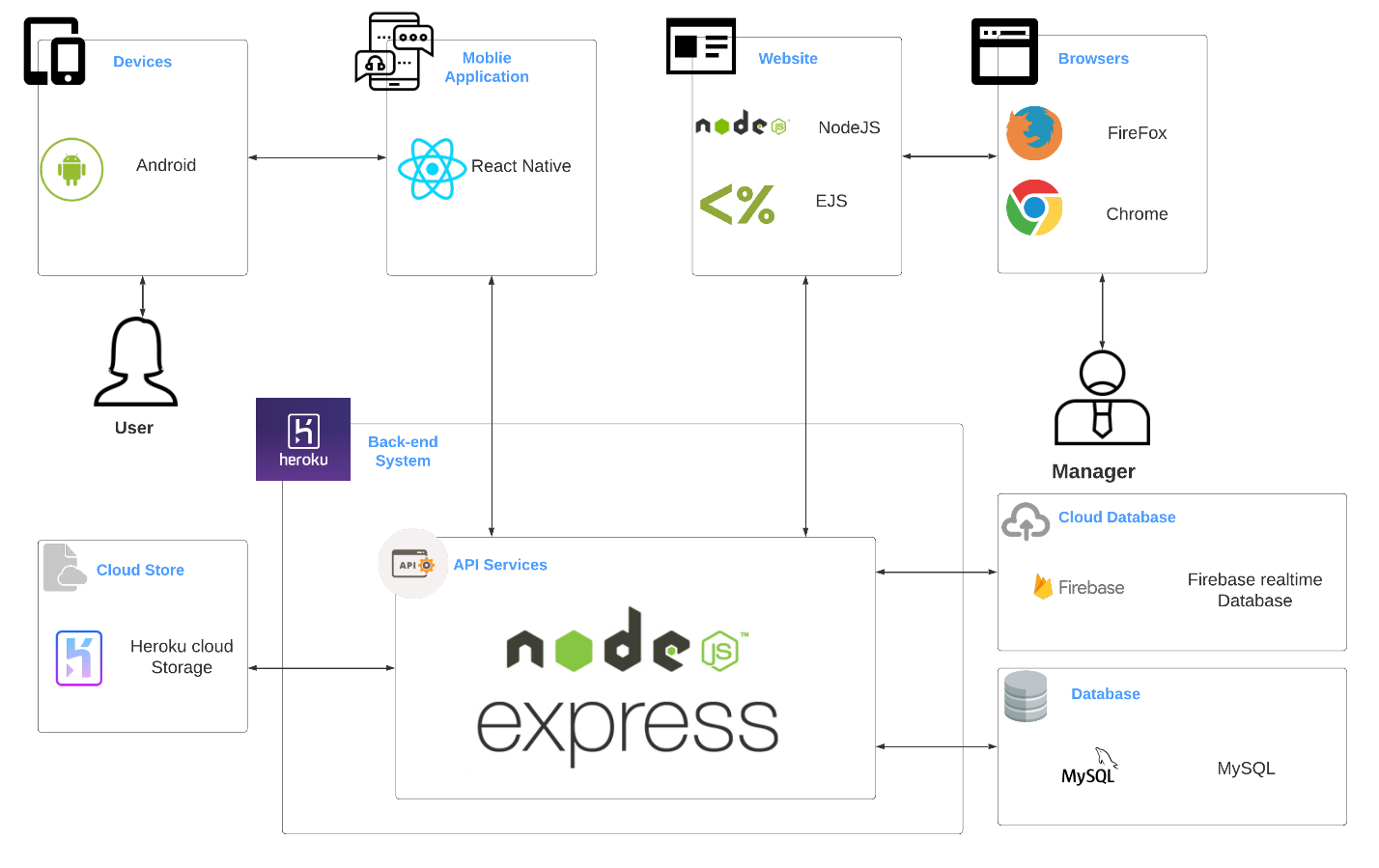


Figure 13 - PetDating system architecture

### 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

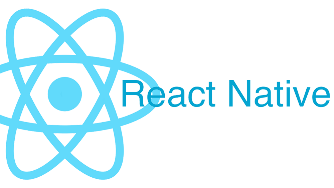


Heroku is a service provider like as hosting, VPS, Cloud, … We use Heroku to deploy API services, hosting for website.

##### 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**.**

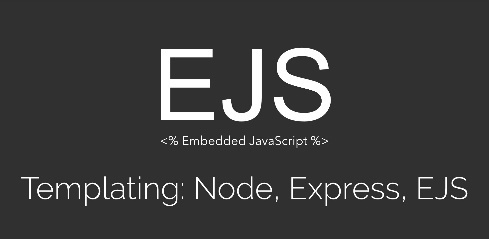
#### 4.2.2.3. Website components

##### 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

## 4.4. Database Design

### 4.4.1. Entity relationship diagram

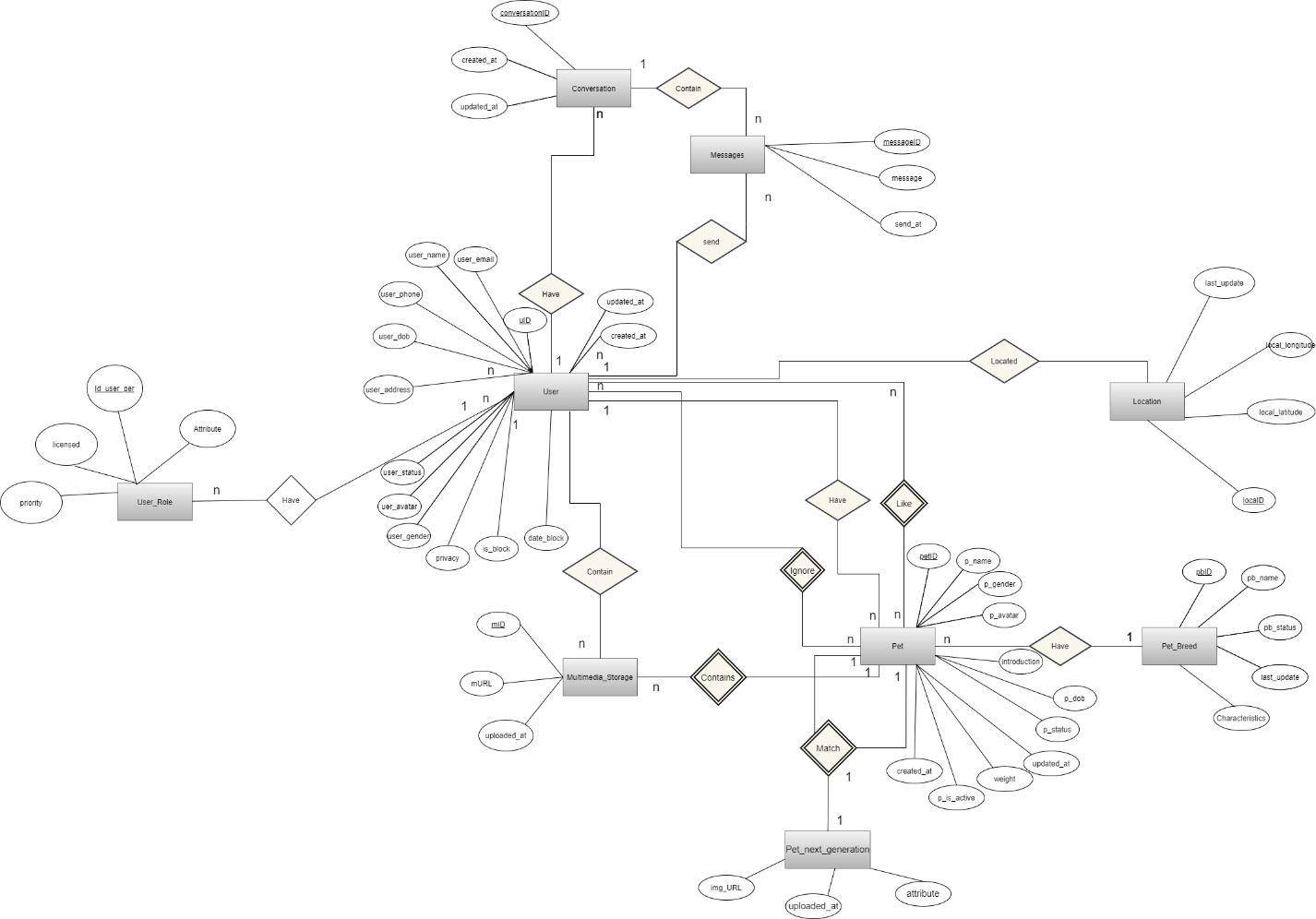


Figure 14 - Entity relationship diagram

|  |  |  |
| --- | --- | --- |
| **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

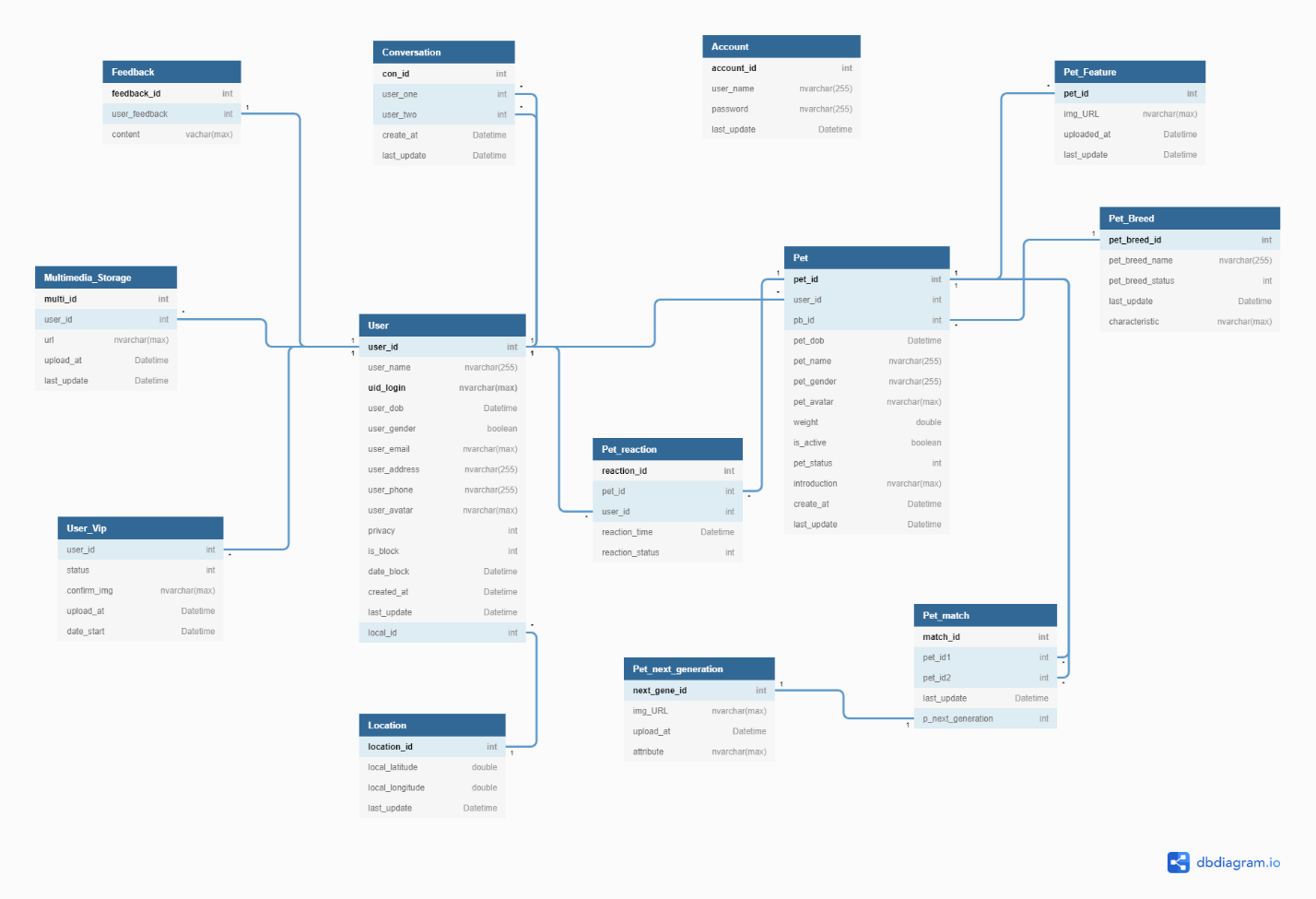


Figure 15 - Database diagram

**Database Explanation:**

**\*User:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Type** | **Allow Null** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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** | **Desceription** |
| 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**

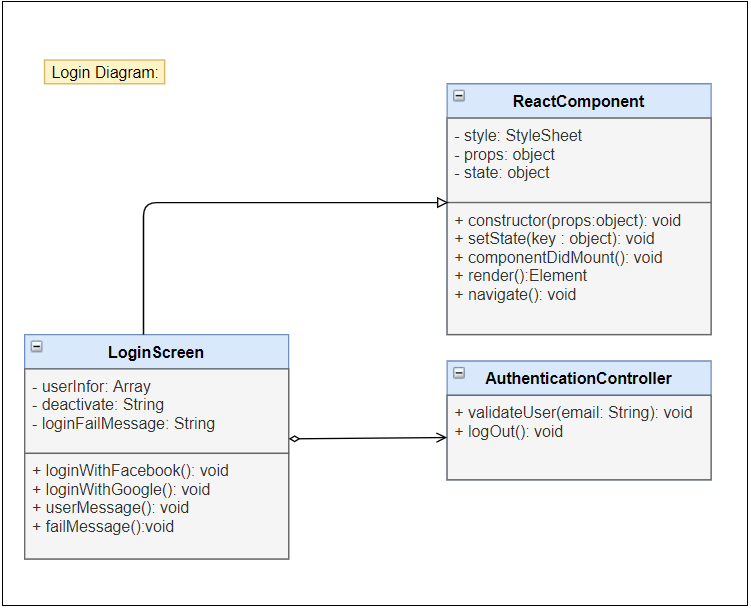
****

Figure 16 - 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**

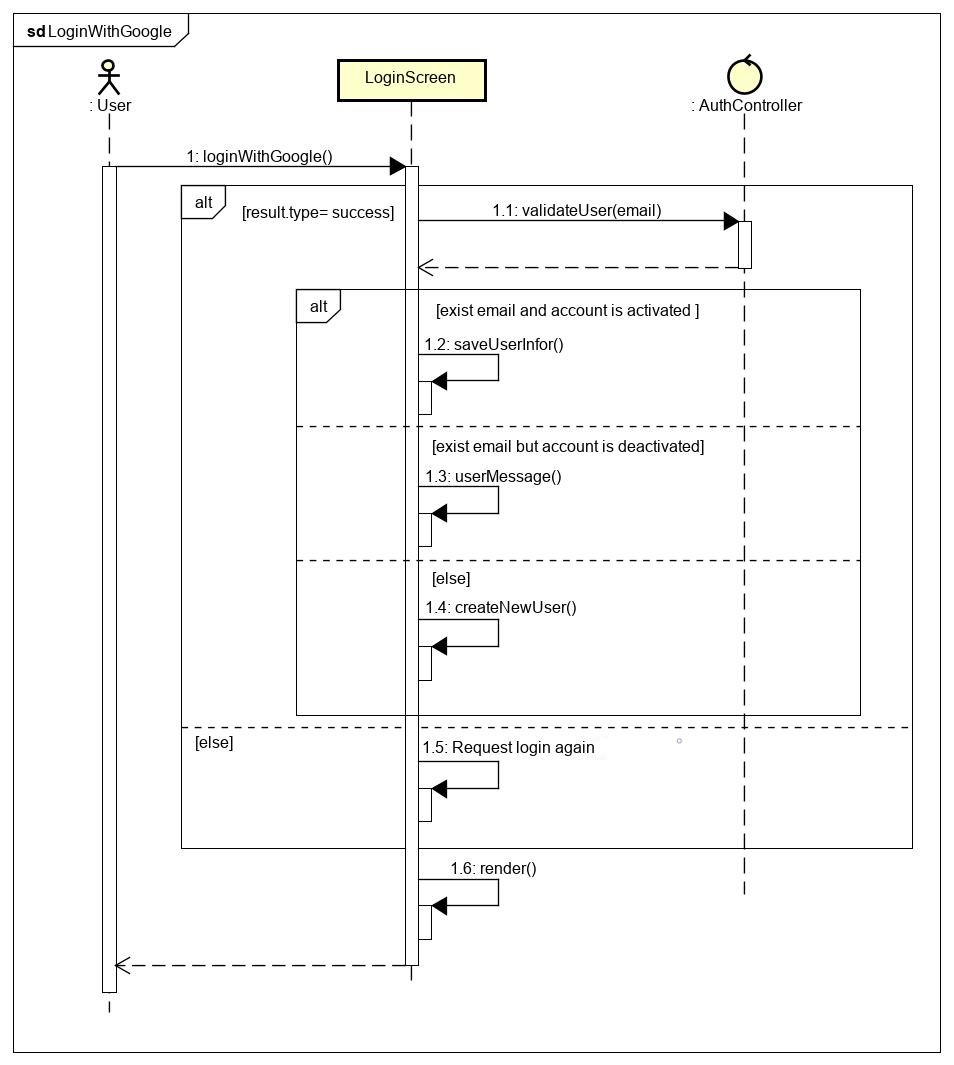


Figure 17 - Login with Google sequence diagram

**\*Login with Facebook sequence diagram**

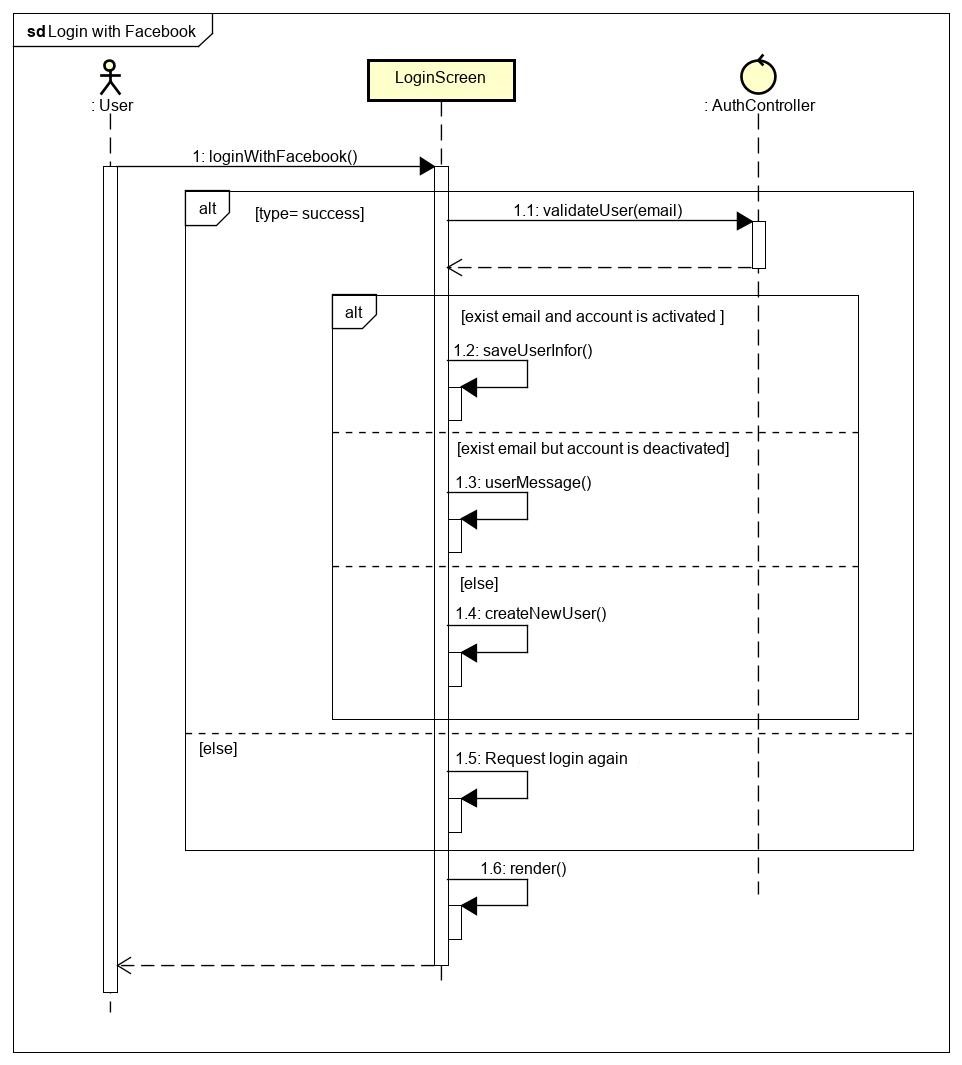


Figure 18 - Login with Facebook class diagram

### s4.5.2. Logout

**\*Logout class diagram**

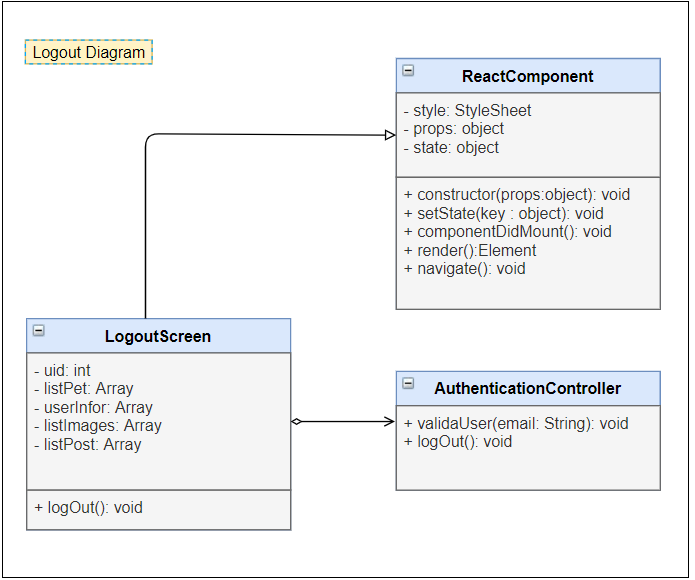
****

Figure 19 - 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**

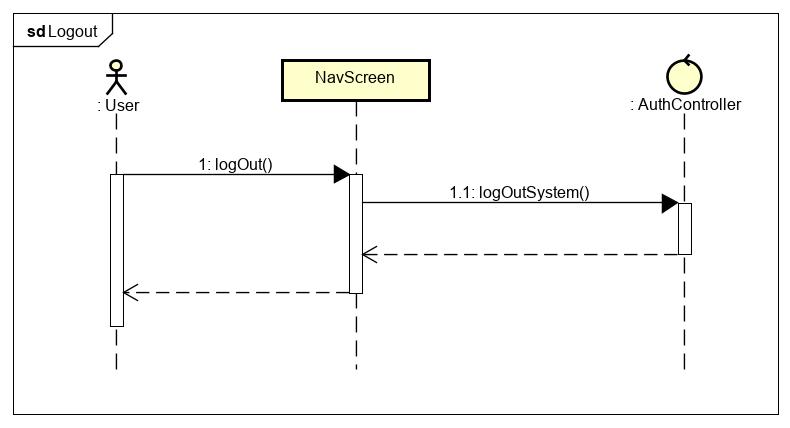


Figure 20 - Logout sequence diagram

### 4.5.3. Matching

**\* Matching Class Diagram**

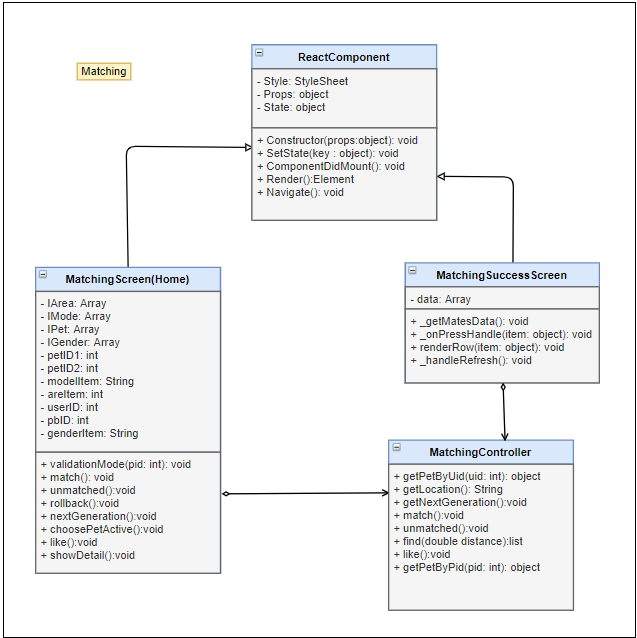
****

Figure 21 - 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**

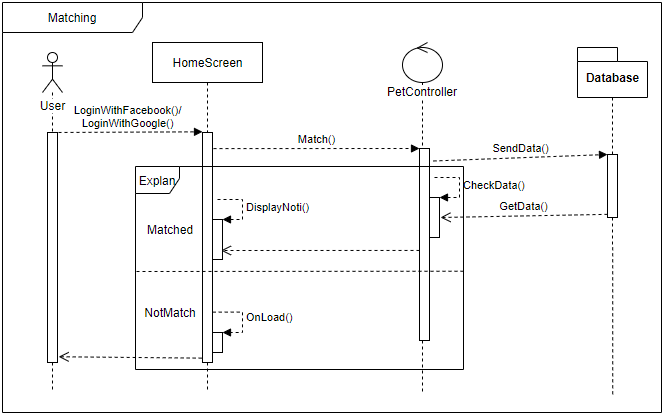
****

Figure 22 - Matching Sequencediagram

### 4.5.4. Chat

**\* View ConversationList Class Diagram**

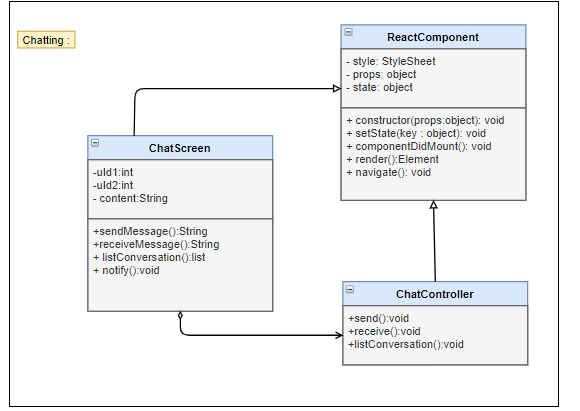
****

Figure 23 - 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 conver | | | sation |  |
| **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**

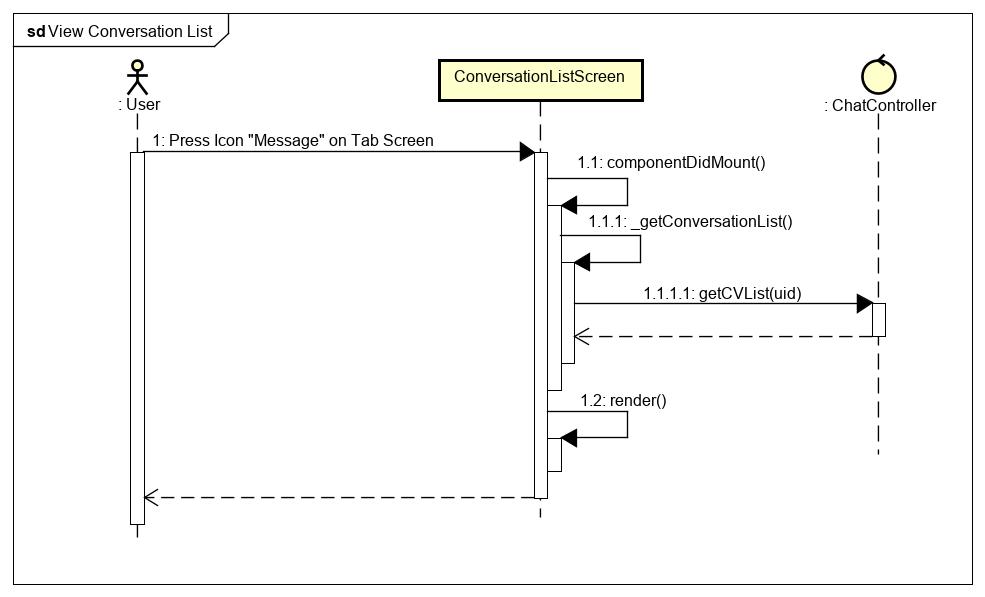


Figure 24 - Conversation list sequence diagram

**\*Send Messages class diagram**

****

Figure 25 - 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**

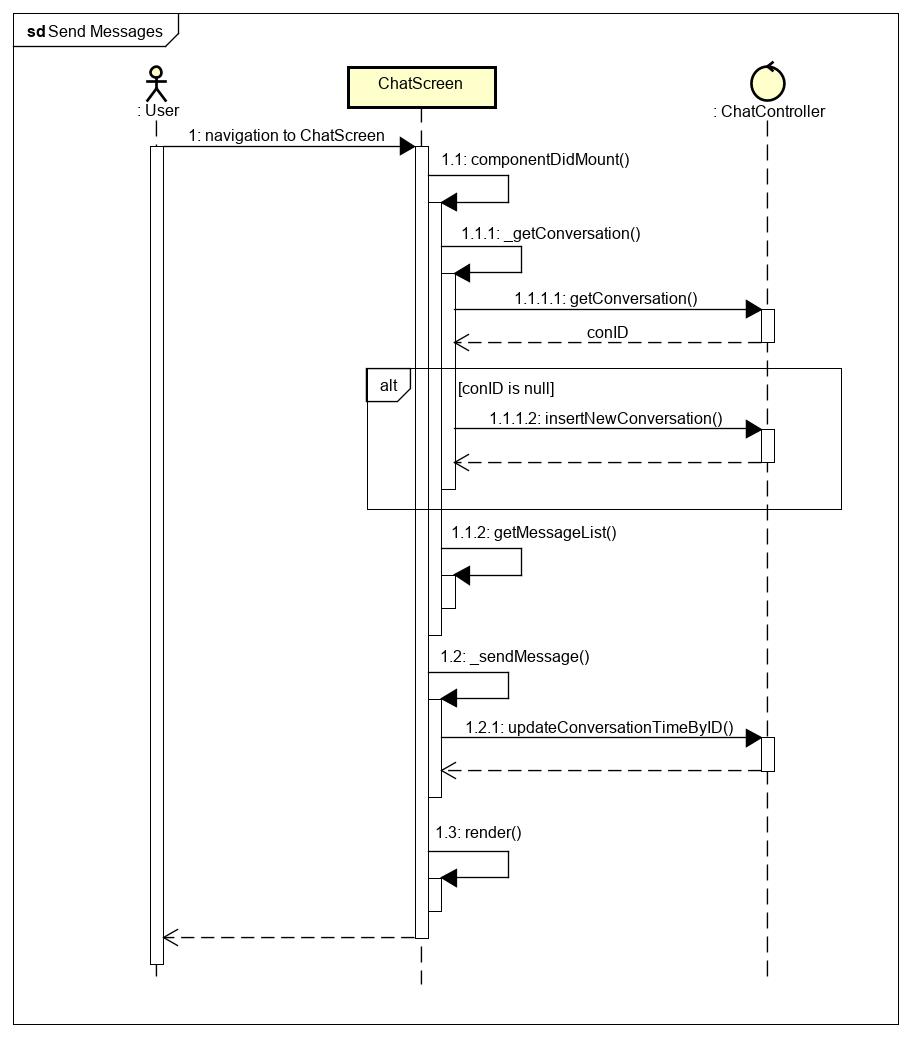
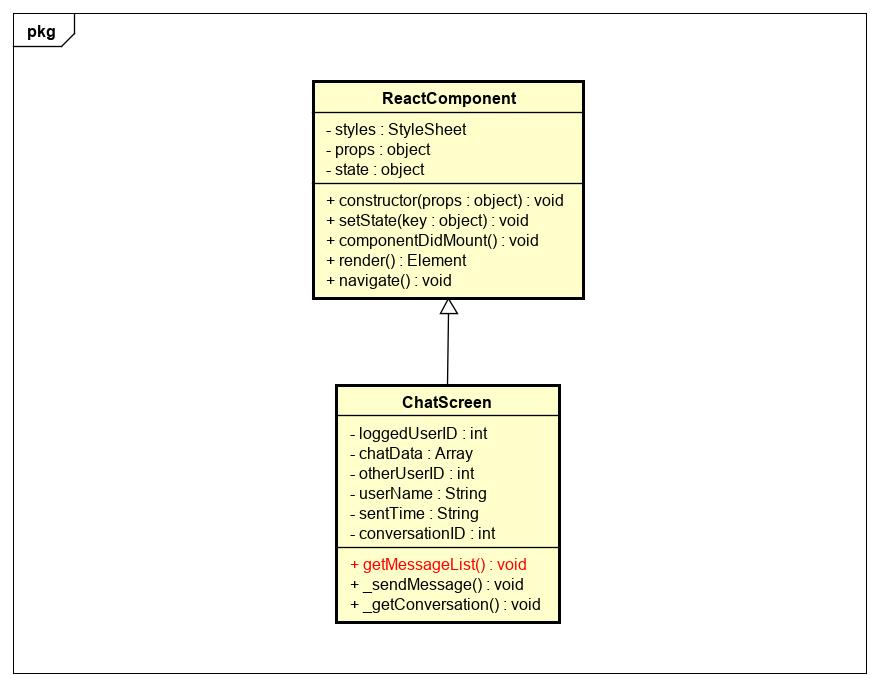
****

Figure 26 - Send message sequence diagram

**\*Receive Messages class diagram**

****  
Figure 27 - 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 28 - Receive message sequence diagram

### 4.5.5. Ranking

**\* Ranking class diagram**

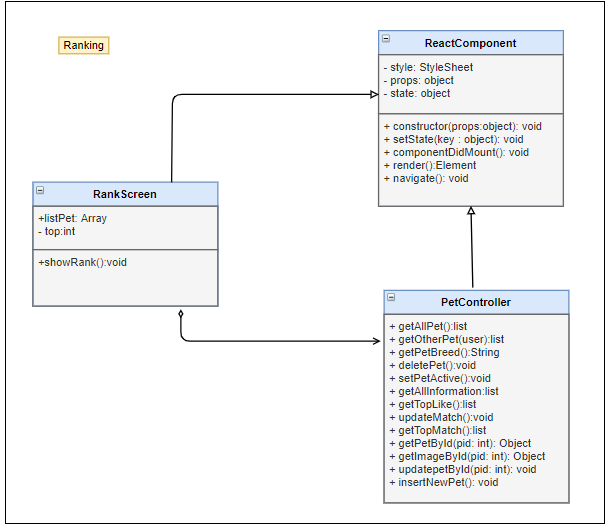
****

Figure 29 – 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**

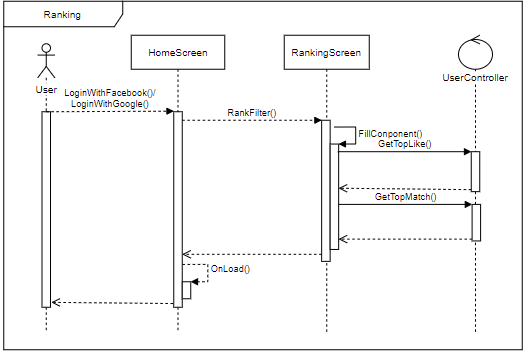


Figure 30 – Ranking sequence diagram

### 4.5.6. Feedback

**\* Feedback class diagram**

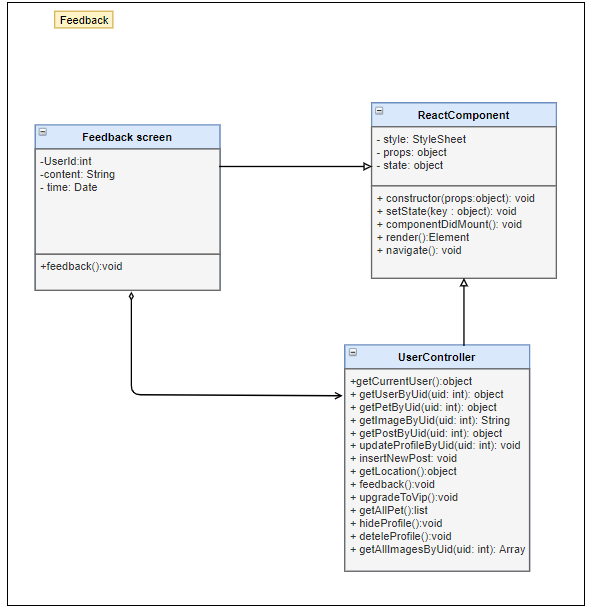
****

Figure 31 – 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**

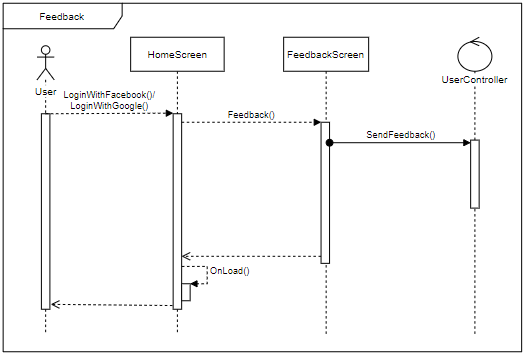


Figure 32 – Feedback sequence diagram

### 4.5.7. React pet

**\* React pet class diagram**

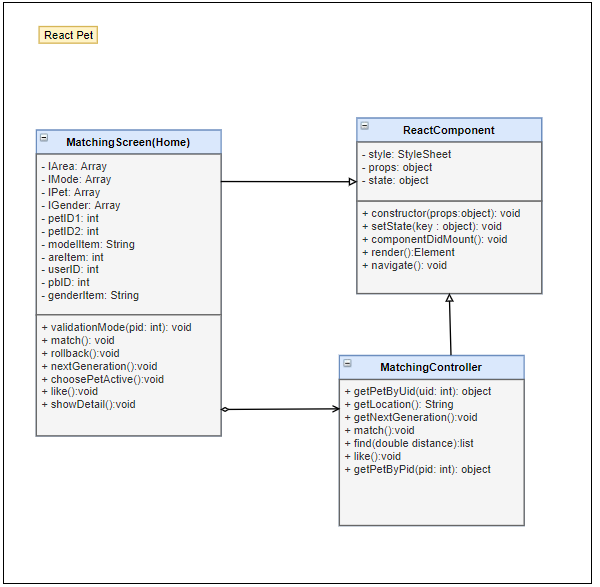
****

Figure 33 – 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**

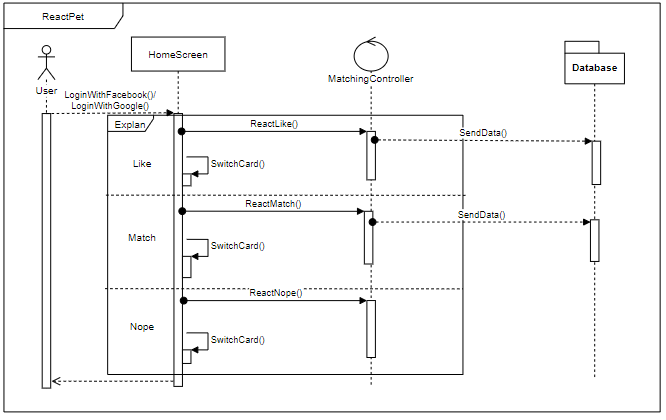


Figure 34 – React pet sequence diagram

### 4.5.8. View other user

**\* View other user class diagram**

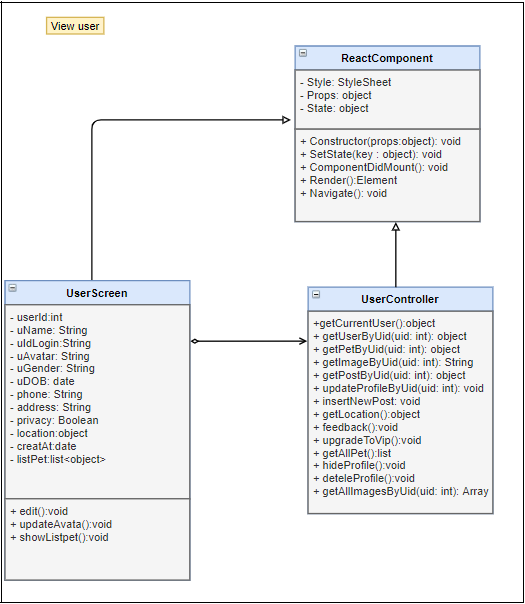
****

Figure 35 – 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**

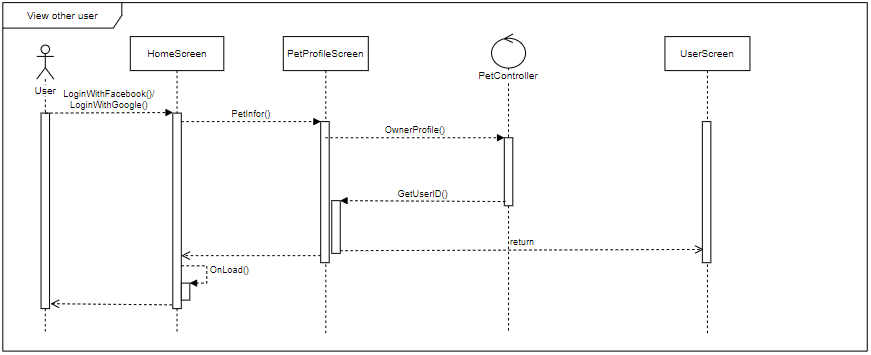


Figure 36 – View other user sequence diagram

### 4.5.9. View other user’s pet

**\* View other user’s pet class diagram**

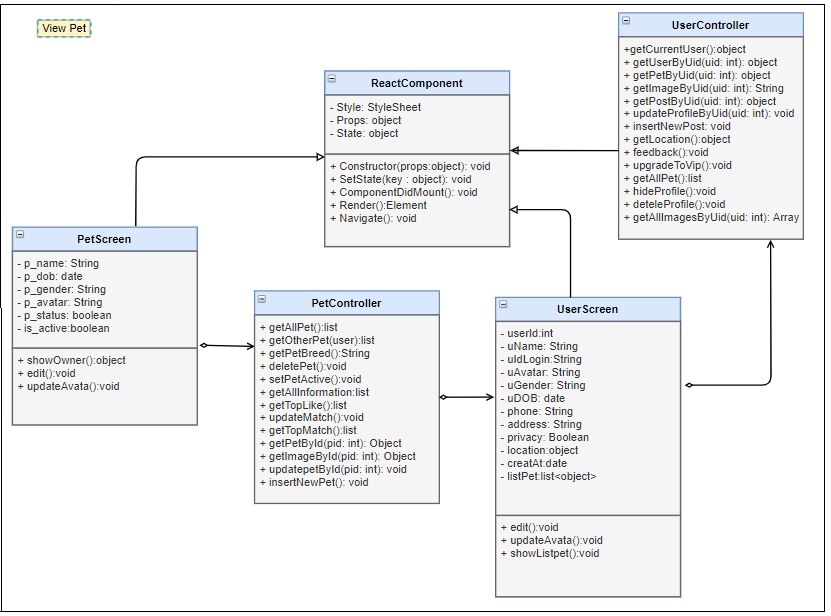
****

Figure 37 – 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**

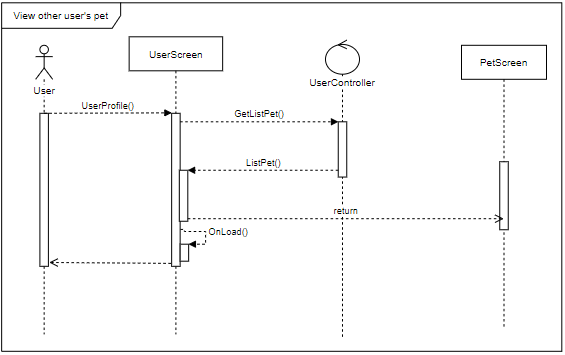


Figure 38 – View other user’s pet sequence diagram

### 4.5.10. Hide profile

**\* Hide profile class diagram**

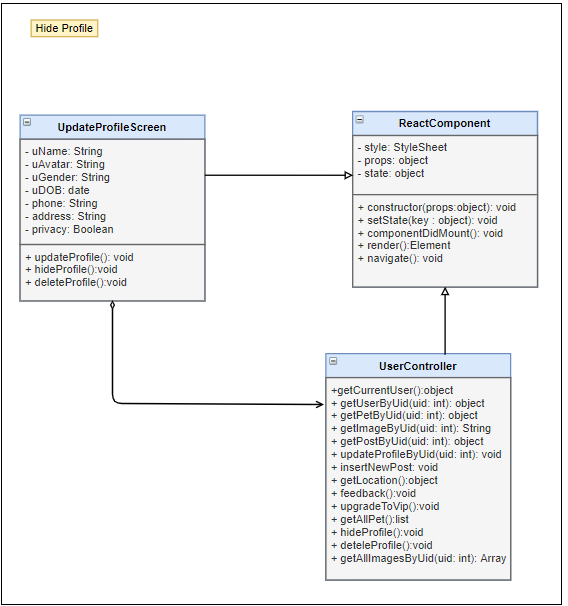
****

Figure 39 – 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**

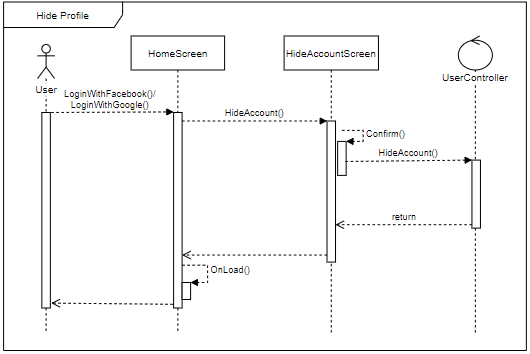


Figure 40 – Hide profile sequence diagram

### 4.5.11. Edit user’s profile

**\* Edit pet’s profile class diagram**

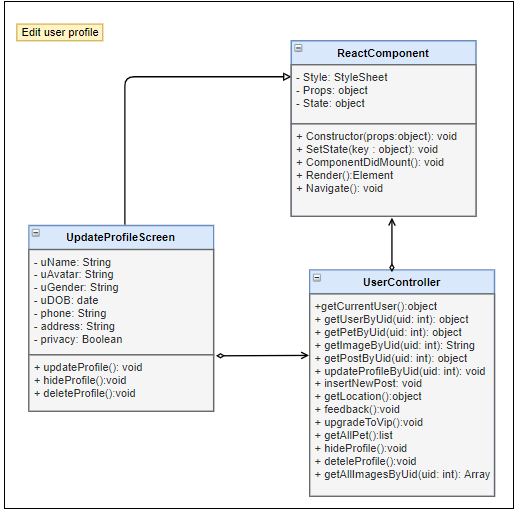
****

Figure 41 – 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**

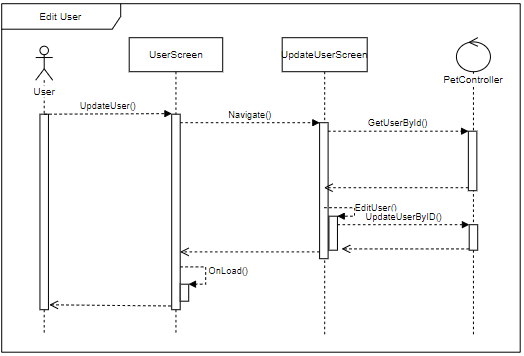


Figure 42 – Edit user’s profile sequence diagram

### 4.5.12. Delete user’s profile

**\* Delete user’s profile class diagram**

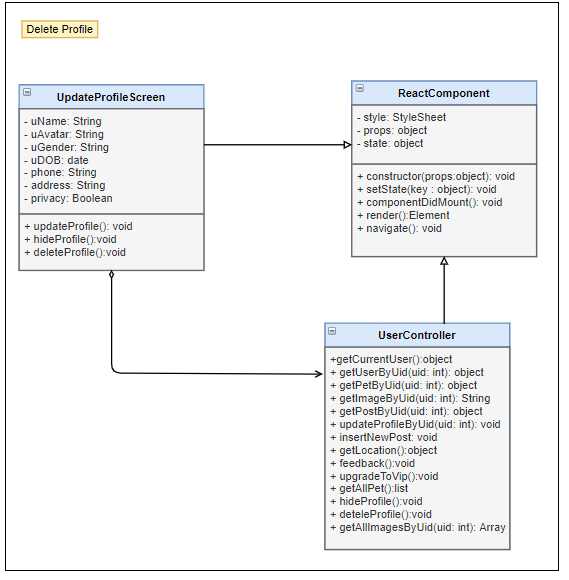
****

Figure 43 – 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**

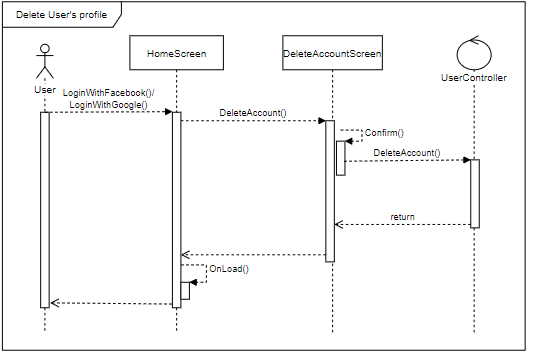


Figure 44 – Delete user’s profile sequence diagram

### 4.5.13. Create new pet

**\* Create new pet class diagram**

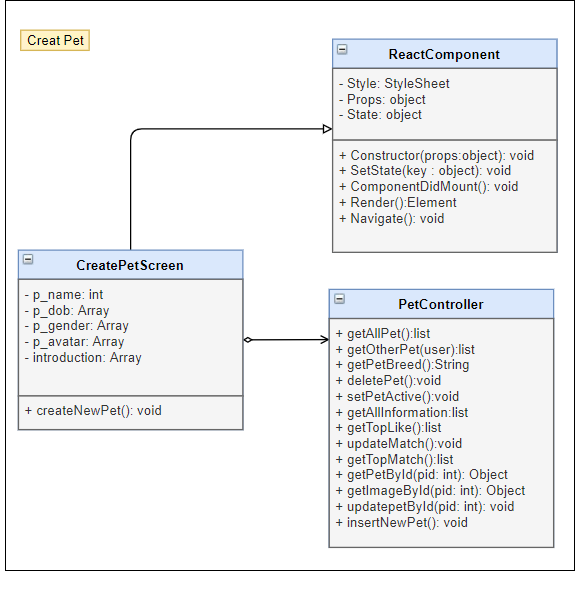
****

Figure 45 – 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**

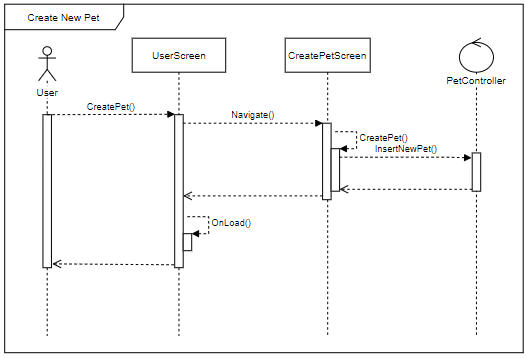


Figure 46 – Create new pet sequence diagram

### 4.5.14. Edit pet’s profile

**\* Edit pet’s profile class diagram**

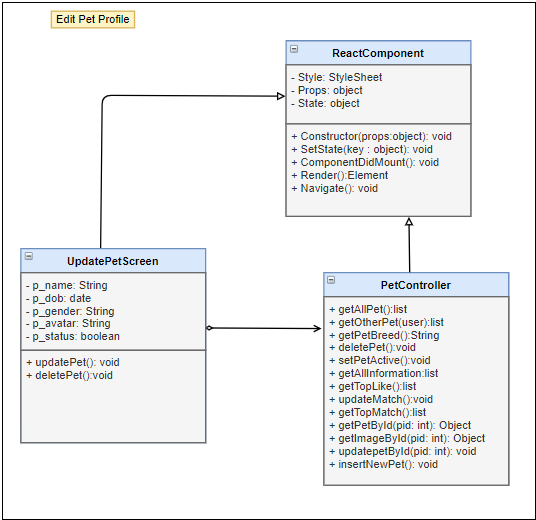
****

Figure 47 – 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**

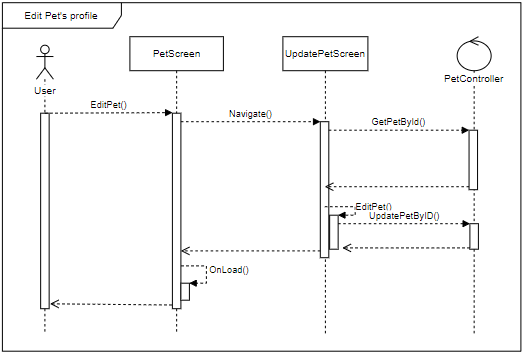


Figure 48 – Edit pet’s profile sequence diagram

### 4.5.15. Delete pet’s profile

**\* Delete pet’s profile class diagram**

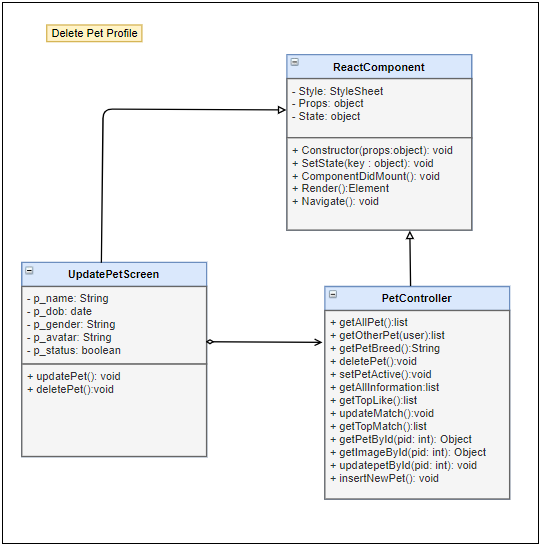
****

Figure 49 – 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**

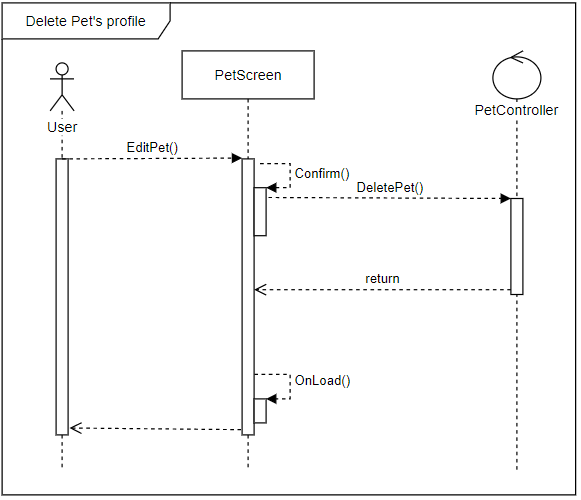


Figure 50 – Delete pet’s profile sequence diagram

### 4.5.16. Find

**\* Find class diagram**

****

Figure 51 – 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**

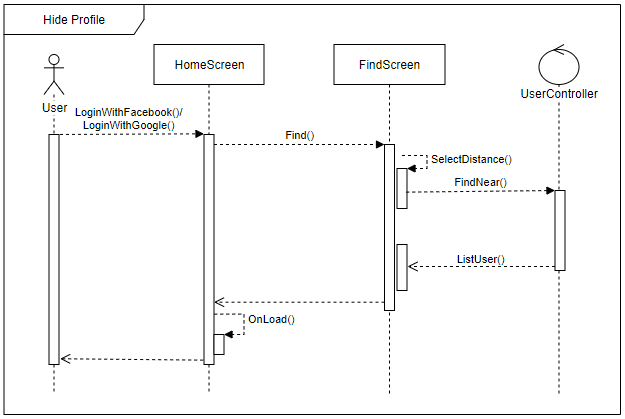


Figure 52 – Find sequence diagram

### 4.5.17. Get next generation pet

**\* Get new generation class diagram**

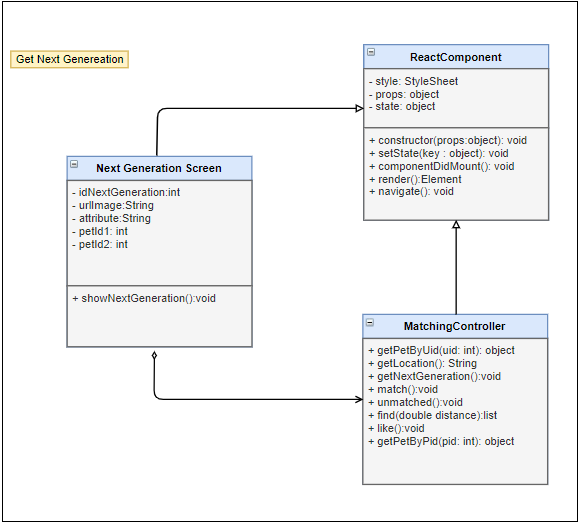
****

Figure 53 – 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 | | |

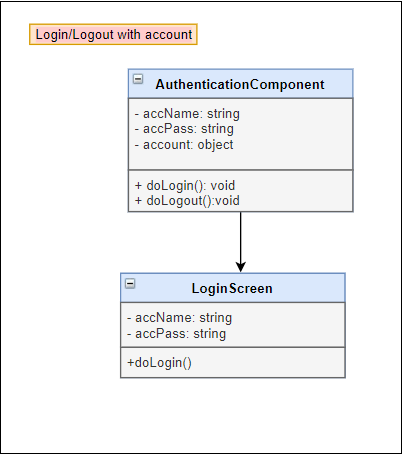
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** |  | | | **Type** | Class |
| **Description** |  | | | | |
| **Attributes** | | | | | |
| **Name** | **Type** | **Visibility** | **Description** | | |
| idNextGeneration | int | private | Id of pet is next generation | | |
| urlImage | 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 54 – Get new generation pet sequence diagram

### 4.5.18. Login/Logout for Admin

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

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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**

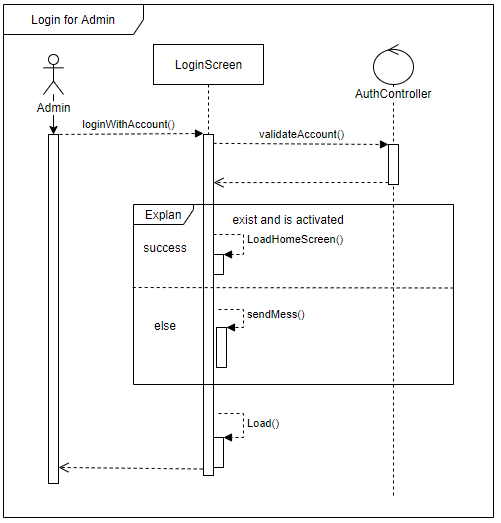
****

Figure 55 – Login for Admin sequence diagram

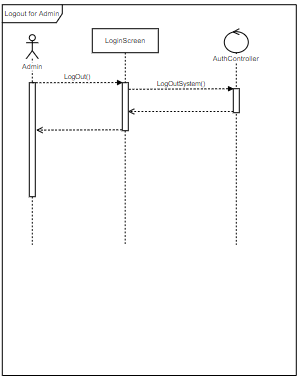
****

Figure 56 – Logout for Admin sequence diagram

### 4.5.20. Enable/Disable Account for Admin

**\* Enable/Disable account class diagram**

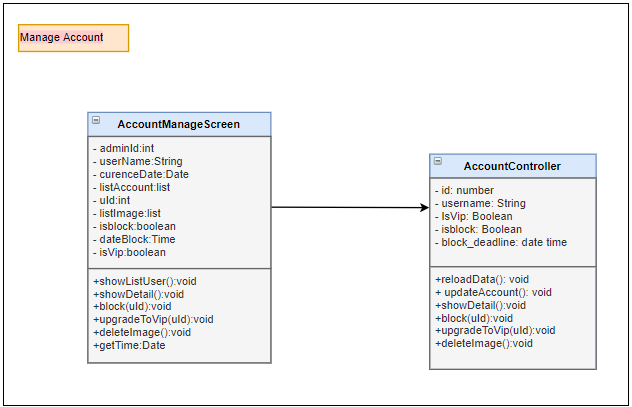
****

Figure 57 – 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**

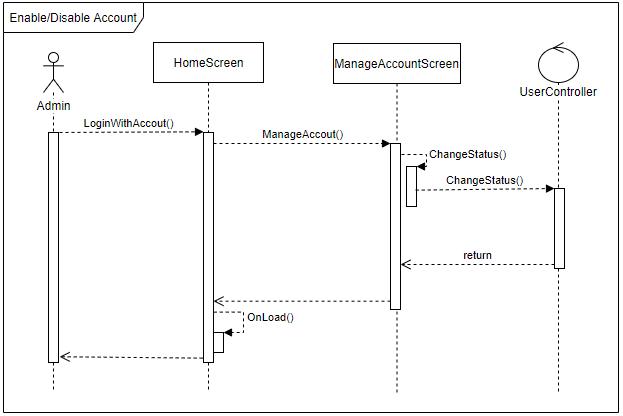


Figure 58 – Enable/Disable Account for Admin sequence diagram

### 4.5.22. Delete Images for Admin

**\* Delete Image class diagram**

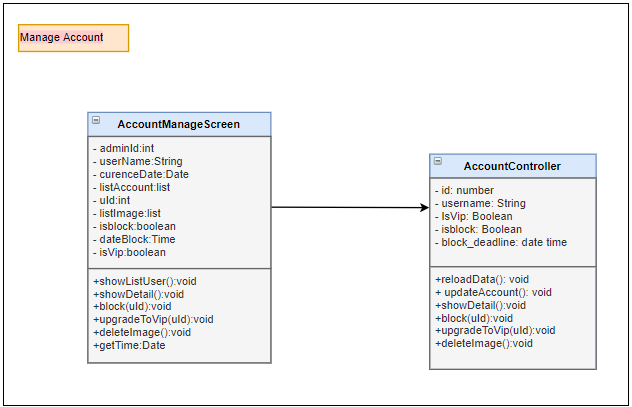
****

Figure 59 – 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**

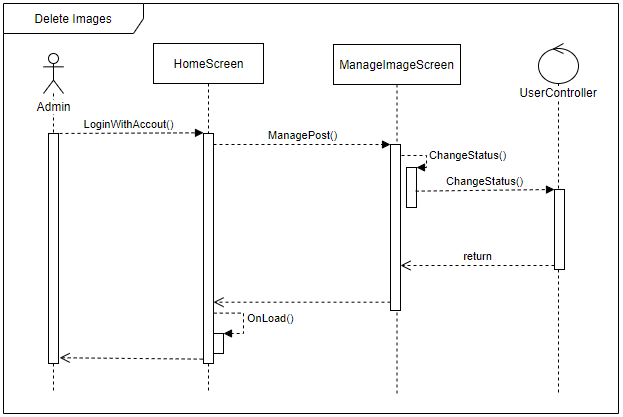


Figure 60 – Delete Images for Admin sequence diagram

### 4.5.23. Manage system for Admin

**\* Manage system class diagram**

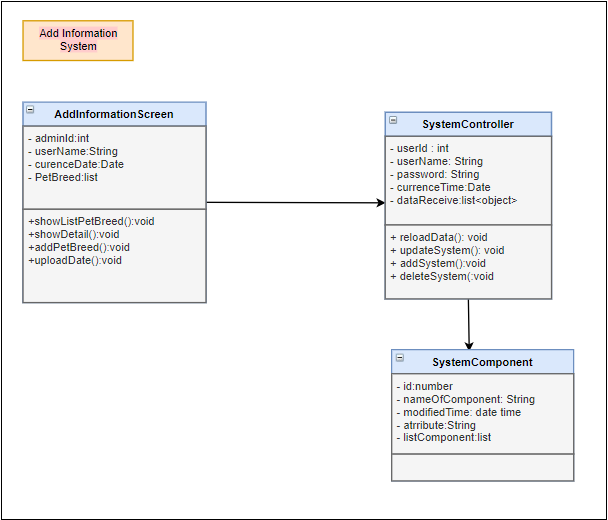
****

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**

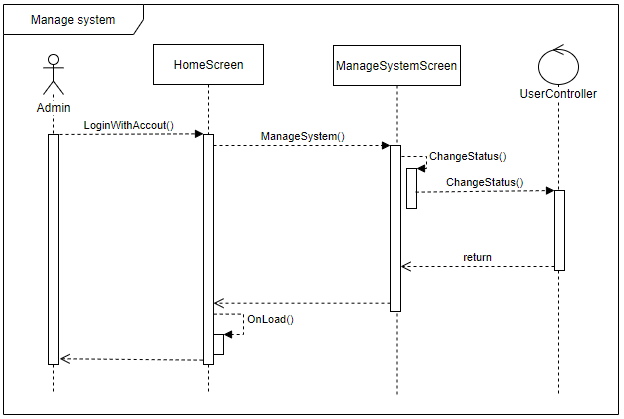


Figure 62 – Manage system for Admin sequence diagram

### 4.5.24. Receive Feedback for Admin

**\* Receive Feedback class diagram**

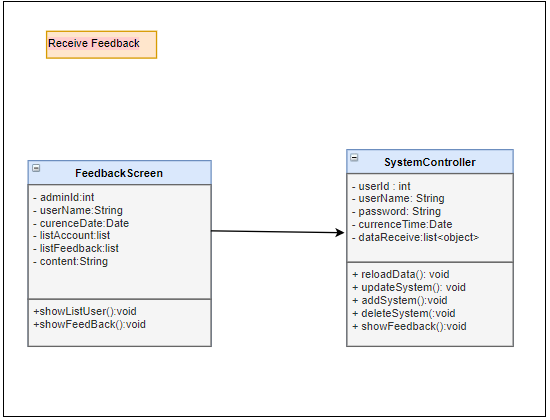
****

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**

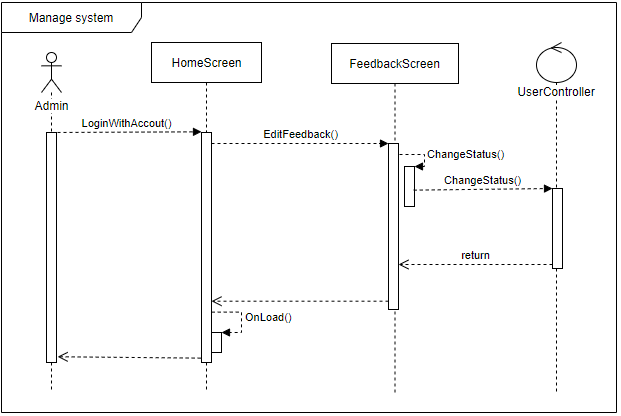
****

Figure 64 – Receive Feedback for Admin sequence diagram

# CHAPTER 5: Software Testing Document

## 5.1. Introduction

### 5.1.1. Purpose

### 5.1.2. Scope of testing

### 5.1.3. Range of testing

## 5.2. Test plan

### 5.2.1. Testing Tools and Environment

### 5.2.2. Resources and Responsibilities

### 5.2.3. Test Strategy

### 5.2.4. Features to be tested

## 5.3. Test case

### 5.3.1. Unit test

### 5.3.2. System testing

### 5.3.3. Acceptance test

### 5.3.4. Defect Log

## 5.4. Test Report

### 5.4.1. Unit test

### 5.4.2. Integration and system tests

# CHAPTER 6: USER MANUAL

## 6.1. Development and deployment guidelines

### 6.1.1.1. Environment for development